iMoby - Intelligent Mobility: Abschlussbericht zum Ende des Projektes

DFKI Robotics Innovation Center

2. August 2012

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Wirtschaft und Technologie unter dem Förderkennzeichen 50 RA 0907 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Deutsches Forschungszentrum für künstliche Intelligenz
Robotics Innovation Center
Prof. Dr. Kirchner
Robert-Hooke-Str. 5
28359 Bremen, Germany

Inhaltsverzeichnis

1 Einleitung 3

2 Kurzdarstellung nach NKBF 98 8.2 I 4
2.1 Aufgabenstellung 4
2.2 Voraussetzungen, unter denen das Vorhaben durchgeführt wurde 4
2.3 Planung und Ablauf des Vorhabens 5
2.4 Wissenschaftlicher und Technischer Stand zu Beginn des Vorhabens 8
2.4.1 Stand der Technik zu Beginn des Vorhabens 8
2.4.2 Fachliteratur und verwendete Dokumentationsdienste 11
2.5 Zusammenarbeit mit anderen Stellen 13

3 Eingehende Darstellung 14
3.1 AP1000: Projektmanagement 14
3.2 AP2000: Design, Integration und Test 19
3.2.1 Meilensteine 28
3.3 AP3000: Plattform-Mobilität 32
3.4 AP4000: Embodied SLAM 72
3.5 AP5000: Autonome Navigation 99

4 Fazit und Ausblick 107

5 Appendix 107
5.1 Classifying Autonomous Systems 108
5.2 Terrain Characterization 119
5.3 iMoby Toolchain 137
5.4 Test Kommunikationsinfrastruktur 146
5.5 Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard 157
5.6 Envire Architektur 165
5.7 Selbstlokalisation auf Basis direktor Propriozeptiver Daten 178
5.8 Selbstlokalisation auf basis indirekter propriozeptiver Daten 184
5.9 Integration von visuellem und embodied SLAM 190
5.10 Planer/Planer integration 197
5.11 Qualitative Pfadplanung 204
5.12 Spezifikation und online Anpassung der Komponentenebene 217

1 Einleitung

Das vorliegende Dokument stellt den Abschlussbericht des Projektes ”Intelligent Mobility” (iMoby) dar. iMoby wurde durch das Bundesministerium für Wirtschaft und Technologie (BMWi, Förderkennzeichen 50 RA 0907) gefördert.

Dieser Abschlussbericht folgt dabei dem in der Anlage 2 zum NKBF 98 gegebenen Muster, in Kapitel 2 wird eine kurze Darstellung des Projektes gemäß Punkt I gegeben. Kapitel 3 stellt die inhaltlichen Entwicklungen in einer eingehenden Darstellung gemäß Punkt II des Musters dar. Die Punkte III (Erfolgskontrollbericht) und IV (Berichtsblatt bzw. Document Control Sheet) werden durch gesondert abgegebene Dokumente erfüllt.

2 Kurzdarstellung nach NKBF 98 8.2 I

In diesem Kapitel wird zunächst die Aufgabenstellung des Projektes beschrieben (Abschnitt 2.1). Anschließend werden die Voraussetzungen des Vorhabens (Abschnitt 2.2) sowie Planung und Ablauf des Vorhabens (Abschnitt 2.3) behandelt. Es folgen Erläuterungen zum wissenschaftlichen Stand (Abschnitt 2.4) und zur Zusammenarbeit mit anderen Stellen (Abschnitt 2.5).

2.1 Aufgabenstellung

Ziel des Projektes „Intelligent Mobility“ ist die Entwicklung neuer Methoden zur robusten autonomen Navigation robotischer Systeme durch Nutzung ihrer propriozeptiven Sensorik zur Selbstlokalisation, Kartengenerierung und Planung.

Es soll eine passende Hardware-Plattform auf der Grundlage des vom DFKI entwickelten hochmobilen und geländegängigen Asguard-Roboters erstellt werden. Die Leistungsfähigkeit der Hardware wird durch Software optimiert, und unterschiedliche Bewegungsmodalitäten werden auf verschiedene Umweltgegebenheiten angepasst. Es werden Modelle entwickelt, die von den Algorithmen zur Selbstlokalisation und Kartengenerierung (SLAM) benötigt werden, ebenso wie Algorithmen zur autonomen Planung dieser Modalitäten.

Die Daten der propriozeptiven Sensorik werden in die Lokalisation und Kartengenerierung eingebunden. Das Ergebnis ist ein „eSLAM“ Algorithmus, welcher in der Lage ist, durch die direkte Wahrnehmung seiner Umwelt unklare oder falsche Positionsangaben zu korrigieren und dadurch eine zuverlässige Selbstlokalisation und Kartengenerierung auch in für visuelle Methoden schwierigen Situationen zu ermöglichen.

Schließlich werden die propriozeptive Sensorik und eSLAM mit der Entscheidungsfindung verknüpft. Dadurch werden Vorhersage und Fehlerbehebung im Navigations-Subsystem ermöglicht und sein robuster und sicherer Betrieb verbessert.

Die Ergebnisse sollen durch kontinuierliche Integrationsarbeit integriert und demonstriert werden.

2.2 Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Das iMoby-Projekt basiert in weiten Teilen auf den Arbeiten, die vor dem Projekt an dem Asguard-System im Zuge der eigenfinanzierten Projekte SentryBot II sowie Intelligent Mobility (intern) entstanden sind. Das ursprünglich für den Sicherheitsbereich entwickelte Asguard-v2-System zeichnet sich durch hohe Flexibilität und Geländefähigkeit sowie geringe mechanische Komplexität und Masse aus. Diese Eigenschaften sind bei Explorationssystemen in Weltraumanwendungen besonders gefragt. Bis zu diesem Zeitpunkt wurde das Asguard-v2-System vollständig vom Operator ferngesteuert. Für die Verwendung im Weltraumbereich war eine Erhöhung der autonomen Fähigkeiten wünschenswert. Weitere Erfahrungen mit der Nutzung von Explorationssystemen sind bei der Entwicklung des CESAR-Systems und der Teilnahme an der

ESA Lunar Robotic Challenge in die Entwicklung des Projekts eingeflossen.

2.3 Planung und Ablauf des Vorhabens

Das Intelligent-Mobility-Projekt war anfangs mit einer Laufzeit von 36 Monaten geplant. Der Projektbeginn war der 1.4.2009. Die Überprüfung der Ziele sollte durch drei Meilensteine jeweils zum Ende eines Projektmonats erfolgen. Der generelle Strukturplan der Arbeitspakete ist in Abbildung 1 dargestellt. Abb. 2 zeigt die zeitliche Anordnung der Arbeitspakete. Die inhaltlichen Pakete 3000 - 5000 werden dabei parallel behandelt und durch die fortlaufenden Arbeitspakete für Management (1000) sowie Integration und Tests (2000) unterstützt.

Am 19.5.2010 ist der erste Meilenstein erfolgreich dem DLR-Projektmanagement in einer Live-Demonstration am DFKI präsentiert worden. Hier ist das vollständig integrierte Asguard-v3-Modell nach Plan demonstriert worden. Dabei wurde mit Hilfe des dGPS-Systems und einer a-priori-Karte die DFKI-Teststrecke autonom durchquert.

Der zweite Meilenstein des iMoby-Projektes ist dem Projektträger am 16.6.2011 in Form einer Videopräsentation vorgestellt worden. Der Inhalt des zweiten Meilensteins war die autonome Navigation auf der Teststrecke ohne Zuhilfenahme von GPS-Daten. Die Ziele des Meilensteins sind nach Absprache mit dem Fördergeber im Bezug auf den ursprünglichen Plan abgeändert worden. Der Ursprungsplan sah vor, bereits ohne a-priori-Karte auszukommen. Aufgrund der problematischen Situation mit der Nutzung von GPS auf der Teststrecke des DFKI wurde auf die Nutzung von GPS verzichtet, aber noch weiterhin a-priori-Information verwendet.

Aufgrund von technischen Problemen mit den Radenkodern des Asguard-v3-Systems gab es inhaltliche Verzögerungen bei der Durchführung von Außentests. Die Verzögerungen wirkten sich hauptsächlich auf das AP3300 und als Folge auf AP4300 aus. Die problematischen Witterungsbedingungen für Außentests in den Wintermonaten hatten zur Folge, dass eine kostenneutrale Verlängerung des Projektes um drei Monate beantragt und vom Mittelgeber genehmigt wurde. Abb. 3 zeigt den geänderten Gantt-Plan.

Das Projekt ist am 30.6.2012 beendet worden. Der Meilenstein 3 wurde am 8.8.2012 in Form einer Videopräsentation beim Projektträger vorgestellt. Der Inhalt des dritten Meilensteins ist die autonome Navigation des Testgeländes ohne Zuhilfenahme von GPS oder a-priori-Kartendaten.

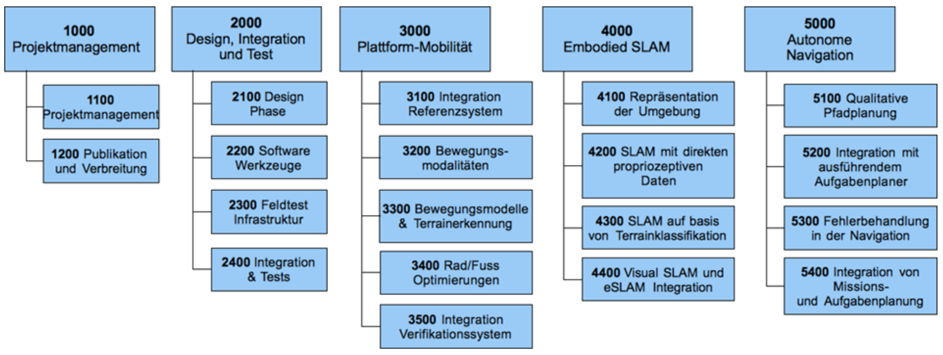
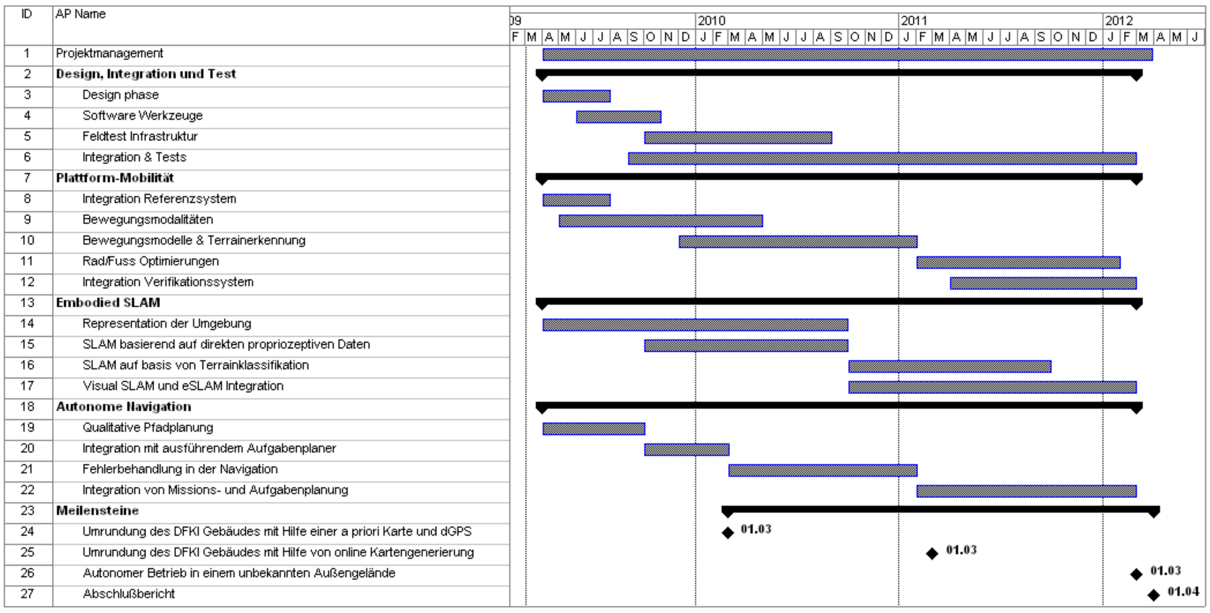
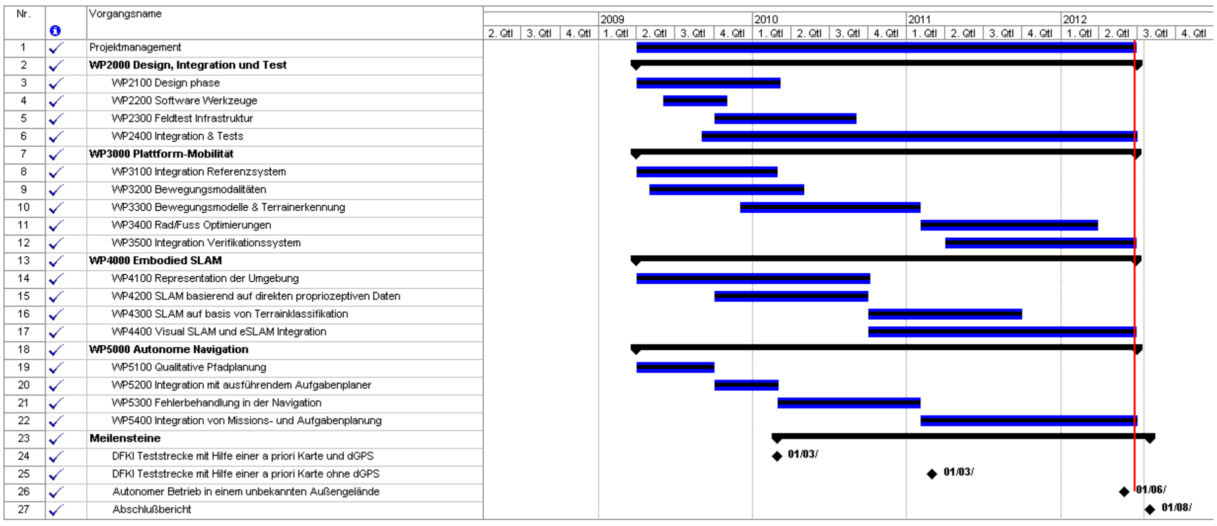

1000 Projektmanagement
1100 Projektmanagement
1200 Publikation und Verbreitung
2000 Design, Integration und Test
2100 Design Phase
2200 Software Werkzeuge
2300 Feldtest Infrastruktur
2400 Integration & Tests
3000 Plattform-Mobilität
3100 Integration Referenzsystem
3200 Bewegungs-modalitäten
3300 Bewegungsmodelle & Terrainerkennung
3400 Rad/Fuss Optimierungen
3500 Integration Verifikationssystem
4000 Embodied SLAM
4100 Repräsentation der Umgebung
4200 SLAM mit direkten propriozeptiven Daten
4300 SLAM auf basis von Terrainklassifikation
4400 Visual SLAM und eSLAM Integration
5000 Autonome Navigation
5100 Qualitative Pfadplanung
5200 Integration mit ausführendem Aufgabenplaner
5300 Fehlerbehandlung in der Navigation
5400 Integration von Missions- und Aufgabenplanung

Abbildung 1: *Strukturplan der Arbeitspakete*



ID	AP Name
1	Projektmanagement
2	**Design, Integration und Test**
3	Design phase
4	Software Werkzeuge
5	Feldtest Infrastruktur
6	Integration & Tests
7	**Plattform-Mobilität**
8	Integration Referenzsystem
9	Bewegungsmodalitäten
10	Bewegungsmodelle & Terrainerkennung
11	Rad/Fuss Optimierungen
12	Integration Verifikationssystem
13	**Embodied SLAM**
14	Representation der Umgebung
15	SLAM basierend auf direkten propriozeptiven Daten
16	SLAM auf basis von Terrainklassifikation
17	Visual SLAM und eSLAM Integration
18	**Autonome Navigation**
19	Qualitative Pfadplanung
20	Integration mit ausführendem Aufgabenplaner
21	Fehlerbehandlung in der Navigation
22	Integration von Missions- und Aufgabenplanung
23	**Meilensteine**
24	Umrundung des DFKI Gebäudes mit Hilfe einer a priori Karte und dGPS
25	Umrundung des DFKI Gebäudes mit Hilfe von online Kartengenerierung
26	Autonomer Betrieb in einem unbekannten Außengelände
27	Abschlußbericht

Abbildung 2: *Ursprünglicher Gantt-Ablaufplan*



Nr.	ℹ	Vorgangsname
1	✓	Projektmanagement
2	✓	**WP2000 Design, Integration und Test**
3	✓	WP2100 Design phase
4	✓	WP2200 Software Werkzeuge
5	✓	WP2300 Feldtest Infrastruktur
6	✓	WP2400 Integration & Tests
7	✓	**WP3000 Plattform-Mobilität**
8	✓	WP3100 Integration Referenzsystem
9	✓	WP3200 Bewegungsmodalitäten
10	✓	WP3300 Bewegungsmodelle & Terrainerkennung
11	✓	WP3400 Rad/Fuss Optimierungen
12	✓	WP3500 Integration Verifikationssystem
13	✓	**WP4000 Embodied SLAM**
14	✓	WP4100 Representation der Umgebung
15	✓	WP4200 SLAM basierend auf direkten propriozeptiven Daten
16	✓	WP4300 SLAM auf basis von Terrainklassifikation
17	✓	WP4400 Visual SLAM und eSLAM Integration
18	✓	**WP5000 Autonome Navigation**
19	✓	WP5100 Qualitative Pfadplanung
20	✓	WP5200 Integration mit ausführendem Aufgabenplaner
21	✓	WP5300 Fehlerbehandlung in der Navigation
22	✓	WP5400 Integration von Missions- und Aufgabenplanung
23	✓	**Meilensteine**
24	✓	DFKI Teststrecke mit Hilfe einer a priori Karte und dGPS
25	✓	DFKI Teststrecke mit Hilfe einer a priori Karte ohne dGPS
26	✓	Autonomer Betrieb in einem unbekannten Außengelände
27	✓	Abschlußbericht

Abbildung 3: *Gantt-Ablaufplan mit kostenneutraler Verlängerung*

2.4 Wissenschaftlicher und Technischer Stand zu Beginn des Vorhabens

2.4.1 Stand der Technik zu Beginn des Vorhabens

In den vergangenen zehn Jahren gab es im Bereich der Weltraumerforschung einige großartige Erfolge: Sojourner (Mars-Pfadfinder) und natürlich die Spirit und Opportunity (MER) Rover. Auf europäischer Seite ist ExoMars ein vielversprechendes Projekt zur Erkundung von Lebenszeichen auf dem Mars. All diese Rover haben eines gemeinsam: Ihr Design. Es handelt sich bei allen Systemen um radgetriebene Roboter mit komplexem Fahrgestell und Lenkungssystem, die hauptsächlich auf Grund der Mobilitätsanforderungen (Gefälle und Geländeart) in diesen Projekten verwendet werden. In einer vorangegangenen Machbarkeitsstudie über die Erforschung der lunaren Polregionen zeigte unsere Analyse, dass radgetriebene Systeme vergleichbaren Laufrobotern wie auch kettengetriebenen Rovern in ihrer Effizienz deutlich überlegen sind. Bei beiden Fortbewegungsarten beschränkt der Faktor Energieverbrauch die Einsetzbarkeit im Weltraum. Kettengetriebene Fahrzeuge sind ineffizient bezüglich des Verhältnisses von Energieverbrauch und Geschwindigkeit. Die Energie, welche Laufsysteme benötigen, um sich aufzurichten, steht ebenfalls in keinem Verhältnis zum Nutzen, es sei denn, es wird eine Form von dynamischem Vorwärtsgehen umgesetzt. Außerdem sind Laufsysteme sehr komplex und dadurch weniger robust. Auf der anderen Seite erreichen sowohl kettengetriebene als auch laufende Systeme auf unebenem Gelände eine weit bessere Mobilität. Laufsysteme schneiden außerdem in Umgebungen mit großen Felsbrocken bei weitem am besten ab. Auf der Grundlage dieser Überlegungen, welche auch für terrestrische Anwendungen gültig sind, wurde im DFKI das Asguard-Projekt aufgesetzt. Das Ziel dieses Projektes war es, ein Lauf-Rad-System zu entwerfen, das die Vorteile eines radgetriebenen Systems bei einfacher Bodenbeschaffenheit mit den Eigenschaften von Laufsystemen auf unebenem Gelände kombiniert. Das Ergebnis dieses Projektes waren zwei Versionen der Asguard-Plattform sowie die Grundlage für den Entwurf des CESAR-Roboters, der 2008 der Gewinner der ESA Lunar Robotics Challenge (LRC) war.

Damit robotische Systeme ihre Handlungen planen können, müssen ein Modell der Umgebung sowie eines der Fähigkeiten zur Fortbewegung des Roboters bekannt sein. Diese Informationen können jedoch im Allgemeinen bei Robotereinsätzen in einem unbekannten Gelände nicht vorweg gegeben werden, da Karten entweder nicht vorhanden oder für die geplante Aufgabe untauglich sind. Daher muss der Roboter in der Lage sein, seine sensorischen Informationen in eine logische Darstellung seiner Umgebung einzubinden und dieses Modell ständig zu aktualisieren. Die wichtigste Information für einen mobilen Roboter ist normalerweise räumlicher Art. Der Roboter muss über Hindernisse, überquerbare Wege und ggf. bestimmte Gegenstände, die in einem Zusammenhang mit dem Auftrag des Roboters stehen, Bescheid wissen. Außerdem muss er diese Informationen in einen räumlichen Zusammenhang setzen, damit sie auf seinen augenblicklichen Zustand bezogen werden können. Ein häufiges Problem bei der Erzeugung einer Repräsentation der Umgebung (Umgebungsmodell) ist, dass die Messungen von dem augenblicklichen Zustand (d.h. der Position) des Roboters abhängen und dass unterschiedliche Messungen nur dann in einem Rahmen zusammengefasst werden können, wenn die Zustände der Messungen bekannt sind. Bei einer unbekannten Umgebung und ohne die Fähigkeit, den absoluten Zustand messen zu können, stellt dies ein echtes Problem dar, da sowohl die Kartengenerierung als auch die Selbstlokalisation voneinander abhängen. Üblicherweise wird dieses Problem mit SLAM-Algorithmen (Simultaneous Localization and Mapping) gelöst, die gleichzeitig Kartenfehler und Lokalisationsfehler berücksichtigen. Die probabilistische Zustandsschätzung bei SLAM wird durch das Prinzip des Bayes-Filterns unterstützt. Die augenblickliche Zustandsschätzung (Karte + Position) wird extrapoliert, um ein erwartetes Sensordatum zu erlangen (d.h., was sollte der Roboter auf der Grundlage der aktuellen Karte sehen?). Diese Vorhersage wird mit den aktuellen Sensormessungen verglichen und die Differenz wird propagiert, wobei Fehlermodelle auf allen Stufen des Modells (Systemfehler, Sensorfehler, Kartenfehler, Lokalisationsfehler) berücksichtigt werden. SLAM-Algorithmen, die derzeit häufig in der Robotik eingesetzt werden, verbinden normalerweise zur Lokalisierung und zur Erstellung von Karten eine Odometrie mit exterozeptiven Sensordaten. In Innenräumen, wo 2D-Karten ausreichen, Laserscanner gut funktionieren und eine radgetriebene Fortbewegung auf ebenem Grund gute Odometriemessungen erlaubt, funktioniert dieser Ansatz gut. Außennavigation stellt wesentlich höhere Anforderungen an das Navigationssystem. Je nach Gelände und Fortbewegungsfähigkeit des Roboters können Odometrie nicht voraussagbar und 2D-Kartenrepräsentationen unzureichend sein. Zur Zeit wird 3D-SLAM mit qualitativ unterschiedlichen Ergebnissen untersucht. Die hohe Zahl von Konfigurationen, die innerhalb des 3D-Raumes möglich sind, führt zu sehr anspruchsvollen Algorithmen, die für die augenblicklich verfügbaren Datenverarbeitungssysteme eine große Herausforderung darstellen.

3D-SLAM für Innenräume ist weniger komplex, da die Ausmaße der Szene begrenzt und die vorhandenen Merkmale normalerweise von besserer Qualität sind (z.B. klar definierte Kanten und Ecken). Zudem sind die Lichtbedingungen kontrolliert und einzelne Merkmale haben einen Höchstabstand von einigen Metern. Eine interessante Arbeit von Davison (Davison, et al., 2004) zeigte, dass 3D-SLAM in Echtzeit mit Standard-Desktopcomputern bei Gebrauch einer einzelnen Kamera möglich ist. Durch eine weitere Verbesserung des Algorithmus, der lokale Koordinatensysteme als Knoten auf einem Graphen betrachtet, wird die Anzahl der Merkmale, die gleichzeitig behandelt werden können, erhöht (Eade, et al., 2007) und dennoch fast die gleiche Qualität und Konsistenz der optimalen Offline-Algorithmen erreicht (z.B. „bundle adjustment“). Andere Methoden wurden für 3D-SLAM im Außenbereich untersucht. Frühere Arbeiten von Montemerlo und Thrun (Montemerlo, et al., 2004) führten mit Hilfe von 3D-Laserscannern und GPS ein 3D-SLAM für ein großes Gebiet durch (ca. 600m Durchmesser). Mit einem weiteren interessanten Ansatz, der auf der bioinspirierten RatSLAM-Methode aufbaut, war es möglich, einige Quadratkilometer von Vorort-Straßennetzen zu kartieren (Milford, et al., 2008). Mit einer Stereokamera (Paz, et al., 2008) wurde ein großes Stadtgebiet erfolgreich kartiert. Eine Reihe von Ansätzen wurde für ein „scan matching“ von großen 3D-Punktwolken benutzt (Borrmann, et al., 2008; Newman, et al., 2006) und führt zu sehr großen Datenmengen. Die meisten dieser augenblicklich in der SLAM-Community benutzten Ansätze haben eines gemeinsam: Sie verwenden exterozeptive Sensordaten für die Messmodelle und propriozeptive Sensordaten für das Odometriemodell. Bis auf einige Ausnahmen (Chitta, et al., 2007; Stasse, et al., 2006) werden keine propriozeptiven Daten zum Abfragen der physikalischen Interaktion des Robotersystems mit der Außenwelt genutzt. Durch diese Trennung zwischen dem Roboterkörper und der Außenwelt wird die Hardware des Roboters zu einem simplen Sensorträger reduziert.

In der Literatur existieren viele unterschiedliche Ansätze zum Forschungsgebiet der autonomen Navigation, natürlich auch mit unterschiedlichen Anforderungen an die Leistungsfähigkeit der verfügbaren Hardware (Rechnerkapazitäten). Im nachfolgenden Absatz wird versucht, einige Begriffe für ein gemeinsames Verständnis des folgenden Textes zu definieren, die manchmal synonym verwendet werden können:

- Bewegungsplanung (motion planning) ist der Prozess, eine Reihe präziser Bewegungen zu erzeugen, die alle geometrischen Einschränkungen berücksichtigen, die durch die Umgebung und die Einschränkungen der Aktuation der betrachteten Roboterplattform entstehen. Im Allgemeinen wird angenommen, dass das System dem erzeugten Bewegungsplan direkt folgen kann (d.h., die Ausführung des Bewegungsplanes wird dem vorhergesagten Plan sehr genau folgen) (Lamiraux, et al., 1999; Lamiraux, et al., 2004; LaValle, et al., 2001; LaValle, 2006).
- Pfadplanung (path planning) ist der Prozess, ein geometrisches Objekt zu erzeugen, dem der Roboter so genau wie möglich folgen sollte. Anders als Bewegungsplanung wird im Allgemeinen zwar nicht erwartet, dass der Roboter dem erzeugten Pfad genau folgen kann, sondern nur, dass er sich entlang des Pfades bewegt. Hier sind die gebräuchlichsten Algorithmen A* und der verwandte Ansatz D* (Stentz, 1994; Philippsen, et al., 2003; Ferguson, et al., 2006).

Autonome Navigation bezeichnet den vollständigen Prozess, durch den sich der Roboter von einem Punkt zu einem anderen bewegt (um einen Ort zu erreichen). Die Ansätze reichen von der reinen verhaltensbasierten Navigation bis zu einer Kombination von Kartengenerierung, Lokalisation und Pfad-/Bewegungsplanung. Für die Auswahl der Ansätze müssen verschieden Kriterien gegeneinander abgewogen werden. In der Regel steigt für die Navigation in schwierigem Gelände auch die Komplexität des Ansatzes und somit die erforderlichen Rechnerkapazitäten. Eine Lösung für das Problem (die Auswahl des optimalen Ansatzes) ist eine Kombination von Bewegungsplanungs- und Pfadplanungsalgorithmus. Einerseits sind Bewegungsplanungsalgorithmen sehr speicherintensiv und benötigen viel Rechenzeit, können daher nur für Kurzstreckenbewegungen verwendet werden. Andererseits können Pfadplanungsalgorithmen nicht zur Lösung schwieriger Situationen verwendet werden, bei denen alle durch die Ausprägung des Roboterkörpers entstehenden Einschränkungen berücksichtigt werden sollten. Aus diesen Gründen basieren die meisten Ansätze zur autonomen Navigation auf der Verbindung eines höherwertigen Pfadplaners, dessen Ergebnis von einem Bewegungsplaner ausgeführt wird (Ingrand, et al., 2007). Es ist bekannt, dass diese Ansätze immer dort zu widersprüchlichen Entscheidungen (decision conflicts) führen können, wo der High-Level-Pfadplaner und der Low-Level-Bewegungsplaner einander widersprechende Entscheidungen treffen. Die Hauptursache hierfür ist, dass der High-Level-Pfadplaner eine „Vermutung“ erzeugt, die nur als ein Teil in die Kostenfunktion des Low-Level-Bewegungsplaners eingeht.

Ein weiterer Nachteil der derzeit verfügbaren Methoden ist das zugrundeliegende Prinzip, dass der Pfadplaner eine Optimierungsmethode für die Menge der Pfade ist, d.h., dass der erzeugte Pfad der bei einer gegebenen Kostenfunktion optimale Pfad ist. Dies ist ein Schwachpunkt, da besonders bei Outdoor-Umgebungen die Karte des Roboters nie perfekt ist und der Roboter nur über unzureichende Informationen bezüglich seiner Bewegung in einem bestimmten Gelände verfügt. Daraus folgt, dass die bei den bisherigen Navigationsmethoden genutzten Optimierungskriterien weniger geeignet sind. Erste Ansätze zur Lösung des oben beschriebenen Problems (Entscheidungsfindungskonflikte) versuchten, sowohl symbolische Planung als auch einen Algorithmus zur 3D-Bewegungsplanung fest miteinander zu verbinden. Obwohl das Ergebnis durch seine Fähigkeit zur Generierung komplexer Multi-Robot-Pläne beeindruckt, kann es zur Lösung von nur mäßig komplexen Problemen mehr als einen Tag benötigen (Gravot, et al., 2003). Neuere Ansätze schlugen vor, beide Detaillierungsgrade in einem Graphen zu integrieren und die niedrigste Ebene nur dann zu benutzen, wenn erforderlich (oder möglich) (Pivtoraiko, et al., 2008). Da die beiden Ebenen nun im selben Entscheidungsfindungsprozeß vertreten sind, zeigt diese Methode nicht das oben beschriebene widersprüchliche Verhalten. Das Problem hierbei ist jedoch, dass diese Bewegungsplanungsansätze durch einen Graphen repräsentierbar sein müssen. Dadurch wird die Auswahl an Methoden zur Bewegungsplanung, die besser an spezielle Roboter wie den Lauf-Rad-Roboter dieses Projektes angepasst werden können, eingeschränkt.

2.4.2 Fachliteratur und verwendete Dokumentationsdienste

- Bedrax-Weiss Tania [et al.] EUROPA2 : Plan Database Services for Planning and Scheduling Applications [Buch].
- Borrmann D [et al.] Globally consistent 3D mapping with scan matching [Journal] // Robotics and Autonomous Systems. - [s.l.] : Elsevier, 2008. - 2 : Bd. 56. - S. 130-142.
- Chitta S [et al.] Proprioceptive localilzatilon for a quadrupedal robot on known terrain [Konferenz] // International Conference on Robotics and Automation (ICRA). - 2007. - S. 4582-4587.
- Davison A J, Cid Y G und Kita N Real-time 3D SLAM with wide-angle vision [Konferenz] // IFAC Symposium on Intelligent Autonomous Vehicles. - 2004.
- Eade E und Drummond T Monocular SLAM as a Graph of Coalesced Observations [Konferenz] // International Conference on Computer Vision (ICCV). - 2007. - S. 1-8.
- Ferguson D und Stentz A Using Interpolation to Improve Path Planning: The Field D* Algorithm [Journal] // Journal of field robotics. - 2006. - 2 : Bd. 23. - S. 79.
- Gravot F, Cambon S und Alami R aSyMov: a planner that deals with intricate symbolic and geometric problems [Konferenz] // International Symposium on Robotics Research. - 2003.

- Ingrand F [et al.] Decisional autonomy of planetary rovers [Journal] // Journal of field robotics. - 2007. - 7 : Bd. 24. - S. 559.
- Joyeux Sylvain [et al.] A Plan Manager for Multi-Robot Systems [Journal] // The International Journal of Robotics Research. - 2008.
- Lamiraux F, Bonnafous D und Lefebvre O Reactive path deformation for nonholonomic mobile robots [Journal] // IEEE Transactions on Robotics. - 2004. - 6 : Bd. 20. - S. 967-977.
- Lamiraux F, Sekhavat S und Laumond J P Motion planning and control for Hilare pulling a trailer [Journal] // IEEE Transactions on Robotics and Automation. - 1999. - 4 : Bd. 15. - S. 640-652.
- LaValle S M Planning Algorithms [Buch]. - [s.l.] : Cambridge University Press, 2006.
- LaValle S M und Kuffner Jr J J Randomized Kinodynamic Planning [Journal] // The International Journal of Robotics Research. - 2001. - 5 : Bd. 20. - S. 378.
- Milford M J und Wyeth G F Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System [Journal] // IEEE Transactions on Robotics. - 2008. - 5 : Bd. 24. - S. 1038-1053.
- Montemerlo M und Thrun S Large-scale robotic 3-d mapping of urban structures [Konferenz] // Proceedings of the International Symposium on Experimental Robotics (ISER). - 2004.
- Newman P, Cole D und Ho K Outdoor SLAM using visual appearance and laser ranging [Konferenz] // IEEE International Conference on Robotics and Automation (ICRA). - 2006. - S. 1180-1187.
- Nuchter A [et al.] Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM [Journal] // Lecture notes in computer science. - 2005. - Bd. 3698. - S. 304.
- Paz L M [et al.] Large-Scale 6-DOF SLAM With Stereo-in-Hand [Journal] // IEEE Transactions on Robotics. - 2008. - 5 : Bd. 24. - S. 946-957.
- Philippsen R und Siegwart R Smooth and efficient obstacle avoidance for a tour guide robot [Konferenz] // International Conference on Robotics and Automation (ICRA). - 2003.
- Pivtoraiko M und Kelly A Differentially constrained motion replanning using state lattices with graduated fidelity [Konferenz] // International Conference on Intelligent Robots and Systems (IROS). - 2008. - S. 2611-2616.
- Stasse O [et al.] Real-time 3d slam for humanoid robot considering pattern generator information [Konferenz] // International Conference on Intelligent Robots and Systems (IROS). - 2006.
- Stentz A Optimal and efficient path planning for partially-knownenvironments [Konferenz] // International Conference on Robotics and Automation (ICRA). - 1994. - S. 3310-3317.

- Spenneberg D [et al.] The Bio-Inspired SCORPION Robot: Design, Control & Lessons Learned [Buchkapitel] // Climbing & Walking Robots, Towards New Applications . - 2007. - S. 197-218.
- Spenneberg D und Kirchner F A Free-Climbing Robot for Steep Terrain [Konferenz] // 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space. - 2008.
- Bartsch S und Planthaber S Scarabaeus: A Walking Robot Applicable to Sample Return Missions [Konferenz] // Eurobot-08. - 2008. - S. 178-183.
- The Orocos Real-time Toolkit [Buch].
- Wulf O [et al.] Ground truth evaluation of large urban 6D SLAM [Konferenz] // International Conference on Intelligent Robots and Systems (IROS). - 2007. - S. 650-657.

Weitere verwendete Fachliteratur ist in den jeweiligen AP-Dokumenten sowie in den im Appendix (Sektion 5) eingefügten Dokumenten angegeben.

2.5 Zusammenarbeit mit anderen Stellen

Das Intelligent-Mobility-Projekt ist in seiner Gesamtheit vollständig durch das DFKI RIC durchgeführt worden. Zusammenarbeit mit anderen Stellen hat es auf der Basis von Informationsaustausch mit anderen Stellen innerhalb und außerhalb des RIC gegeben. Insbesondere wurden Lösungen zu inhaltlichen Problemen mit den Projekten RIMRES, VIRGO, CUSLAM und AVALON abgestimmt. Diese Herangehensweise hat es ermöglicht, weitgehende Kompatibilität zwischen den in verschiedenen Projekten entwickelten Softwaremodulen zu erreichen.

3 Eingehende Darstellung

3.1 AP1000: Projektmanagement

Dieses Arbeitspaket behandelt die Managementaufgaben des gesamten Projektes, die Vorbereitung der Projektreviews und Projektberichte, den permanenten Kontakt zum Projektträger sowie entsprechende Aktivitäten zur Dissemination der Projektergebnisse.

WP1100: Projektmanagement

Hauptaufgabe des Projektmanagements im Laufe dieses Projektes war die Koordination des Projektstarts, Beschaffung notwendiger Komponenten und die Anwerbung aber auch die Einarbeitung der Mitarbeiter.

Es wurden wöchentliche Projektmeetings abgehalten, die in der Regel etwa 60 Minuten in Anspruch genommen haben. Dabei wurden die erreichten Ergebnisse sowie das weitere Vorgehen bei den einzelnen Arbeiten besprochen.

Zusätzlich wurde ein Ansatz zur "fortschreitenden Entwicklung" des Robotersystems eingeführt, der direkt zur Evolution des gesamten Systems beiträgt. Dafür wurden dreiwöchige Iterationen definiert, in denen Meilensteine definiert, realisiert und schließlich getestet werden.

Die Struktur einer dreiwöchigen Iteration unterteilt sich, wie in Abbildung 4 dargestellt, in folgende Abschnitte:

- Im iMoby-Meeting zu Beginn der ersten Woche werden die Ziele für die nächste Iteration definiert (*Design Meeting*). Relevante Aufgaben werden den Mitarbeitern und Studenten zugewiesen, um die Ziele erreichen zu können. Diese Aufgaben müssen bis zum Ende der zweiten Woche erledigt werden. Ein **Trac-System** wird dabei eingesetzt, um die einzelnen Aufgaben verwalten und deren aktuellen Zustand verfolgen zu können.

- Zu Beginn der zweiten Woche werden die Aufgaben und Lösungen besprochen. Wenn nötig, wird die Aufgabe entsprechend angepasst, um auf unvorhergesehene Probleme reagieren zu können.

- Am Ende der zweiten Woche sollten die Aufgaben erledigt und die Ziele erreicht sein. Dies wird durch ein Outdoor-Experiment mit dem kompletten Robotersystem überprüft. Dieses Experiment soll die Auswirkung von Änderungen oder Neuerungen im System zeigen und es gleichzeitig erlauben, sie zu testen. Wenn nötig, können auch Ergebnisse offline verarbeiteter Daten vorgestellt werden. Experimente dieser Art tragen dazu bei, Projektmitarbeitern Daten für ihre weitere Forschungsarbeit zu liefern.

- Die dritte Woche wird dazu genutzt, die aufgenommenen Daten zu analysieren und über weitere Ziele für die nächste Iteration nachzudenken.

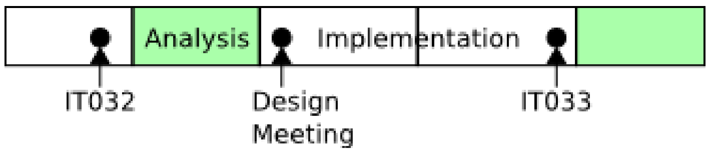


Abbildung 4: *Struktur einer Iteration*

Jeder Mitarbeiter sollte innerhalb einer Iteration dennoch Zeit für Grundlagenforschung haben, die an sich nicht zur Iteration zählt. Deshalb verfolgen wir das Ziel, 60% der Zeit für Grundlagenforschung und 40% für Arbeit an der Iteration sowie an administrativen Aufgaben zu nutzen.

Das Ziel dieser Iterationen ist es, das Referenzsystem ständig mit neuen Ideen und Ansätzen zu erweitern. Dadurch sollen neue Erkenntnisse schnell ins System integriert werden können.

Weitere Arbeiten, die in WP1100 durchgeführt wurden, waren die kontinuierliche Kontrolle des Projektfortschritts und die Anpassung der Planung, die Organisation und Verwaltung von Dienstreisen und Urlaubsanträgen, sowie die Überprüfung getätigter Bestellungen.

Auch das Berichtswesen ist in diesem Arbeitspaket durchgeführt worden. Alle drei Monate wurden Quartalsberichte über den Projektfortschritt verfasst. Alle sechs Monate wurden Zwischenberichte für den Mittelgeber erstellt.

WP1200: Publikation und Verbreitung

Innerhalb der Projektlaufzeit wurden 16 Publikationen verfasst, von denen 10 in internationalen Journals und bei Konferenzen veröffentlicht worden sind. Mindestens drei Publikationen, die zumindest teilweise Projektergebnisse aus iMoby enthalten, sind noch in Planung.

Weiterhin sind die Ergebnisse und Zwischenergebnisse bei einer Reihe von internen und externen Veranstaltungen, Besuchen und Messen vorgestellt worden. Zusätzlich gab es eine eingehende Berichterstattung über Teile des Projektes in der Presse.

Eingereichte Publikationen

- J. Schwendner, P. Paranhos and C. Gaudig: "*Simultaneous Localisation and Mapping using Terrain Classificiation*", bei International Conference on Robotics Systems (IROS2012)
- Joyeux, S. and Kirchner, F. "*Leaving Choices Open in Planner/Planner Integration*", bei Internation Conference on Robotics and Systems (IROS2009)
- "*High-level specification and online adaptation of a component layer*", bei International Conference on Intelligent Robots and Systems (IROS 2010, full paper submitted)

- "*High-level specification and online adaptation of a component layer*", bei International Conference on Robotics and Automation (ICRA 2011)
- "*Robot Localisation using Direct Embodied Data for a Leg/Wheel Hybrid in Partially Known Environments*", bei International Conference on Robotics and Automation (ICRA 2011)
- Jakob Schwendner, Sylvain Joyeux: "*Robot Localisation for Direct Embodied Data for a Hybrid Leg Wheel System*", bei International Conference on Robotics Systems (IROS2011)

Eingereichte und angenommene Publikationen

- Schwendner, J. and Kirchner, F. "*eSLAM: Self Localisation and Mapping using Embodied Data*", KI Journal 2009.
- Schwendner, J. and Joyeux, S. "*Classifying Autonomy for Mobile Space Exploration Robots*", bei International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010)
- Joyeux, S., Kirchner, F. and Lacroix S. "*Managing Plans: Integrating Deliberation and Reactive Execution Schemes*", in Journal of Robotics and Autonomous Systems, Volume 58, Number 9, 2010.
- Babu, A., Joyeux, S., Schwendner, J., Grimminger, F. "*Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard*", bei International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010)
- Joyeux, S., Schwendner, J., Kirchner, F., Babu, A., Grimminger, F., Machowinski, J., Paranhos, P. and Gaudig, C."*Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI*", in KI - Künstliche Intelligenz, Special Issue on Outdoor Robotics, 2011
- Sylvain Joyeux, Jan Albiez: "*Robot development: from components to systems*", bei 6th National Conference on Control Architectures of Robots (2011)
- Schwendner, J., Joyeux, S. "*Self Localisation using Embodied Data for a Hybrid Leg-Wheel Robot*ïn IEEE International Conference on Robotics and Biomimetics (ROBIO 2011)
- J. Schwendner: "*Terrain Aided Navigation for Planetary Exploration Missions*", bei International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2012)
- S. Joyeux and T. Roehr: "*Building Robust Component-based Systems with Rock*", bei International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2012)
- J. Schwendner: "*Map Segmentation based SLAM using Embodied Data*", IEEE International Conference on Multisensor Fusion and Information Integration, 2012

Pressespiegel

Es folgt eine Übersicht über die Verwertung von Inhalten des iMoby-Projekts durch die Presse. Die Liste zeigt das Datum, die Art und das Medium der Berichterstattung. Registriert sind nur Inhalte, die direkt durch das DFKI betreut worden sind. Die Liste zeigt nicht die Weiterverwertung der Inhalte.

Datum	Gegenstand	Institution/Medium
05/05/12	Kunststoff Kumpel mit Superhirn - Über Roboter und künstliche Intelligenz	Deutschlandradio
12/2011	Selbstständige Roboter	DAAD-Magazin Letter
11/2011	Sendebeitrag über Asguard auf dem arabischen Sender Al Jazeera	Al Jazeera
11/10/11	Wie lernen Roboter?	ARTE X:enius
09/09/11	Beitrag in der Gadget Show, England	The Gadget Show
01/09/11	Künstliche Intelligenz: Der Geist aus Silizium	GEO Kompakt
04/05/11	Roboter: Die überforderten Helfer	Saarbrücker Zeitung
19/03/11	Roboter für den Höllenjob	Spiegel online
19/01/11	Hochtechnologie aus Bremen	Weser Kurier
14/10/10	Satellite Data for Unmanned Mobile Systems	Spotlight on a Dynamic Region (CEON Imagebroschüre)
13/10/10	Bremens Kinder greifen nach den Sternen, Roboter für Weltraum und Tiefsee	KiTa Bremen
26/04/10	Leistungsschau der Landroboter	www.bundeswehr.de

Messen/Events

Die Inhalte des iMoby-Projektss und/oder die Systeme sind auf zahlreichen Messen und Veranstaltungen gezeigt worden. Die folgende Tabelle zeigt die Termine und die Art der Veranstaltung.

Datum	Gegenstand	Institution/Medium
6.7./03/12	Nationale Konferenz zur Raumfahrtrobotik 2012	DLR / BMWi Berlin
29/09/11	Eröffnung Außenstelle Osnabrück	DFKI/Uni Osnabrück
20./21.08.11	Systempräsentation Tag der offenen Tür der Bundesregierung	BMWi
14/04/11	Girls' Day/Zukunftstag	RIC
29/06/11	Tag der offenen Tür für Schüler	Schulklassen
21.-30.01.11	Messe: Internationale Grüne Woche, Asguard auf dem KTBL-Gemeinschaftsstand	Internationale Grüne Woche/Berlin
20/01/11	Ausstellungseröffnung zur Reihe EINFACH WISSENSWERT, Thema "Intelligente Systeme", Demo Asguard	Haus der Wissenschaft Bremen
2./3.10.10	Präsentation und Demos im Rahmen der Feierlichkeiten in Bremen Überseestadt	Tag der Deutschen Einheit
5/3/2010	Girls' Day/Zukunftstag	Schüler

Besucher

Im Zeitraum vom 28.4.2009 bis zum 7.12.2012 haben 30 Termine stattgefunden, in denen die Inhalte des iMoby-Projektes interessierten Besuchern aus Industrie, Wissenschaft, Politik und Raumfahrtagenturen (DLR/ESA) präsentiert worden sind. Die Ergebnisse des Projektes sind auf positive Resonanz gestoßen. Bestehende Kooperationen mit Industriepartnern konnten so gefestigt und neue Kontakte geknüpft werden. Auf Wunsch vieler Firmenbesucher werden keine Einzelheiten zu den Besuchsterminen veröffentlicht.

3.2 AP2000: Design, Integration und Test

Dieses Arbeitspaket beinhaltet zwei Aufgabenstellungen:

- In der Anfangsphase des Projektes werden die Anforderungen an das System definiert, die durch die vorgeschlagenen Methoden erfüllt werden sollen. Dazu zählen maximale Neigungen, die höchste zu erreichende Geschwindigkeit, und der Energieverbrauch. Gleichzeitig werden Testszenarien und eine Testmethodik entwickelt, durch die Vorteile der vorgeschlagenen Ansätze demonstriert werden können.

- In der ersten Hälfte des Projektes werden die notwendigen Instrumente (Hardware und Software) zur Unterstützung der Feldversuche entwickelt. Die regelmäßigen Feldversuche werden im Allgemeinen in der Umgebung des DFKI Bremen, aber auch einmal jährlich während eines dreitägigen Ausflugs in eine anspruchsvollere Umgebung (z.B. Gebirge) durchgeführt.

WP2100: Design-Phase

Das Ziel dieses Arbeitspaketes war es, das Ergebnis des Projekts im Bereich *Autonomy and Mobility* einzuordnen.

Das **Dokument zu Terrain-Typologien** wurde fertiggestellt (Sektion 5.2). Auf Basis der darin gewonnenen Erkenntnisse ist es nun möglich, Boden- und Oberflächenstrukturen systematisch zu klassifizieren und dem Robotersystem als zusätzliche Umweltinformation bereitzustellen. Die Parameter, die maßgeblich zur Charakterisierung beitragen, sind die *Neigung*, die *Höhe eines Hindernisses*, seine *Auftrittshäufigkeit* und *Flexibilität* wie auch die *Scherfestigkeit* sowie die *Kohäsion* des Substrats. Manche Parameter müssen experimentell bestimmt oder geschätzt werden, da in der Literatur nicht alle Informationen zu finden sind.

Die Arbeiten zu den **Leistungskriterien** zur Evaluierung des Entwicklungsfortschritts eines Systems wurden abgeschlossen. Es wurden unter anderem Kriterien aus den Bereichen Lokomotion (Geschwindigkeit, Manövrierfähigkeit, ...), Navigation (Effizienz, Geschwindigkeit, ...) und Fehlertoleranz (Messfehler, Systemfehler, ...) in die Betrachtung einbezogen. Die **Definition der Szenarien** und Betrachtungen zum **Stand der Technik** – siehe Abbildung 5 – hängen direkt mit den Erkenntnissen zur Leistungsbewertung zusammen.

Der D2.1 Autonomy Bericht, der eine Grundlage zur Klassifizierung und zum Vergleich von autonomen System darstellt, ist als eine Publikation im Bereich des Benchmarking bei der i-SAIRAS 2010 veröffentlicht worden. Der Bericht ist in Sektion 5.1 zu finden.

Weiterhin ist ein Bericht zur Charakterisierung von Terrain erstellt worden, um eine einheitliche Beschreibung von Oberflächenbeschaffenheiten zu ermöglichen. Der Bericht ist in Sektion 5.2 eingefügt.

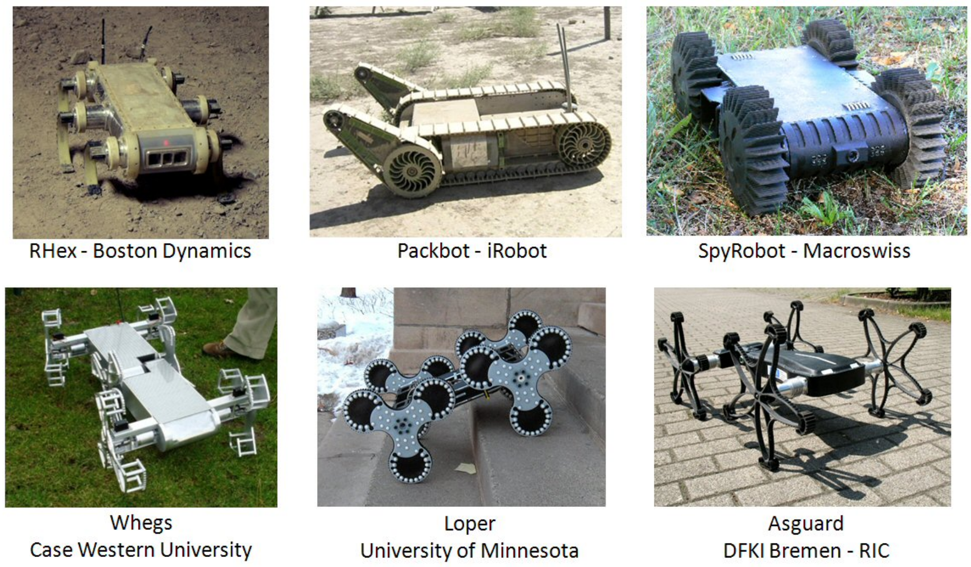


Abbildung 5: *State of the Art bei Whegs-basierten Robotern*

WP2200: Software-Werkzeuge

Innerhalb des Projektes wurden zwei Software-Werkzeuge entwickelt:

Orogen Orogen ist ein einfach zu benutzendes Generierungs-Werkzeug, das den Lernprozess für das modulare Integrationsframework Orocos drastisch vereinfacht. Es ist vor allem nützlich, um Studenten und neuen Projektmitarbeitern die Mitarbeit zu vereinfachen. Abbildung 6 stellt den typischen Ablauf bei Verwendung von orogen dar. Dokumentation zu orogen sowie das Software-Paket selbst sind frei verfügbar `http://github.com/doudou/orogen/`.

Autoproj Es wurde ein **Build-System** entwickelt, das an die Struktur und Funktionalität der genutzten Software-Architektur angepasst ist. Dokumentation zu autobuild sowie das Software-Paket selbst sind frei verfügbar `http://doudou.github.com/autoproj/`.

Im Laufe des Projektes wurden mindestens fünf interne Software-Workshops abgehalten, um die Projektmitarbeiter in die Entwicklungsumgebung einzuführen. Dabei wurden das Software-Build-System, das Versionsverwaltungssystem **git** sowie das modulare Integrationsframework näher beschrieben. Dadurch sind nun alle Projektmitarbeiter in der Lage, zur Integration von neuen Komponenten beizutragen. Die Workshops wurden auch für Mitarbeiter anderer Projekte geöffnet, so dass eine Verbreitung der Methoden auch über Projektgrenzen hinaus erreicht werden konnte.

Ein **Build-Server** (Continuous Integration Server) wurde aufgesetzt, der täglich die aktuelle Entwicklerversion des Softwaresystems übersetzt. Dadurch kann festgestellt werden, ob die Systemsoftware durch aktuelle Änderungen beschädigt wurde.

Über die Laufzeit des Projektes wurde die entwickelte Software-Infrastruktur in weiten Teilen optimiert und an die Bedürfnisse des Entwicklungsprozesses angepasst.

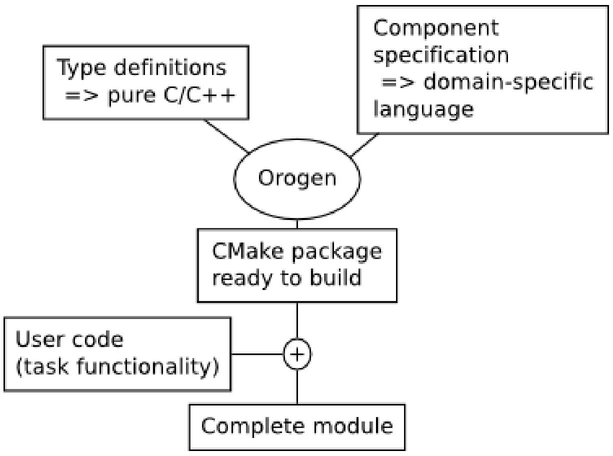

Type definitions
=> pure C/C++
Component specification
=> domain-specific language
Orogen
CMake package ready to build
User code (task functionality)
+
Complete module

Abbildung 6: *Arbeitsablauf bei der Orocos-Modulgenerierung. Der Entwickler erstellt eine abstrakte Beschreibung (eine .orogen-Datei) und übersetzt diese mit Hilfe des orogen-Werkzeugs in einen Orocos-Modulstub. Die eigentliche Modulimplementierung kann dann in C++ geschrieben werden, ohne dass der Entwickler sich um Details des Integrationsframeworks kümmern muss.*

Die Bedienung des ***orogen*-Werkzeugs** zur Generierung von Orocos-Modulstubs berücksichtigt Erweiterungen zur Orocos-Middleware, mit denen sich E/A-Aufgaben effizienter in Orocos-Modulen abbilden lassen. Weiterhin wurde die *domänenspezifische Modellierungssprache* (Domain-Specific Language; DSL) von orogen weiter optimiert und vereinfacht, um Orocos-Module noch einfacher und konsistenter entwickeln zu können.

Ein Präsentation einer Untermenge der in iMoby entwickelten Softwaretools ist in der Präsentation in Sektion 5.3 zu finden.

WP2300: Feldtest-Infrastruktur

Die **Infrastruktur für Feldtests** wurde beschafft und entsprechend in Betrieb genommen. Wie in Abbildung 7 dargestellt, ist sie jetzt zentraler Bestandteil der dreiwöchigen Iterationen. Es stehen Tische, Stühle, Laptops und eine Werkzeugausrüstung für mechanische und elektrotechnische Arbeiten zur Verfügung. Zusammen mit der in den ersten Monaten erarbeiteten Vorgehensweise zur **Planung von Feldtests** wird sichergestellt, dass alle Arbeiten auch außerhalb des Labors durchgeführt werden können.

Für das Testgelände am DFKI-RIC-Labor Bremen wurde die **dGPS Base Station** in ein wetterfestes Gehäuse integriert – siehe Abbildung 7 – und auf dem Gebäude installiert. Die mit diesem System gewonnenen Positionsdaten werden als Ground-Truth-Referenz zur Evaluierung des Roboterverhaltens verwendet. Nach einer Recherche zu existierenden Lösungen ist eine freie Planungssoftware für GPS (Abb. 8) gefunden worden, die das Durchführen von Experimenten erheblich vereinfacht, da sie es ermöglicht, eine zeitliche Vorausabschätzung der Güte der GPS-Satellitenkonfiguration und somit eine angepasste Experimentplanung durchzuführen.

Eine **Operator Control Unit** (OCU), wie sie in Abbildung 9 dargestellt ist, wurde in-

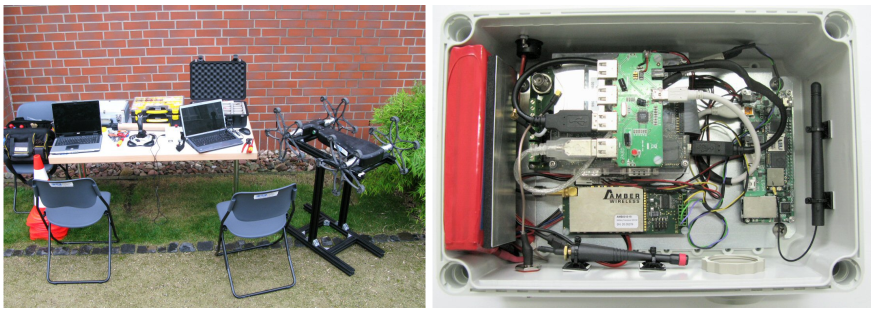
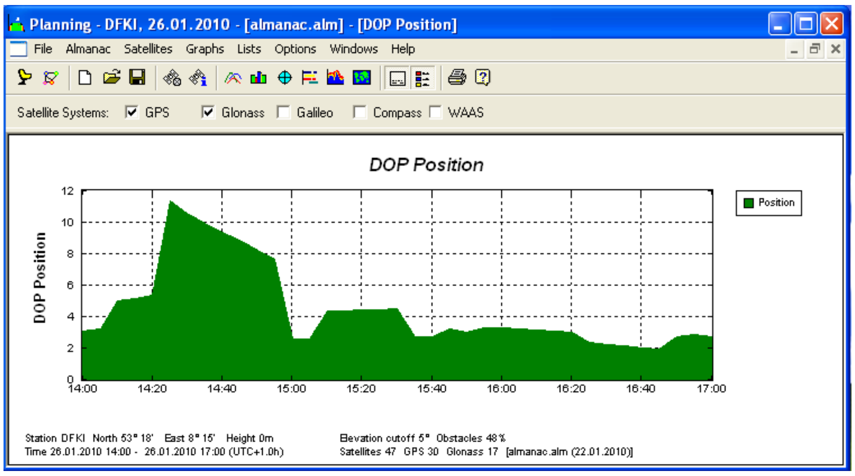


Abbildung 7: *Die iMoby-Feldtest-Infrastruktur: iMoby-Iteration (Feldtest) und dGPS-Außenstation*



Satellite Systems: [✓ GPS] [✓ Glonass] Galileo Compass WAAS

Station DFKI North 53° 18' East 8° 15' Height 0m
Time 26.01.2010 14:00 - 26.01.2010 17:00 (UTC+1.0h)

Elevation cutoff 5° Obstacles 48 %
Satellites 47 GPS 30 Glonass 17 [almanac.alm (22.01.2010)]

Abbildung 8: *Screenshot der GPS-Planungssoftware*

tegriert und steht nun zur Steuerung des Robotersystems zur Verfügung. Entwickelt wurde hier eine generische Kontrolleinheit auf Basis eines Linux-Systems, mit dem auch komplexe Steuerungs- und Überwachungsaufgaben durchgeführt werden können. Die wichtigsten Zustandsdaten des Roboters können direkt an der OCU über eine Anzeige abgelesen werden. Desweiteren unterstützt die OCU das Erstellen von experimentellen Logdaten, indem sie die Möglichkeit bietet, Start- und Stop-Punkte von Experimenten in den Logdaten zu setzen.

Dieses Arbeitspaket wurde nach Absprache mit dem DLR vor dem geplanten Datum abgeschlossen.

Die Feldtest-Infrastruktur wurde früher als geplant benötigt. Daher wurde der entsprechende Arbeitsaufwand in den ersten Monaten der Projektlaufzeit geleistet.

Es wurden außerdem Tests der Kommunikationsinfrastruktur durchgeführt. Die Testergebnisse sind in Sektion 5.4 eingefügt.

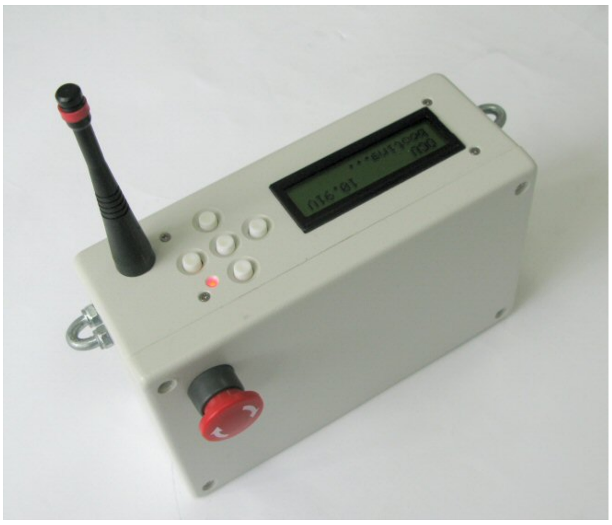


***Abbildung 9:** Die Operator Control Unit*

WP2400: Integration und Tests

Die Ergebnisse aus den AP3000-5000 sind kontinuierlich auf dem Asguard-System integriert und getestet worden. Im Laufe des Projektes war dabei eine Verschiebung von Aktivitäten an der Hardware zu Beginn des Projektes hin zu einer stabilen Plattform, die jeweils nur Softwareanpassungen benötigt, zu erkennen.

Die Simulation des Roboters ASGUARD wurde in die Supervisionsschicht integriert. Dadurch ist es möglich, eine Vielzahl von Softwarepaketen ohne größere Modifikation sowohl in der Simulation als auch auf der eigentlichen Hardware zu testen.

Für das Stereokamerasystem wurde ein Treiber sowie ein Framework-Modul geschrieben. In Kombination mit dem entwickelten Hardware-Trigger-Modul können hochsynchrone Stereobilder mit hoher Framerate aufgenommen werden. Das zur Ansteuerung des Stereokamerasystems verwendete Softwaremodul besteht aus zwei Modulen, sodass jedes Modul eine monokulare Kamera ansteuert. Dadurch ist ein flexiblerer Einsatz der Kameras möglich, z.B. für Algorithmen, die keine Stereodaten erfordern (wie z.B. die visuelle Terrainklassifikation, s. WP4300). Die für Stereodaten unbedingt erforderliche Synchronisierung der beiden Kameras, die über eine per CAN-Bus angebundene Triggerplatine erfolgt, wird hierdurch nicht negativ beeinflusst.

Das Software-Framework, das im Projekt eingesetzt wird, wurde weiterentwickelt. Die neue Version des Frameworks bietet die Möglichkeit, die Komponenten noch modularer zu entwickeln, wodurch es möglich wird, diese in anderen DLR-geförderten Projekten (RIMRES) ohne Anpassungen wiederzuverwenden. Dazu bietet die neuen Version des Frameworks eine überarbeitete API, die es ermöglicht, übliche Probleme bei der Komponentenentwicklung einfacher zu lösen. Dies resultiert in einer schnelleren und einfacheren Komponentenentwicklung.

Es hat sich während der Entwicklung einiger Softwaremodule gezeigt, dass viele Komponenten Daten von einem Koordinatensystem in ein anderes umrechnen müssen. So müssen z.B. die Daten des Laserscanners in das Koordinatensystem des Roboters oder in das Welt-Koordinatensystem umgerechnet werden. Dabei wurde es umso schwieriger und fehleranfälliger, diese Umrechnung durchzuführen, je mehr Zwischenschritte benötigt wurden. Um dieses Problem zu lösen, wurde ein Transformations-Stack eingeführt, der automatisch (sofern möglich) Transformationen aus einem Koordinatensystem in ein anderes Koordinatensystem berechnet. Zusätzlich bietet der Transformations-Stack die Möglichkeit, zwischen zwei aufeinanderfolgenden Transformationen zu interpolieren. Dadurch ist es möglich, Transformationen zu erzeugen, die "näher" an dem Zeitpunkt liegen, an dem die umzurechnenden Daten aufgenommen wurden. Dies führt somit zu einer höheren Genauigkeit bei der Umrechnung der Daten.

Nach der Entwicklung des Transformations-Stacks zeigte sich, dass für dessen Benutzung in jeder Komponente, die ihn einsetzt, immer wieder derselbe Programmcode benötigt wurde. Da sich dieses Problem nicht durch die Sprachmittel von C++ lösen ließ, wurde unser Programmcode-Generierungswerkzeug um eine Plugin-Schnittstelle erweitert und ein Plugin für den Transformations-Stack geschrieben. Dadurch ist es nun möglich, den Transformations-Stack mittels vier statt ca. 40 Zeilen Programmcode einzubinden, was die Komponentenentwicklung merklich beschleunigt sowie vermeidbaren Programmierfehlern entgegenwirkt.

Während der Tests mit Asguard V3 trat ein Problem auf, bei dem die Motortreiber falsche Radpositionen für den äußeren Encoder meldeten. Die Analyse des Problems ergab, dass die in Asguard V3 eingesetzten Motortreiber starke elektromagnetische Störungen erzeugen. Durch diese Störungen wurde die Index-Leitung so stark beeinträchtigt, dass sie falsche Nullstellungen meldete. Um das Problem zu beheben, wurde der Motortreiber um einen sogenannten *snubber*-Schaltkreis erweitert, der die elektromagnetischen Störungen stark mindert. Da das Problem durch den *snubber*-Schaltkreis zwar seltener auftrat, aber nicht komplett behoben wurde, wurde zusätzlich die Firmware der Motortreiber so modifiziert, dass sie strengere Validitätsprüfungen bei einem Nullstellenpuls (Länge, Stellungen der Encoderleitungen relativ zum Nullstellenpuls) durchführt. Hierdurch wurde das Problem schlussendlich gelöst.

Um das generelle EM-Problem noch weiter zu reduzieren, wurde die Arbeit an einer neuen Platine für die Motortreiber begonnen.

Es wurden weitere Probleme bei der Hardware festgestellt: Elektromagnetische Störungen sorgten für Rauschen im Signal der Motor-Encoder (ähnlich den Problemen, von denen schon berichtet wurden). Die Störung wurde behoben.

Um eine tiefergehendere Analyse der experimentellen Daten zu ermöglichen, wurde ein Software-Tool entwickelt, das es erlaubt, extern aufgenommene Videos zeitsynchron mit intern aufgezeichneten Betriebsdaten darzustellen. Somit ist es nun möglich, verschiedene Bewegungszustände des Roboters zeitgenau auf die internen Datenströme abzubilden.

Es wurde ein Fehler beim Daten-Logging behoben. Das Problem bestand darin, dass es beim Einschalten der Kameras dazu kommen konnte, dass das System in einen Zustand von Input/Output-Starvation geriet. Dies wiederum führte zu Lücken in den aufgezeichneten Daten von bis zu 3 Sekunden.

Ein Referenz-Datenset von einer anderen Umgebung als der Roboter-Teststrecke des DFKI wurde erstellt, welches GPS als Groundtruth nutzt. Anhand dieser Daten wurden die Ergebnisse aus WP4200 erneut evaluiert.

Das Corridor Servoing (CS), d.h. das Verhalten zur Durchquerung visuell begrenzter Korridore, ist durch die Einbindung von eSLAM-Komponenten erheblich verbessert worden. Dadurch ist das CS in der Lage, den Holzstapel auf der Roboter-Teststrecke zu überqueren. Außerdem ist auch die Überquerung der Brücke mit dem neuen Ansatz nach wie vor möglich und sogar verbessert worden.

Beim Überqueren des Holzstapels sind einige Schwierigkeiten aufgetreten, da das System durch Punktdrehungen seine Richtung ändern muss. Auf Grund der hohen auftretenden Kräfte sind die Motoren durch die Strombegrenzung abgeschaltet worden. Es hat einige Versuche mit 'soft turn'-Techniken gegeben, die eine Vor- und Zurückbewegung beim Drehen anwenden. Das Problem ist durch Reduktion der maximalen Motorspannung gelöst worden. Es gibt bei der Rotation weiterhin hohe Drehmomente an den Rädern. Diese Probleme sollen durch die Drehmoment-Regelung in WP3300 behoben werden.

Um die internen Zustände des Roboters besser nachvollziehen zu können, sind einige Arbeiten zur Visualisierung durchgeführt worden. Ein Großteil der online verwendeten Algorithmen verfügt jetzt über Visualisierungskomponenten.

Bei der Verifikation der Motorströme des AsguardV3-Systems zur Abschätzung der Motorgröße für das Verifikationssystem sind mit den Datenblättern inkonsistente Messwerte aufgetreten. Die bisher durchgeführte Methode zur Kalibrierung der Stromwerte hat sich daraufhin als unzureichend erwiesen. Aus diesem Grund ist eine neue Methode der Kalibrierung auf Basis der Drehmomentwerte durchgeführt worden. Hier wird nicht mehr der Stromfluss direkt gemessen, da die Messung von gepulsten Strömen sich mit den vorhandenen Messgeräten nicht genau darstellen lässt. Als Alternative wird der Stromfluss über das Drehmoment abgeschätzt. Als Ergebnis der Kalibrierung durch die neue Methode hat sich ergeben, dass die bisherige Strommessung den Strom gerade im unteren PWM-Bereich zu niedrig gemessen hat. Abb. 10 zeigt die Ergebnisse vor und nach der Kalibrierung.

Im November und Dezember 2011 aufgezeichnete Logdaten haben bei einer Analyse gezeigt, dass Probleme bei der zeitlichen Einordnung der Sensorinformationen auftreten. Eine genaue zeitliche Korrelation der Sensorinformationen ist für Lokalisierung und Kartenbau notwendig. Aus diesem Grund wurde hier Zeit investiert, um die Synchronisation der Sensordaten zu verbessern. Diese Verbesserungen kommen auch anderen Projekten wie RIMRES und Virgo zugute. Des Weiteren wurde ein Modul zur Bereinigung von Sensordaten aus dem Laserscanner geschrieben. Dieses Modul filtert Sensorperzepte, die durch den Körper des Systems selbst hervorgerufen und fälschlicherweise der Umgebung zugeordnet werden. Diese Änderung wurde notwendig, da das System jetzt den Sensorkopf zur Hindernisvermeidung aktiv neigt.

Die bei den Experimenten erfassten Daten sind auf dem DFKI-internen Fileserver abgelegt worden. Zusammenfassend sind 214 Sätze von Experimentaldaten mit einem Gesamtvolumen von 822GB erhoben worden.

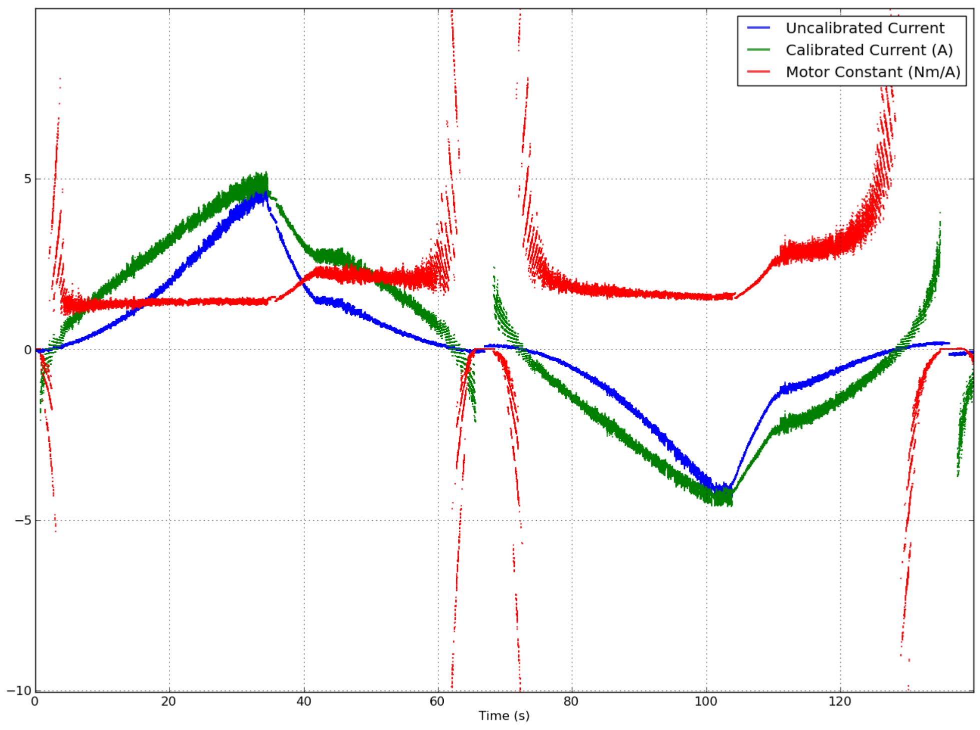


Abbildung 10: *Ein Diagramm, das den unkalibrierten Strom, den kalibrierten Strom und die resultierende Motorkonstante zeigt.*

3.2.1 Meilensteine

Um den Fortschritt des Projektes überprüfen zu können, sind drei Meilensteine absolviert worden.

- Meilenstein 1 - Navigation auf der Teststrecke mit Hilfe von GPS und a-priori Karte
- Meilenstein 2 - Navigation auf der Teststrecke ohne GPS und mit a-priori Karte
- Meilenstein 3 - Navigation auf einem unbekanntem Gelände ohne GPS und ohne a-priori Karte

Meilenstein 1 Im ersten Quartal 2010 ist der Meilenstein 1 (autonome Navigation in bekanntem Gelände und mit Hilfe von GPS) erfolgreich durchgeführt worden, und am 19.5.2010 im DFKI dem Projektträger präsentiert worden. Abb. 11 zeigt die vorgegebene und tatsächlich gefahrene Wegstrecke.

- 3D-Punktwolke → Befahrbarkeitskarte (s. Abb. 11) → Pfad → Pfadverfolgung
- Posenschätzung über GPS
- Fahrt um das Gebäude aus Sicherheitsgründen vermieden (Teich)
- Video auf Wunsch verfügbar

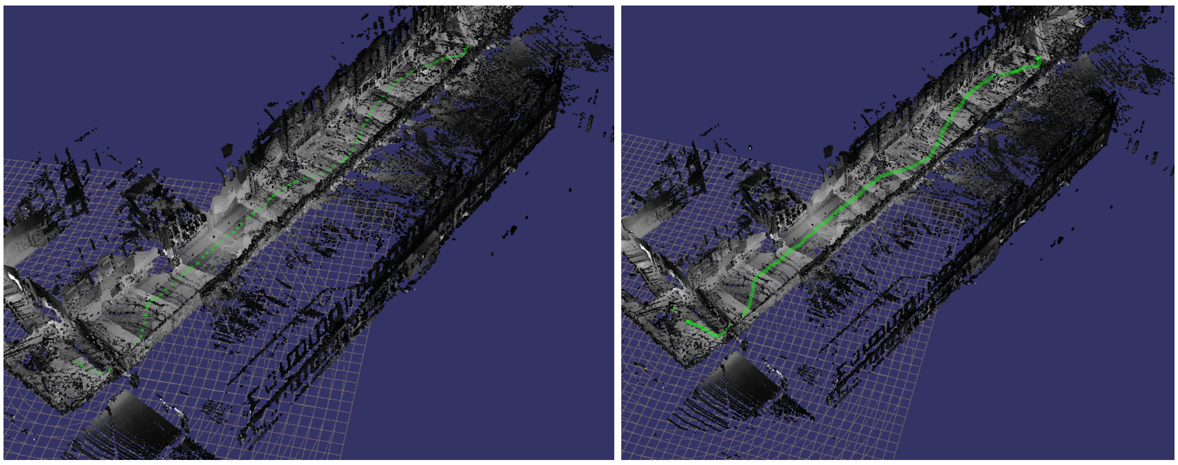


Abbildung 11: *Meilenstein 1: (links) Vorgegebener Pfad auf der DFKI-Teststrecke. (rechts) Tatsächlich gefahrene Wegstrecke des Roboters ASGUARD.*

Meilenstein 2 Der zweite Meilenstein des iMoby-Projektes ist dem Projektträger am 16.6.2011 in Form einer Videopräsentation vorgestellt worden. Der Inhalt des zweiten Meilensteins war die autonome Navigation auf der Teststrecke ohne Zuhilfenahme von GPS-Daten. Die Ziele des Meilensteins sind nach Absprache mit dem Mittelgeber im

Bezug auf den ursprünglichen Plan abgeändert worden. Der Ursprungsplan sah vor, bereits ohne a-priori-Karte auszukommen. Aufgrund der problematischen Situation mit der Nutzung von GPS-Daten auf der Teststrecke des DFKI wurde schon auf die Nutzung von GPS verzichtet, aber noch weiterhin a-priori-Information verwendet (Siehe Abb. 12).

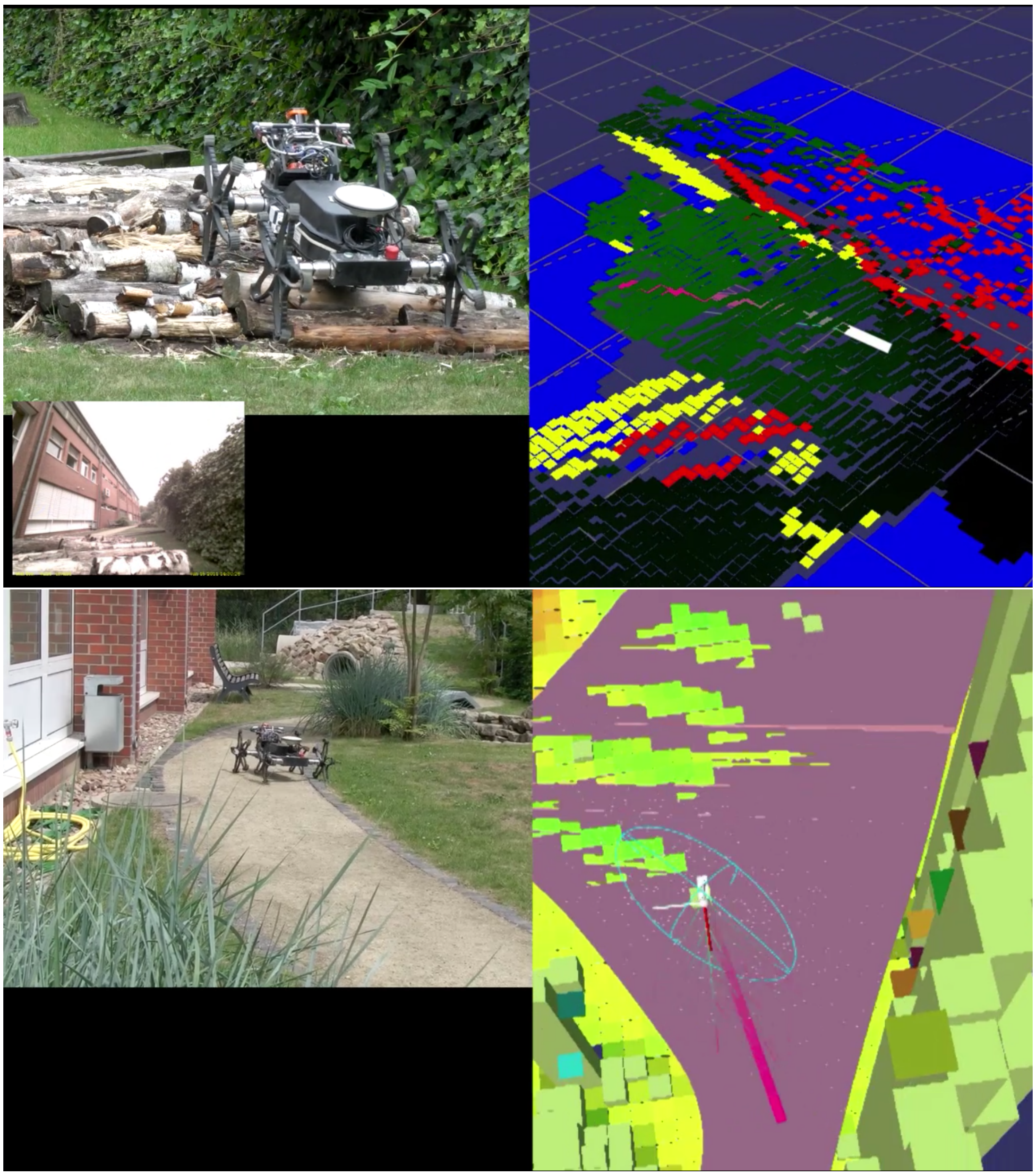


Abbildung 12: *Meilenstein 2: (oben) Lokale Karte des Corridorservoings, das eine lokale Hindernisvermeidung implementiert und auch schwierige Hindernisse wie den Holzstapel oder die Brücke auf der DFKI-Teststrecke autonom überwinden kann. (unten) In einfacheren Passagen wird das Korridor-Following-Verhalten angewandt, welches weniger rechenintensiv ist und somit schneller durchgeführt werden kann.*

Meilenstein 3 Der Meilenstein 3 wurde am 8.8.2012 in Form einer Videopräsentation beim Projektträger vorgestellt. Der Inhalt des 3. Meilensteins ist die autonome Navigation auf dem Testgelände ohne Zuhilfenahme von GPS-Daten oder a-priori-Karteninformation. Der Test ist auf einem Gelände mit sandigem Untergrund und leichtem Bewuchs durchgeführt worden. Der Bewuchs stellt Hindernisse dar, die eine aktive Nutzung verschiedener im Projekt entwickelter Techniken erfordert. Zuerst wird ein Punkt 50 Meter entfernt vom Ausgangspunkt angefahren. Die dabei erstellte Karte wird genutzt, um einen Plan zurück zum Startpunkt zu erstellen (siehe Abb. 13). Bei Fehlern in der Durchführung des Plans werden verschiedene Techniken eingesetzt, um einen Neuplanungsschritt durchzuführen.

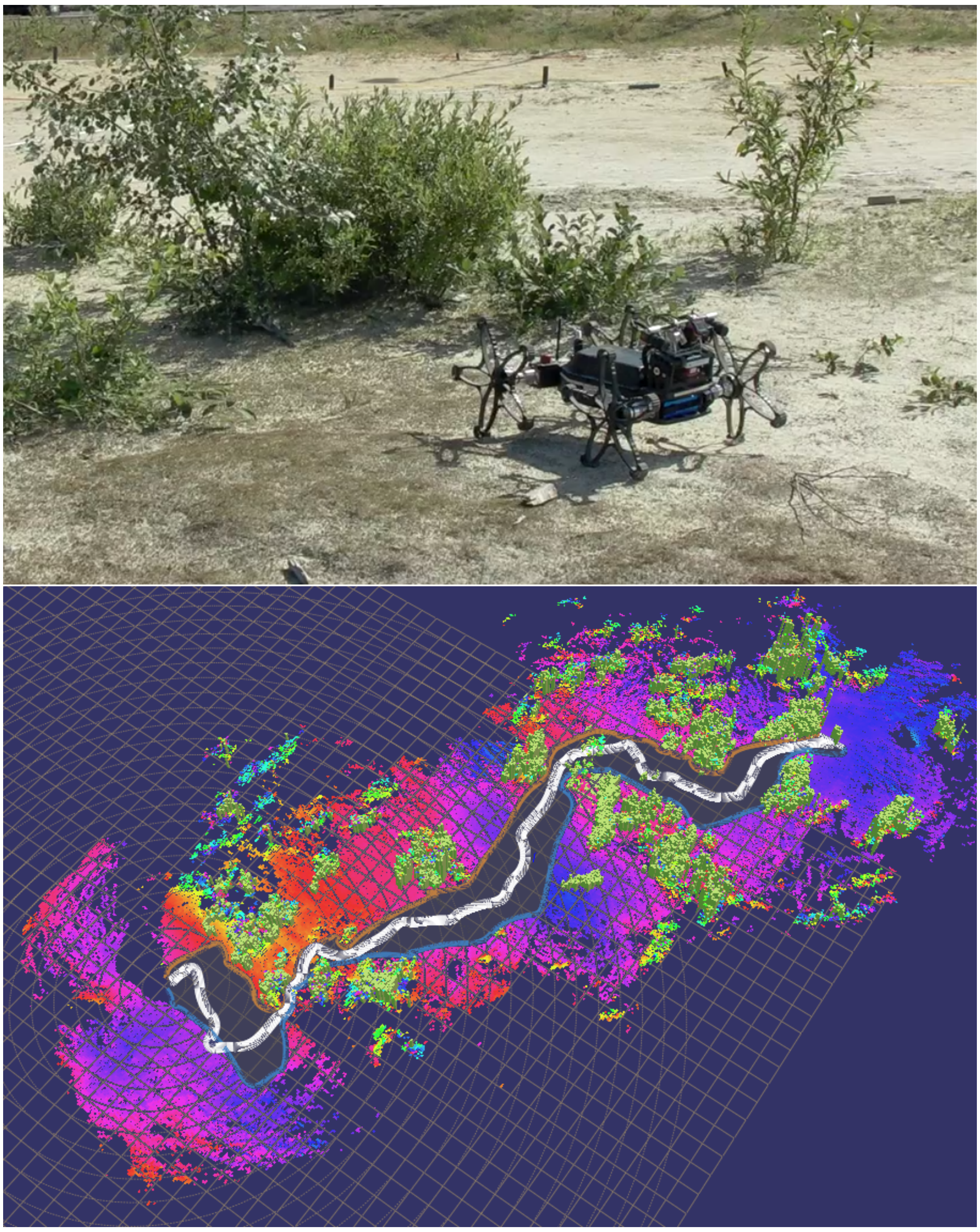


Abbildung 13: *Meilenstein 3: (oben) Der Meilenstein 3 ist auf eine Sandfläche mit leichtem Bewuchs durchgeführt worden. Das Gelände ist dem Roboter unbekannt. Bei der Exploration zu einem 50 Meter entfernten Zielpunkt erzeugt das System eine Karte, die dann zur Planung des Rückwegs genutzt wird (unten).*

3.3 AP3000: Plattform-Mobilität

Die Aufgabe dieses Arbeitspaketes besteht in der Festlegung und Implementierung der Bewegungs- und Mechanikaspekte sowie der Low-Level Regelung. Es ist geplant, im Laufe des Projektes zwei Versionen der Hardware-Plattform zu bauen; eine am Anfang zur Unterstützung der Arbeit (AP3100) und eine zum Abschluss, um auf den Ergebnissen des bisherigen Projektes aufzubauen (AP3500). Der Rest des Arbeitspaketes beschäftigt sich mit der Implementierung und Untersuchung der Steuerungsmethoden (AP3200), die an das geplante Fuß-Rad-Design angepasst wurden, sowie mit der Optimierung der für die Fortbewegung wichtigsten Hardwarekomponenten, den Rädern und dem Design des Fußes (AP3400).

WP3100: Integration Referenzsystem

Die Anforderungen an den neuen Asguard-Prototypen wurden gesammelt, und in Form eines Referenzsystems umgesetzt.

Das **Referenzsystem** (Abbildungen 14 und 15) basiert auf der neuen, modularen Asguard-Plattform Version 3, die mehr Möglichkeiten zur Erweiterung bereithält. Dabei konnten das **neue Gehäuse** (Abbildung 14) und das leistungsfähigere Computersystem (Abbildung 14) erstmals ausgiebig getestet werden. Der **Sensorkopf** (Abbildung 14) wurde mit den wichtigsten Sensoren bestückt, womit erste Erkenntnisse mit der neuen Plattform gesammelt werden konnten.

Folgende Sensoren wurden für das Referenzsystem beschafft und zusammen mit den bereits vorhandenen integriert (siehe Abbildung 16):

- Kamera: HDR-fähige AVT-Guppy-Kamera
- Laserscanner: Hokuyo URG-04LX
- PMD-Kamera: PMD[vision] CamCube 2.0
- IMU: XSens IMU und DFKI IMU

Weiterhin wurden die in Abschnitt 3.2 beschriebenen **OCU**- und **dGPS-Base-Station**-Komponenten entwickelt. Wichtiger Meilenstein bei der Entwicklung des neuen Referenzsystems waren die elektronischen Module *Power Supply*, *Main Board* und *Supervisor Board* und die Integration mit dem **PC104-Stack**.

Des Weiteren wurden für alle elektronischen Module Ersatzteile gefertigt, um einen reibungslosen Verlauf des Projektes auch bei Ausfall einzelner Module zu gewährleisten. Das Referenzsystem ist durch eine Abdeckung aus GFK (Abb. 17) erweitert worden, die innen mit einem dünnen Kupferblech ausgekleidet ist. Dieser Schritt ist hilfreich, um die Schirmung der elektronischen Komponenten zu verbessern. Elektromagnetische Interferenzen haben in der Vergangenheit zu Problemen mit der Qualität des GPS-Signals geführt.

Es folgen nun Details zu den einzelnen Komponenten des Referenzsystems.

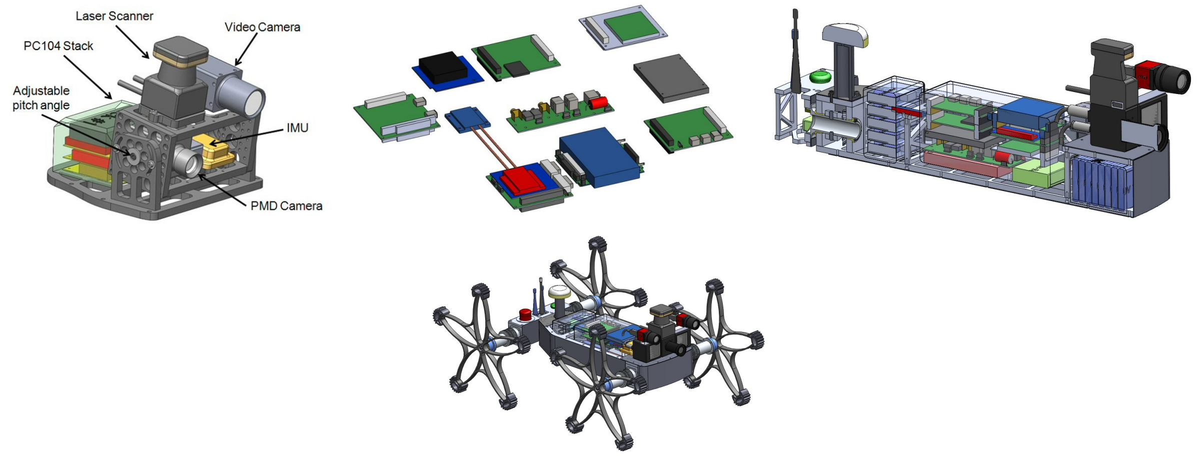


Abbildung 14: *Teile des Asguard-Referenzsystems: Gesamtsystem, Asguard-Sensorkopf, PC-Komponenten, Asguard-Rumpf*

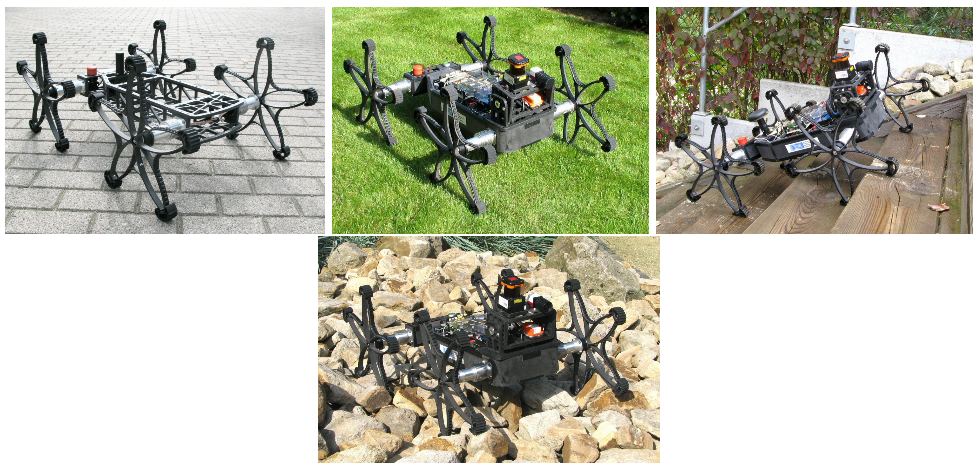


Abbildung 15: *Asguard im Außeneinsatz*

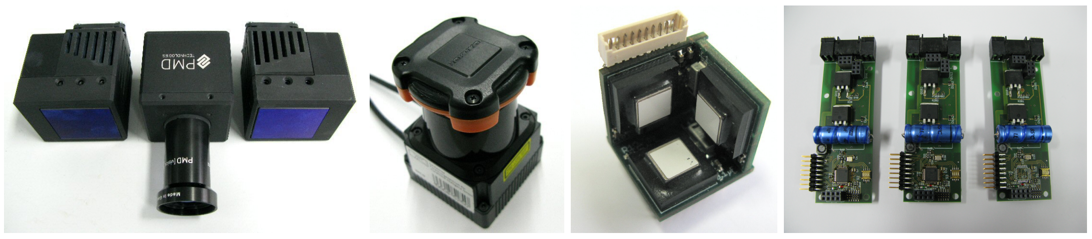


Abbildung 16: *Sensoren des Asguard-Referenzsystems: PMD-Kamera, Hokuyo-Laserscanner, DFKI-IMU, H-Brücken-Module*

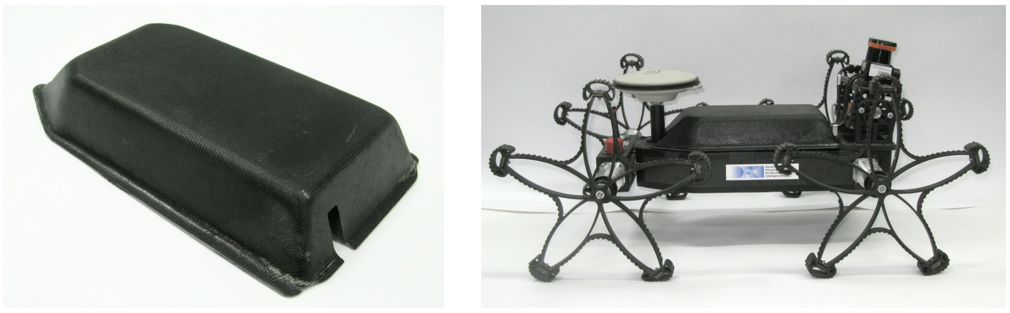


Abbildung 17: *Abdeckung aus GFK für mechanischen Schutz und EM-Abschirmung*

3.3.0.1 Radsensormodul Die verwendeten Motoren verfügen über integrierte magnetische Sensoren, die zur präzisen Geschwindigkeitssteuerung aller Räder genutzt werden.

Aufgrund der elastischen Kupplung zwischen Motor und Rad ist ein zusätzlicher Radsensor erforderlich, um die absolute Position aller Radfüße in Echtzeit zu messen.

Wasserdichte Sensormodule wurden entworfen, konstruiert und getestet (Abb. 18).

Der Radsensor ist eine modulare Einheit, die zwischen dem Rad und dem Motormodul montiert werden kann. Die Enkoderscheibe weist 1024 Inkremente pro Umdrehung auf. Der gleiche optische Enkoder wird zur Erfassung des Winkels des Körperfreiheitgrades verwendet.

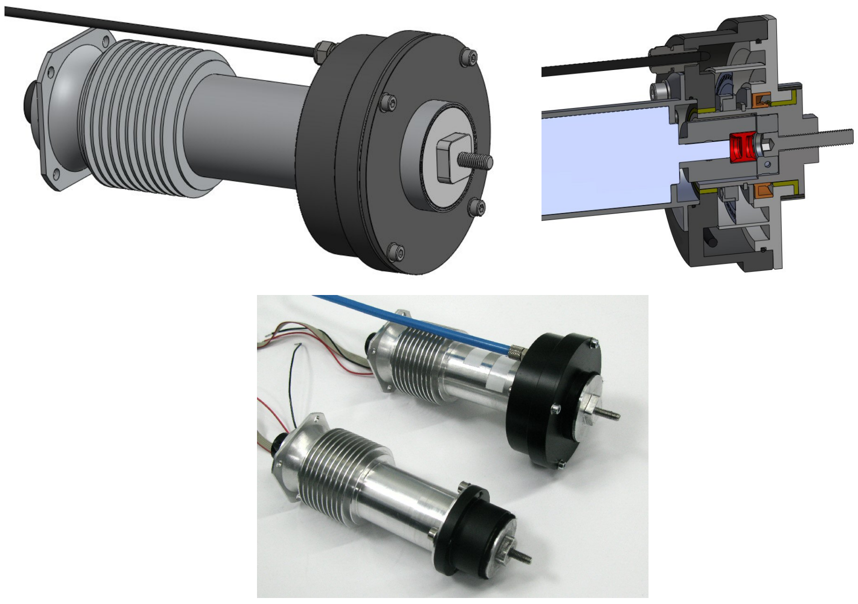


Abbildung 18: *Radsensormodul*

3.3.0.2 Körperfreiheitsgradsensor ASGUARD weist einen passiven Körperfreiheitsgrad auf, der es dem System erlaubt, sich an unebenes Terrain anzupassen.

Zuvor verfügte die Steuerungssoftware über keinerlei Information in Bezug auf den Winkel dieses Gelenks. Daher wurde ein optischer Enkoder (mit Enkoderscheibe) integriert, um die Winkelposition dieses Freiheitsgrades in Echtzeit zu erfassen (Abb. 19). Der Bewegungsumfang des Körpergelenks liegt bei +/- 35 Grad. Die Enkoderscheibe weist 1024 Inkremente pro Umdrehung auf.

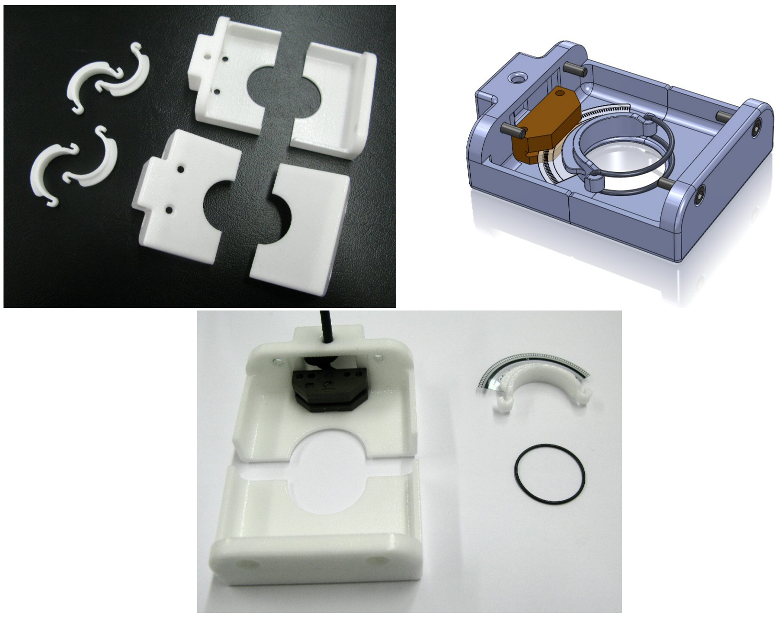


Abbildung 19: *Körperfreiheitsgradsensor*

3.3.0.3 Motortreiber Die Firmware der Motortreiber wurde modifiziert, um die Daten des zweiten Radenkoders auszulesen.

Die Motortreiber können nun über einen GPIO abgeschaltet werden.

3.3.0.4 Hauptplatine Die Hauptplatine wurde so modifiziert, dass die Motortreiber über einen GPIO abgeschaltet werden können. Dadurch ist es möglich, die Motortreiber auch dann abzuschalten, wenn das PC-System versucht, sie über den CAN-Bus neu zu starten.

3.3.0.5 Entwicklung des Sensorkopfes Um Ideen im Bereich der Sensorintegration zu testen, wurde eine Halterung für die Sensorik entwickelt (Sensorkopf). In Abbildung 20 ist die Halterung mit diversen Sensoren zu sehen.

Aufgrund der sternförmigen Räder des Roboters ASGUARD ist das System während der Fortbewegung beträchtlichen Vibrationen und Stößen ausgesetzt. Um exakte Sensordaten der Umgebung zu erhalten, ist es wünschenswert, die auf dem Roboter befindlichen Sensoren (z.B. Stereokamera und Laserscanner) zu stabilisieren.

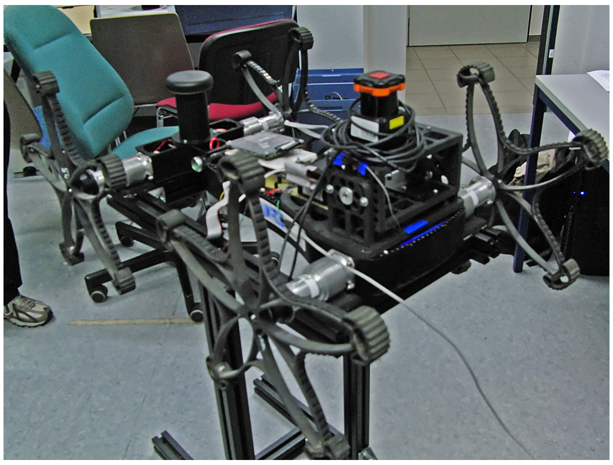


Abbildung 20: *Erster Prototyp des Sensorkopfes. Dabei handelt es sich um eine feststehende Halterung für die Sensorik. Sie dient auch dazu, weitere Informationen zur möglichen Befestigung von Sensoren an der Roboterplattform zu erlangen. Sie ist die Basis für das Design des endgültigen Sensorkopfes.*

Ein erster Sensorkopf-Prototyp mit passiver Stabilisierung wurde konstruiert und getestet. Das Konzept ähnelt dem Design der SteadyCam, die zur Kamerastabilisierung bei Filmproduktionen zum Einsatz kommt. (Abb. 21)

Der Sensorkopf besteht aus einem parallelen Vierstangenmechanismus und einer speziell abgestimmten Feder. Es wurden Luftdruckfedern gewählt, weil hier die Federsteifigkeit über den Luftdruck eingestellt werden kann. Außerdem zeichnen sich Luftdruckfedern durch ihr im Vergleich zu Stahlfedern geringes Gewicht aus.

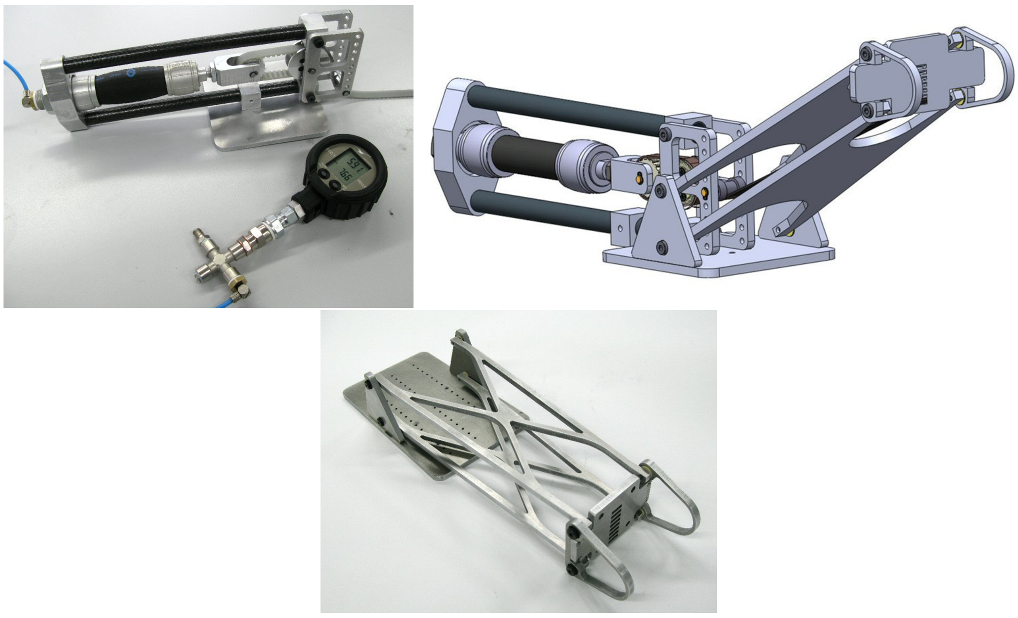


Abbildung 21: *Sensorkopf*

Die Entwicklungen für einen gefederten Sensorkopf haben sich als mechanisch komplex erwiesen. Weiterhin haben sich durch Versuche mit einem starren Konzept keine signifikanten Nachteile für die Sensorwerte ergeben. Aus diesem Grund ist die weitere Entwicklung des gefederten Kopfes nicht weiter verfolgt worden.

Auch ohne die geplante Entwicklung des Sensorkopfes wird eine aktive Neigungsachse für die Sensorik (Laserscanner und Stereokamera) benötigt. Diese ermöglicht eine situationsadaptive Erfassung des Nah- oder Fernbereichs.

- Eine Neigeeinheit für den Laserscanner und die Stereokamera wurde entworfen und angefertigt.
- Die Neigeeinheit wurde am ASGUARD-Sensorkopf montiert. (s. Abb. 22 und 23)

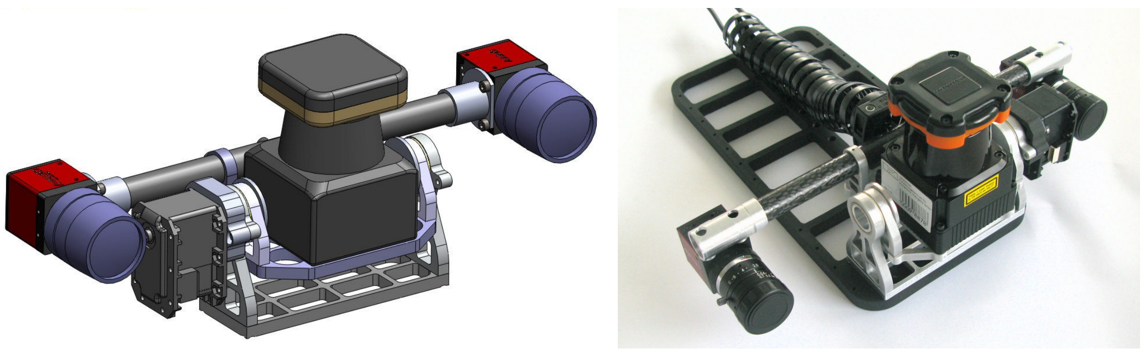


Abbildung 22: *Neigeeinheit, links: Entwurf, rechts: Fertiggestellte Einheit*

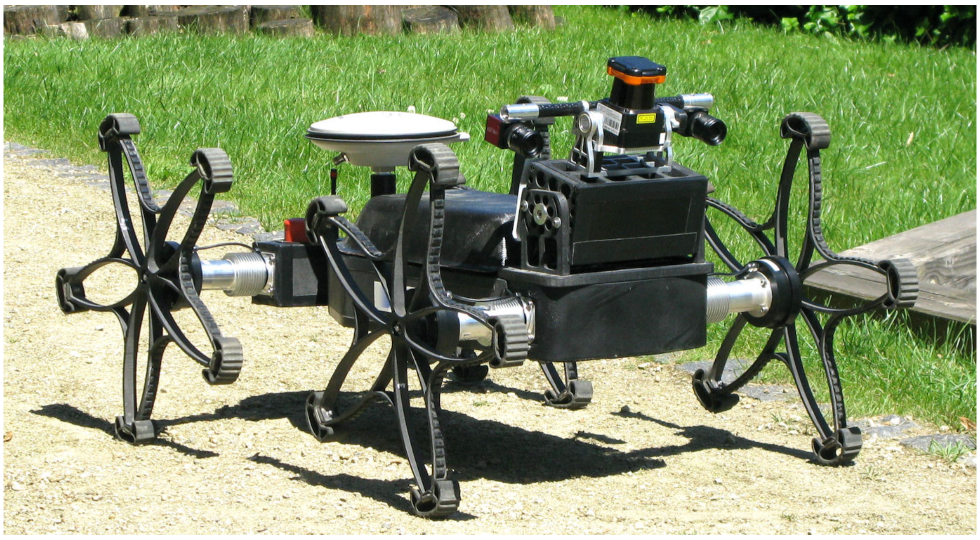


Abbildung 23: *Neigeeinheit, montiert auf dem Roboter ASGUARD*

WP3200: Bewegungsmodalitaten

Grundsätzlich verfolgt dieses Arbeitspaket zwei unterschiedliche Entwicklungsansätze:

- Top-Down-Ansatz. Das Ziel hierbei ist es, ein dynamisches Modell der Asguard-Plattform in Interaktion mit dem Untergrund zu erstellen. In Kooperation mit der Simulationsgruppe wird dabei ein Modell entwickelt, dessen Eigenschaften am Ende mit denen des realen Systems verglichen werden.
- Bottom-Up-Ansatz. Hier werden verschiedene Ideen in Außenversuchen am realen System getestet.

Wir verfolgen diese zwei Ansätze gleichzeitig, da sie sich gegenseitig positiv beeinflussen und jeweils für den anderen Ansatz wertvolle Informationen generieren können. Zudem können wir so kontinuierlich von neuen Entwicklungen profitieren (durch den Bottom-Up-Ansatz), ohne die Grundlage der Modellierung zu verlieren.

Für den Top-Down-Ansatz wurde in Kooperation mit der Simulationsgruppe ein Modell (multi body system model) des Asguard-Systems erstellt, um die Interaktionen des

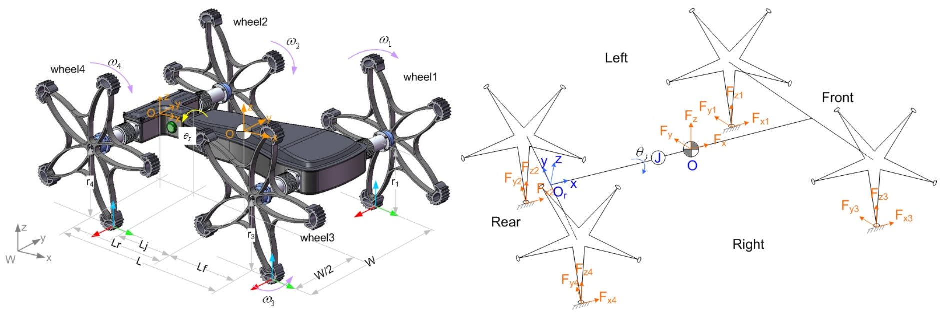


Abbildung 24: *Asguard-Modell: Parameter des Systems, Free Body Diagram*

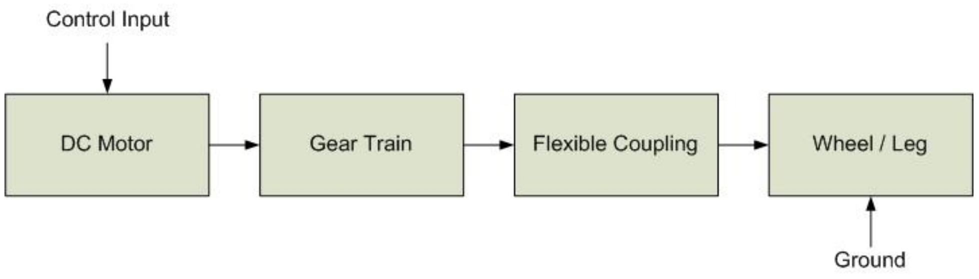


Abbildung 25: *Zusammenhangskette von der Steuerung bis zum Bodenkontakt*

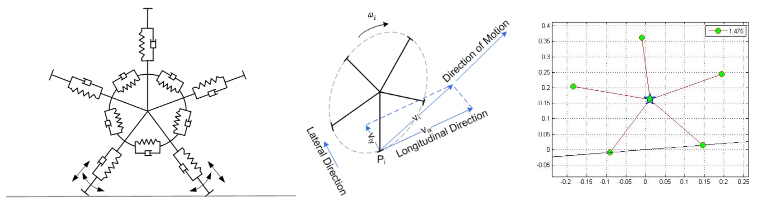


Abbildung 26: *Radmodell: Theoretisches Radmodell, Externe Kräfte, Simulation Visualisierung*

Systems mit seiner Umwelt kontrolliert nachvollziehen zu können. Es ist damit möglich, beide Lokomotionsaspekte – also Bein- und Radantriebe – getrennt oder in Kombination zu untersuchen.

Über ein **Simulationsmodell** (Abbildung 24) können hier diverse Untergründe mit verschiedenen Reibungskonstanten getestet und eine entsprechende **Steuerung** für den Antrieb optimiert werden (Abbildung 25). Ebenso wird das Design des Rades optimiert (Abbildung 26). Dies ist von besonderem Interesse, da sich bei der Iteration IT03 gezeigt hat (siehe Abschnitt 3.2), dass nicht synchronisierte Radstellungen einen sehr starken Einfluss auf die Stabilität des Systems und die Qualität der Kameradaten haben.

Die Daten aus dem Modell wurden genutzt, um einen neuen Motorcontroller (PIV) zu entwickeln, zu implementieren und teilweise zu verifizieren (Abb. 27). Es handelt sich dabei um einen kaskadierten Regler, der eine innere PI-Schleife und einen äußere P-Positionsschleife besitzt. Geschwindigkeit und Beschleunigung werden dabei als Feed-Forward-Signale verwendet. Weiterhin gibt es eine Anti-Windup-Funktion und eine Geschwindigkeitssignalglättung.

Dieser Regler wurde mit einem klassischen PID-Regler verglichen (Abb. 28). Wegen der hohen externen Lasten, ergibt sich in der Geschwindigkeit keine nennenswerte Verbesserung bei der PIV-Regelung gegenüber dem PID-Regler. Jedoch stellt der PIV-Regler sicher, dass sich die aktuelle Position nicht auf Grund von Fehlern in der Geschwindigkeit von der Referenzposition entfernt.

Des Weiteren ist ein Tuning des Reglers vorgenommen worden (Velocity Smoothing, Velocity Loop Integral und Position Loop Tuning, siehe Abb. 29 und 30)

Da eine bessere Bewegungscharakteristik bei synchronisierter Radbewegung festgestellt wurde, wurde der PIV-Regler für diese Aufgabe parametrisiert. Dabei wurden die gegenüberliegenden Räder mit einem Phasenversatz von $\pi/2$ synchronisiert. Weiter-

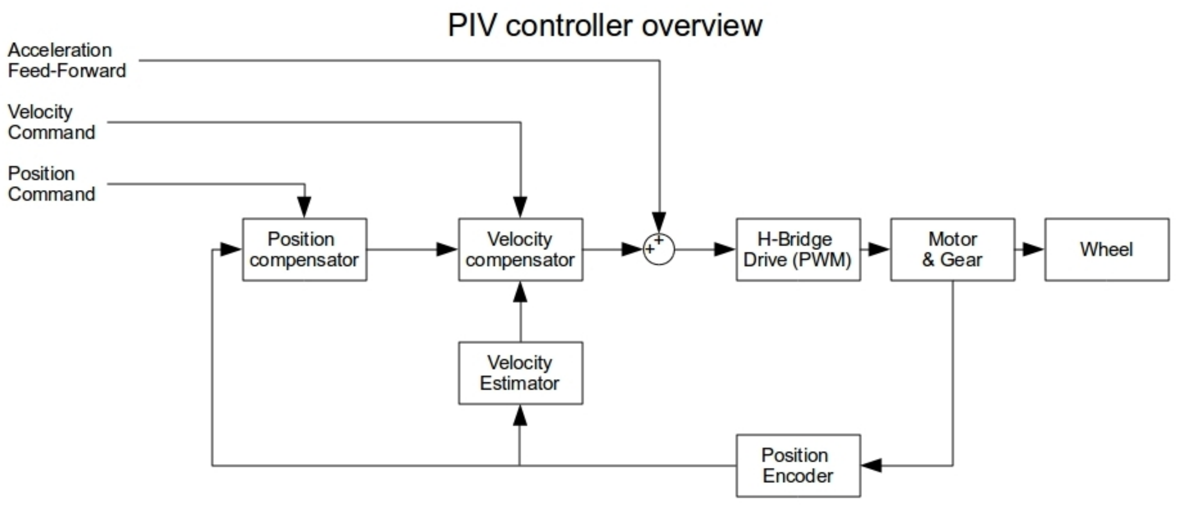


Abbildung 27: *Übersicht der neuen Motorregelung*

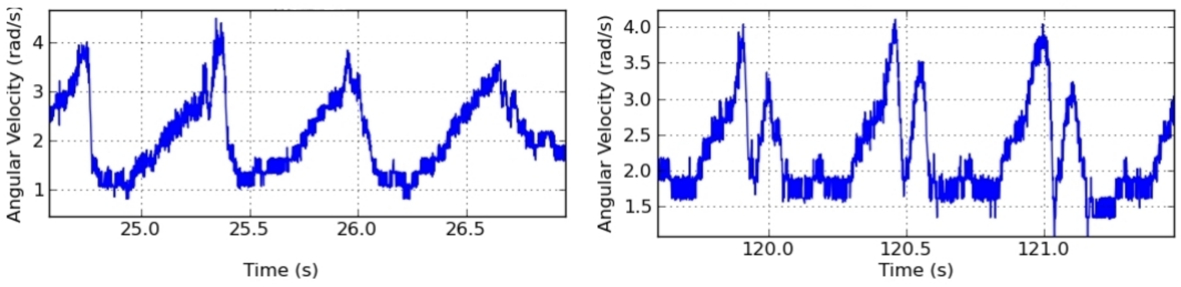


Abbildung 28: *PID (links) vs. PIV (rechts) Regelung*

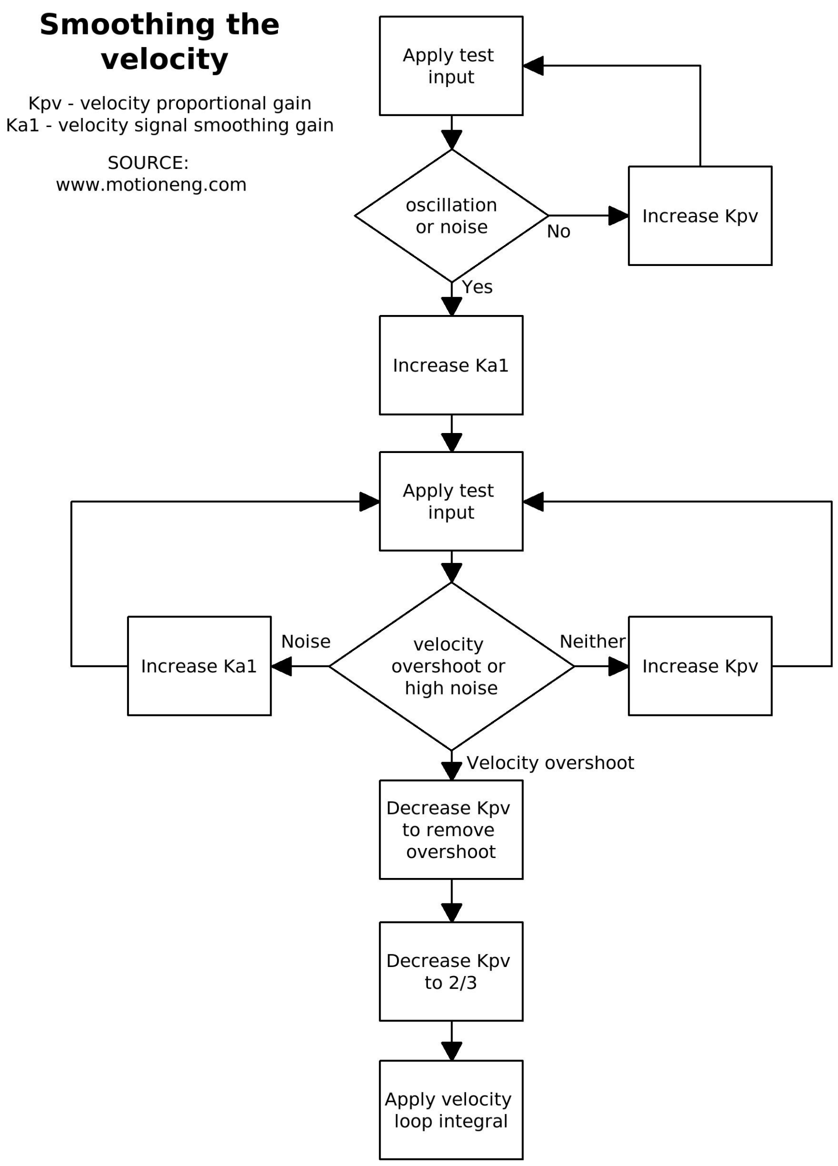

Smoothing the velocity
Kpv - velocity proportional gain
Ka1 - velocity signal smoothing gain
SOURCE: www.motioneng.com
Apply test input
oscillation or noise
No
Increase Kpv
Yes
Increase Ka1
Apply test input
velocity overshoot or high noise
Noise
Increase Ka1
Neither
Increase Kpv
Velocity overshoot
Decrease Kpv to remove overshoot
Decrease Kpv to 2/3
Apply velocity loop integral

Abbildung 29: *Tuning des PIV Reglers - Phase 1*

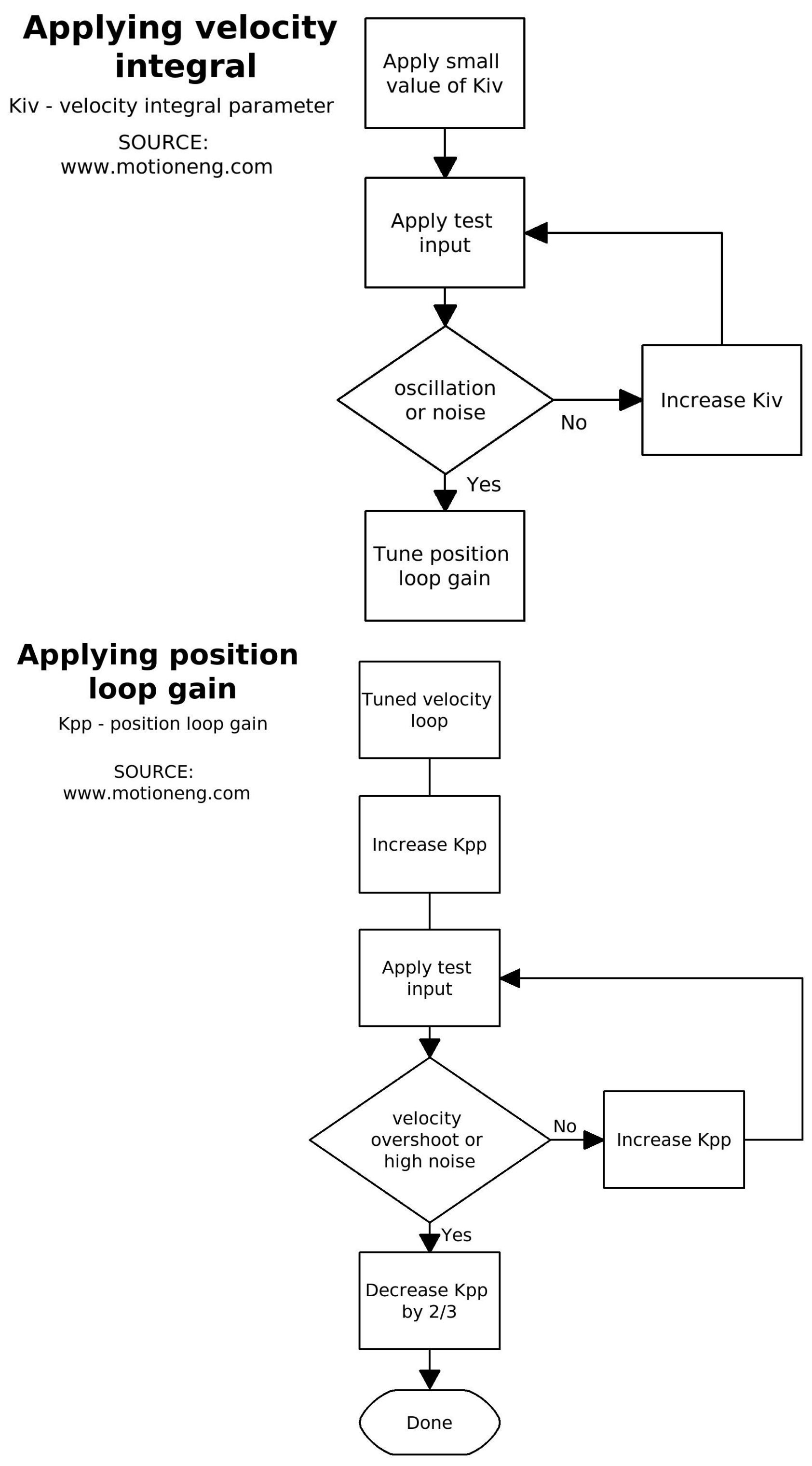


Abbildung 30: *Tuning des PIV Reglers - Phasen 2 und 3*

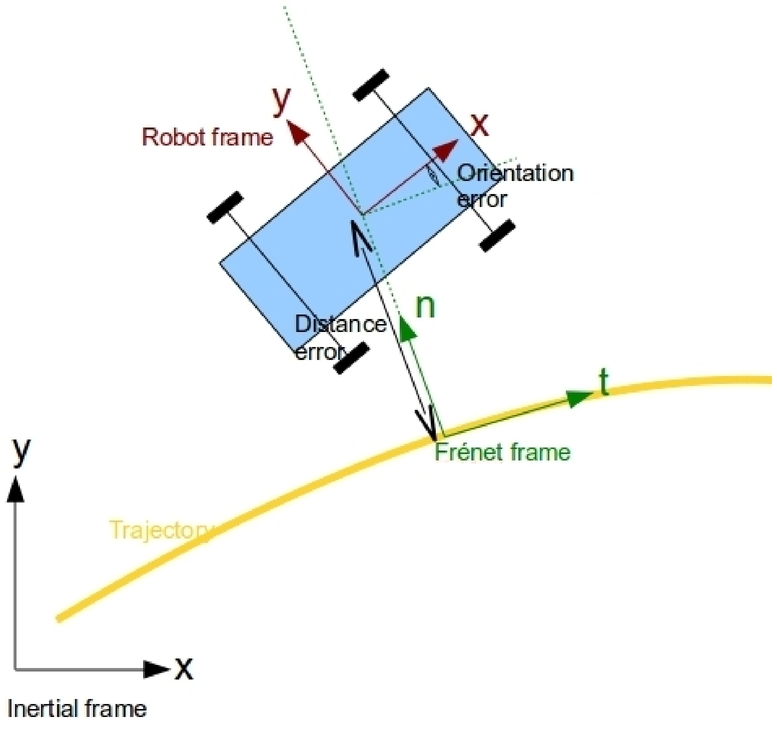


Abbildung 31: *Definition des Fehlers in der Wegpunktnavigation*

hin wurde eine Kalibrierung der Radposition im PWM-Modus durchgeführt.

Außerdem sind Arbeiten an einem Regler für Trajektorienverfolgung durchgeführt worden. Kubische Splines werden mit Hilfe der SISL Library (The SINTEF Spline Library) repräsentiert. Es wird dabei immer der von der aktuellen Position des Roboters nächstgelegene Punkt auf der Trajektorie als Zielpunkt verwendet und die Frenet-Frames als Eingabe für den Trajektorienregler verwendet. Daraus wird der Positions- und Orientierungsfehler gebildet (Abb. 31). Der Regler zur Trajektorienverfolgung ist anhand von verschiedenen Testdaten in der Simulation verifiziert worden (sinusoidal: Abb. 32, linear: Abb. 33).

Parallel zu den Entwicklungen zur Trajektorienverfolgung ist ein Softwaremodul zur Wegpunktnavigation implementiert worden. Dieses Modul ist in einem realen Versuch verifiziert worden (Abb. 34). Das Modul kann sowohl Wegpunkte aufzeichnen als auch abfahren.

3.3.0.6 Radsynchronisierungstests Abb. 35 zeigt die Motordaten bei synchronisierten Rädern (kein Rad-Offset) im Vergleich zu Rädern mit Offset. Die Konfiguration mit Offset führt zu einer verbesserten Vorwärtsbewegung des Systems.

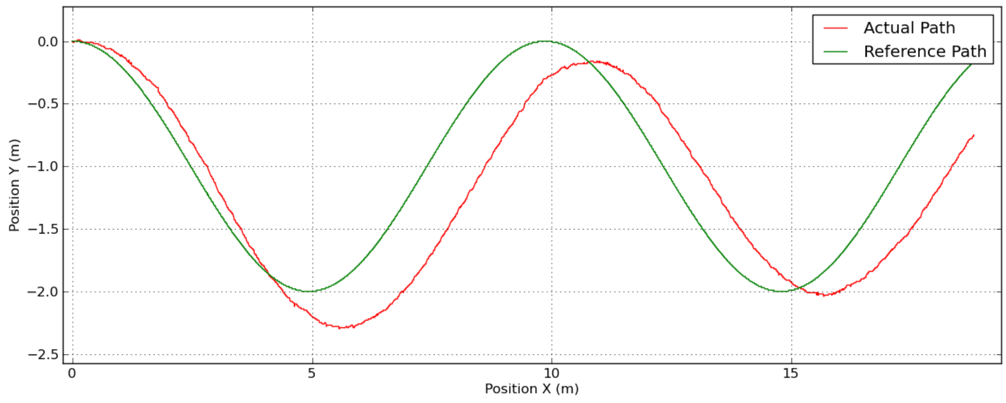


Abbildung 32: *Wegpunktnavigation einer künstlichen sinusoidalen Trajektorie*

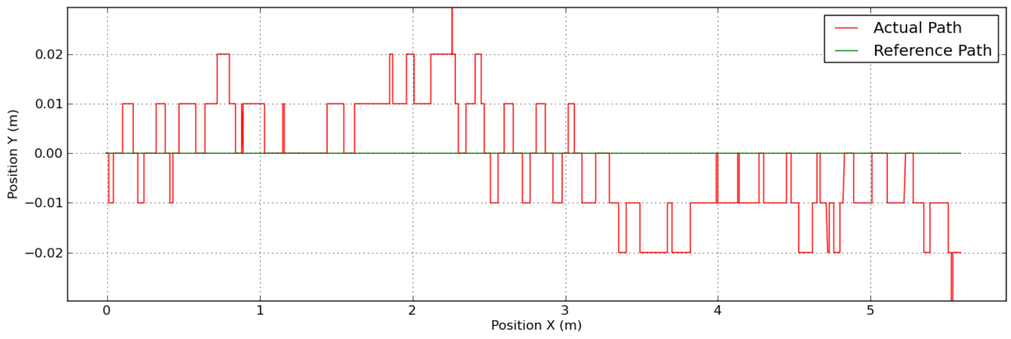


Abbildung 33: *Wegpunktnavigation einer künstlichen linearen Trajektorie*

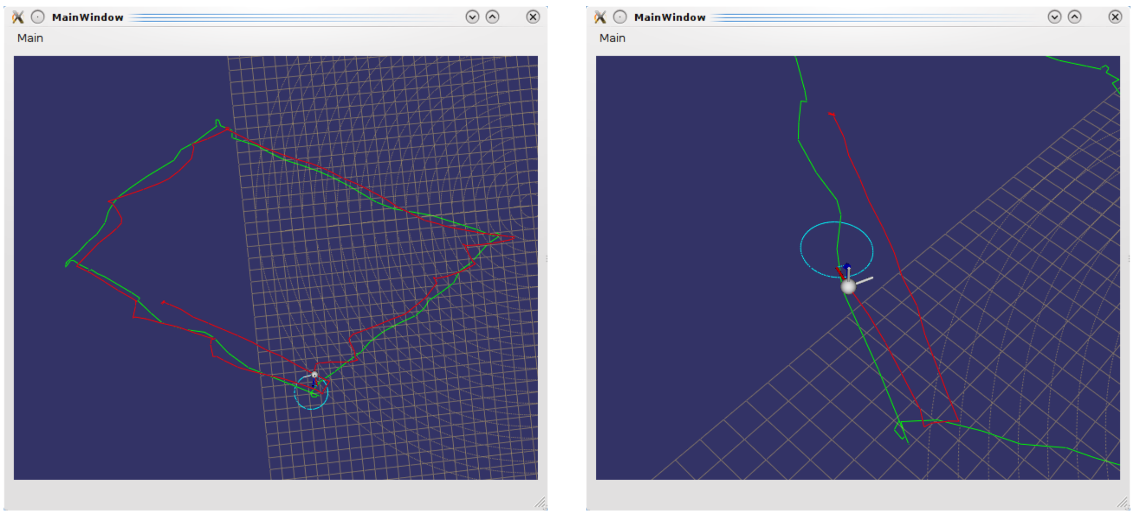


Abbildung 34: *Einfache Wegpunktnavigation auf einer Sandfläche in der Nähe des DFKI-Gebäudes. Rechts: Detail*

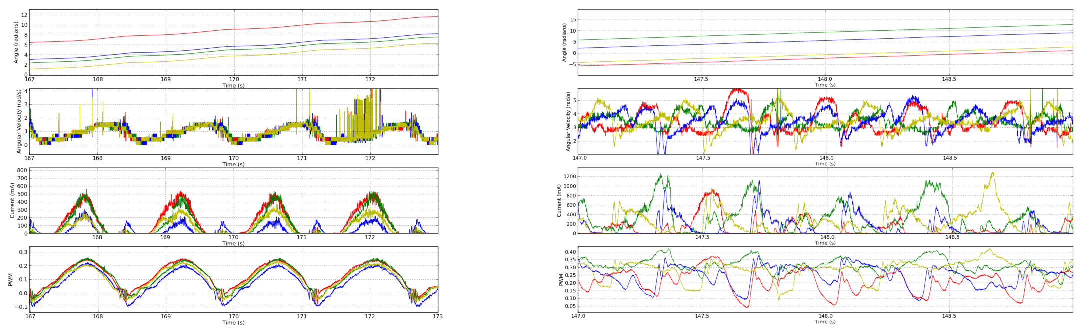


Abbildung 35: *Radsynchronisierungstests*

3.3.0.7 Kurven-Klasse und Trajektoriencontroller Integration und Tests wurden zunächst in der Simulation durchgeführt. Danach erfolgte die Integration auf dem Roboter, die Testergebnisse sind in Abb. 36 dargestellt. Der Fehler des Trajektorien-Controllers ist in Abb. 36 unten dargestellt.

Es hat sich gezeigt, dass der Controller ohne aktive Orientierungsregelung funktioniert.

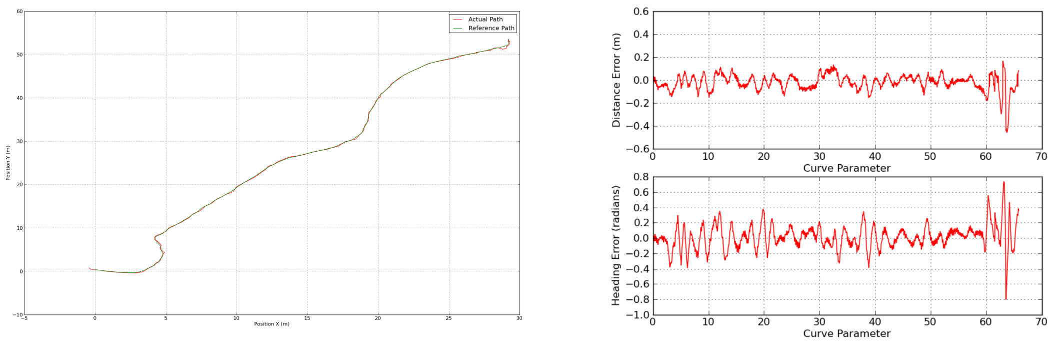


Abbildung 36: *Trajektoriencontroller*

3.3.0.8 Erweiterter Trajektorien-Controller Das Design dieses Controllers basiert auf der “chained form” für nicht holonome Systeme. Es wurde vom Modell eines Einrad-Roboters abgeleitet. Eine aktive Regelung der Orientierung wurde implementiert. Die Ergebnisse der Tests in der Simulation zeigt Abb. 37.

Die Auswirkungen verschiedener Radsynchronisierungen auf die Effizienz der Fortbewegung wurden untersucht (Publikation i-SAIRAS 2010). Es hat sich herausgestellt, dass eine Synchronisierung der linken Räder und eine davon unabhängige Synchronisierung der rechten Räder zu guten Ergebnissen führt. Als positiver Nebeneffekt ist hier eine Drehung des Roboters ohne Verlust der Synchronisierung möglich.

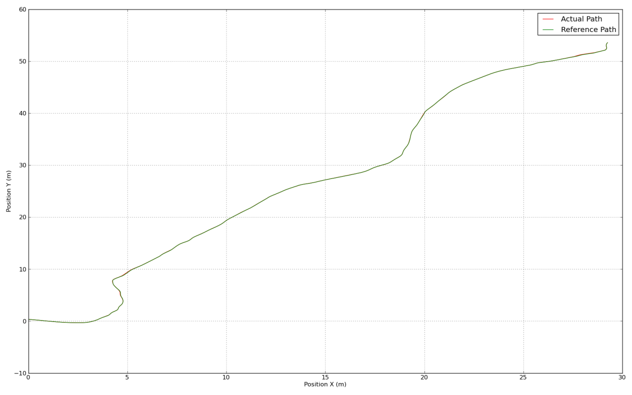


Abbildung 37: *Erweiterter Trajektorien-Controller*

Die Publikation zur Effektivität der Radsynchronisierung ist in Sektion 5.5 eingefügt.

WP3300: Bewegungsmodelle und Terrainerkennung

3.3.0.9 Bewegungsmodelle Es wurde ein einfaches Odometriemodell erstellt und dieses für die Nutzung in WP4200 durch ein stochastisches Modell erweitert.

In Tests hat sich gezeigt, dass für das Befahren schmaler Passagen eine angepasste Navigationsstrategie erforderlich ist. Daher wurde ein Verfahren implementiert, das Laserscans nutzt, um mit einem Visual-Servoing-Ansatz Korridoren zu folgen.

Der statische Fall der Bodenreaktionskräfte wurde analysiert und ein entsprechendes Modell entwickelt. Dieses kann unter anderem dazu genutzt werden, auftretende Drehmomente an den Rädern abzuschätzen und mit den anhand von internen und externen Encodern gemessenen Werten zu vergleichen.

Darüber hinaus wurde ein dynamisches Bewegungsmodell für den Roboter Asguard entwickelt, welches u.a. zur Feinabstimmung des Reglers genutzt werden kann.

Die Drehmomente an den Radkupplungen waren Gegenstand verschiedener Arbeiten. Zum Einen wurden hier Kalibrierungen durchgeführt, zum Anderen wurde ein Software-Modul implementiert und getestet, welches die Drehmomente an allen vier Rädern ausliest. Außerdem wurde ein GPI-Regler (Generalized Proportional Integral) entwickelt, der durch die Elastizität der Kupplungen auftretende Effekte z.T. kompensieren kann.

Um das Bewegungsverhalten des Asguard-Systems zu verbessern, ist ein GPI-Regler getestet worden. Nach einer durchgeführten Systemidentifikation und Anpassung des Reglermodells ist der Regler am existierenden Radteststand getestet worden. Der GPI-Regler wurde ursprünglich für kleine Auslenkungen im Bereich von bis zu 5° entwickelt. Die Auslenkung der Kupplung im ASGUARD ist jedoch größer, was zu Problemen mit der Performance oder sogar der Stabilität des Reglers führt. Aufgrund der hohen Elastizität der Radkupplung hat der Regler keine zufriedenstellenden Resultate geliefert.

Um die Effekte der Elastizität zu verringern ist ein sogenanntes “Input Shaping” angewandt worden, welches die Dynamik des Systems verändert, indem das Eingangssignal modifiziert wird. Am Teststand sind dabei gute Resultate erzielt worden, wenn keine Last am Rad angebracht war.

Ein neuer Regler, der das Drehmoment kontrolliert, wurde implementiert und erprobt. Der Regler ermöglicht es, Drehmomentprofile für die Räder zu definieren, die anschließend abgefahren werden.

Es wurde ein kaskadierter Position-Geschwindigkeit-Drehmoment-Regler (PVT) implementiert und verifiziert, der die Sensorinformation der äußeren Radenkoder nutzt. Tests mit diesem Regler haben gute Resultate gezeigt. Die Funktion des Reglers verschlechtert sich, wenn die verwendeten Kupplungselemente über den Verlauf der Nutzung weicher werden. Dies kann durch eine Rekalibrierung des Kupplungsmodells kompensiert werden.

Weiterhin wurden Tests durchgeführt, um zu verifizieren, wie stark die Kupplungen ihre Eigenschaften im Laufe der Nutzung verändern (z.B. weicher werden). Dabei wurde festgestellt, dass die Kupplungen zu Beginn eine starke Veränderung aufweisen und sich später nur noch wenig ändern. Die Tests wurden mit Hilfe von Logdaten durchgeführt. Hier wurden die Motorströme mit den Drehmomentwerten der Kupplungen über mehrere Wochen aufgezeichnet und verglichen.

Es wurde außerdem an einem Regler gearbeitet, der eine Lastverteilung auf alle Kontaktpunkte des Roboters mit der Umwelt durchführt. Das Ziel ist es, ein gleichmäßiges und energieeffizientes Bewegungsverhalten zu erreichen. Dazu wurde ein Drehmomentregler modifiziert, um nur noch die Traktionskräfte und nicht mehr die Gravitationskräfte zu regeln. Tests haben gezeigt, dass dieser Ansatz noch Probleme wie z.B. ungewollt harte Bewegungen mit sich bringt. Diese werden vermutlich durch die unausgeglichene Massenverteilung zwischen den Rädern verursacht. Eine gleichmäßigere Verteilung der Systemmasse zwischen den Rädern war zu diesem Zeitpunkt nicht mehr möglich.

Als finaler Ansatz für den Rad-Controller wurde ein kaskadierter Regler eingesetzt, der eine Kaskade aus Drehmoment-, Geschwindingkeits- und Positionsregelung darstellt. (s. Abb. 38)

Das Drehmoment wird hier über die Deflektion der elastischen Radkupplung geschätzt. Dabei kommt ein Hysteresemodell zum Einsatz (s. Abb. 39).

Es zeigt sich im Vergleich zum reinen Motorregler eine Verringerung des Radpositionierfehlers um etwa 70% (s. Abb. 40). Außerdem ergibt sich eine bessere Radsynchronisierung und eine gleichmäßigere Fortbewegung des Roboters.

Die Auswirkungen unterschiedlicher Rad-Offsets wurden untersucht (s. Abb. 41). Dabei hat sich gezeigt, dass eine kreuzweise Synchronisierung am effizientesten ist, während eine Links-Rechts-Synchronisierung der Räder bei leicht verringerter Effizienz eine differentielle Steuerung des Roboters ermöglicht.

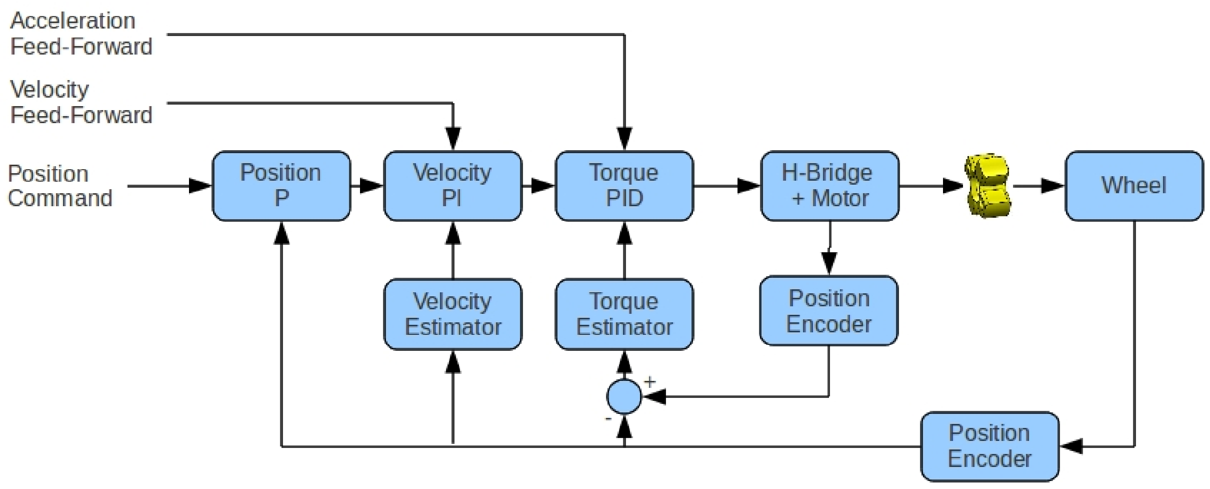


Abbildung 38: *Kaskadierter Rad-Controller*

Das Widerstreben des Systems gegen eine geradlinige Bewegung ist bei vollständiger Kreuzsynchronisierung um das Vierfache geringer als bei identisch positionierten Radfüßen. Die Vibrationen lassen sich um die Hälfte reduzieren, das Regelverhalten des Gesamtsystems verbessert sich.

Der neu entwickelte Regler für die Rotation des Roboters dreht das System während der Phase, in der die Räder positive Drehmomente erfahren. In der folgenden Phase negativer Drehmomente werden die Räder neu positioniert (s. Abbildungen 42 und 43). Die Obergrenze für die PWM-Steuerung der Motoren wird dabei dynamisch gesetzt. Die hinteren Räder drehen sich bei diesem Verhalten mit einer höheren Geschwindigkeit.

Das erreichbare Drehmoment ist durch Nutzung dieses Reglers höher als zuvor (s. Abb. 44).

3.3.0.10 Terrainerkennung Es wurden Versuche durchgeführt, um anhand der Beschleunigungs- und Motordaten des Roboters mittels bekannter Lernverfahren den Untergrund zu klassifizieren. Dazu wurden zunächst Datensätze erstellt, bei denen der Roboter in einer definierten synchronisierten Gangart über verschiedene Untergründe gefahren ist. Auf die erstellten Datensätze wurden anschließend verschiedene Lernverfahren angewandt, um Geländeklassifikatoren zu erzeugen. Die resultierenden Klassifikatoren zeigten schlechte Resultate auf den Evaluierungsdatensätzen. Daraus wurde geschlussfolgert, dass es nicht ohne Vorverarbeitung der Datensätze möglich ist, Geländeklassifikatoren anzulernen.

Optimierung der Drehung: Eine Rotation des Roboters erfordert hohe Raddrehmomente, da die Räder bei dieser Bewegung aufgrund ihrer Anordnung seitwärts rutschen. Je höher die Bodenhaftung ist, desto schwieriger ist die Drehung des Systems.

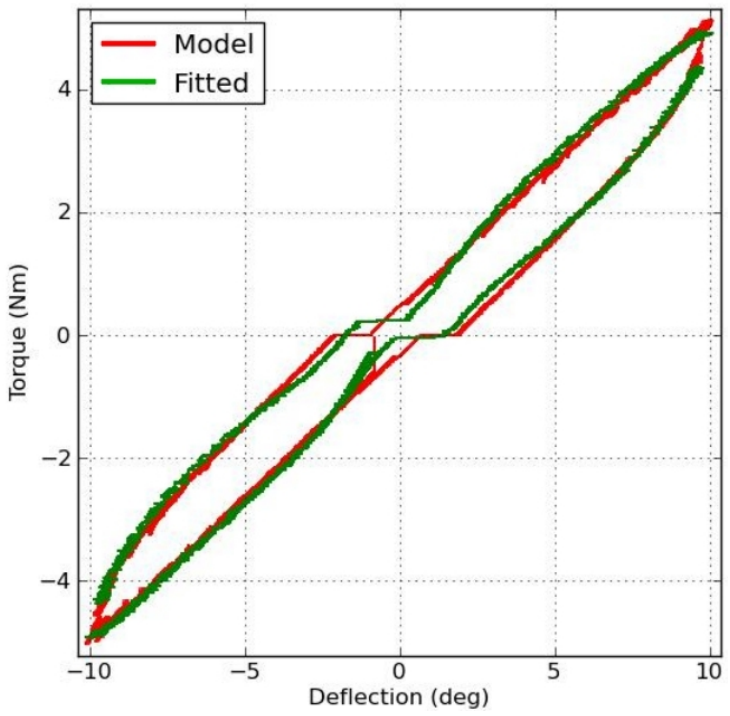


Abbildung 39: *Vom Rad-Controller genutztes Hysteresemodell*

Außerdem führt die hohe Bodenreibungskraft zu einer Verdrehung des passiven Körpergelenks und somit zu einer ungleichmäßigen Verteilung der Drehmomente. Dies führt u.U. zur Überstromabschaltung der Motormodule.

Die Versuche wurden auf einem 2m x 2m großen Stück Kunstrasen mit einer Stärke von 20mm durchgeführt, um dichtes Gras zu simulieren.

Es wurden verschiedene Ansätze zur Verbesserung des Drehvorgangs untersucht:

- **Drehung auf Basis des Geschwindigkeitsreglers:** Eine Drehung, bei der der Standard-Geschwindigkeitsregler genutzt wird, führt bei starker Bodenhaftung immer zur Überstromabschaltung. Außerdem wird das passive Körpergelenk dabei stark verdreht.
- **Drehung mit hoher Hinterradgeschwindigkeit:** Laufen die Hinterräder bei der Drehung mit einer höheren Geschwindigkeit als die Vorderräder, kann auf hartem Untergrund eine Verbesserung der Rotation des Systems beobachtet werden. Die Hinterräder verlieren dabei zeitweise den Bodenkontakt und verringern so den seitwärts gerichteten Teil der Reibungskraft.
- **Dynamische Drehung:** Bei der Nutzung dieses Reglers wird der Drehradius vergrößert, sobald sich die Raddrehmomente in der Sättigung befinden. Versuche zeigen, dass der Regler seine Aufgabe erfüllt, der Wendekreis des Roboters jedoch vergrößert wird.
- **Rotation um einen Drehpunkt:** Bei diesem Regler dreht sich das System um das sogenannte kritische Rad (KR), das sich hier nicht dreht, sondern während

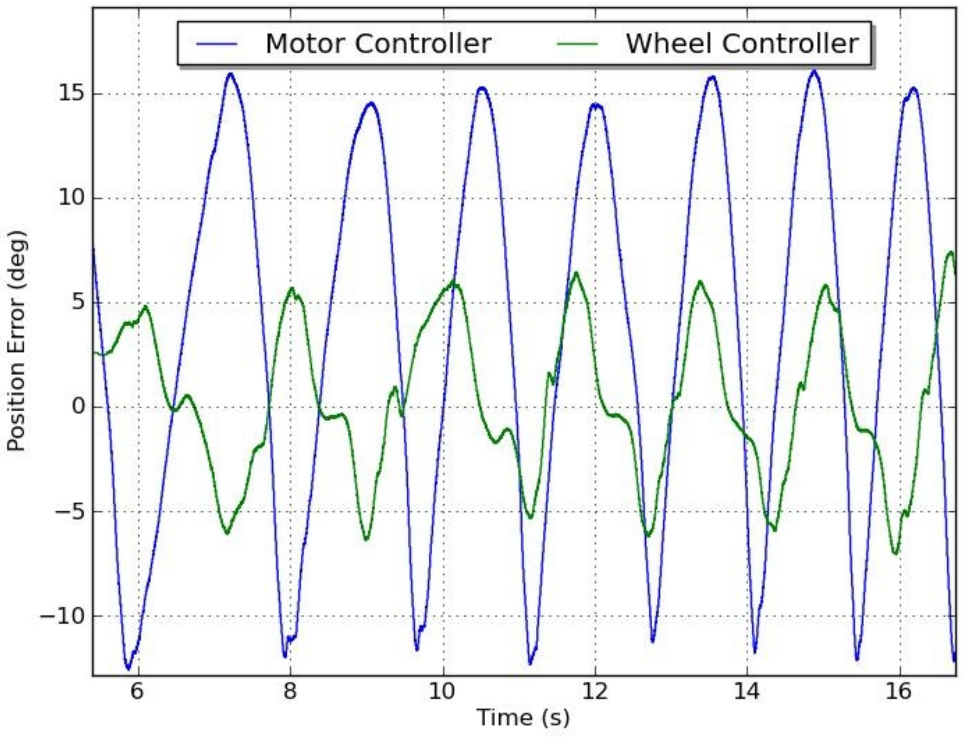


Abbildung 40: *Um etwa 70% verringerter Positionierfehler des Rad-Controllers verglichen mit reiner Motorregelung.*

des gesamten Vorgangs mit nur einem der fünf Radfüße Bodenkontakt hat. KR ist hier das am stärksten belastete Rad. Bei einer Drehung im Uhrzeigersinn ist es das vordere rechte Rad, sonst das vordere linke. Dieser Ansatz zeigt eine Verbesserung des Drehverhaltens, reicht jedoch nicht für Gras aus.

- **Ausschließliche Drehmomentregelung:** Dieser Regler beaufschlagt jedes Rad mit dem maximalen Drehmoment. Versuche zeigen ein unvorhersehbares Verhalten des Gesamtsystems.
- **Zweifuß- zu Einfußstand-Drehung:** Der Roboter dreht sich nur, wenn das kritische Rad vom Bodenkontakt mit einem Fuß zum Kontakt mit zwei Füßen wechselt. Vom Zweifuß- bis zum Einfußkontakt drehen sich alle Räder in dieselbe Richtung. Es wird eine temporäre Reglersättigung auf Basis des Motorstroms verwendet und das maximale Drehmoment begrenzt. Außerdem werden zeitgesteuerte Reglermodi genutzt, um verklemmte Radkonfigurationen zu beheben.

Es hat sich gezeigt, dass die besten Ergebnisse bei der Kombination der Einfuß-zu-Zweifuß-Methode mit der hohen Hinterradgeschwindigkeit erzielt werden.

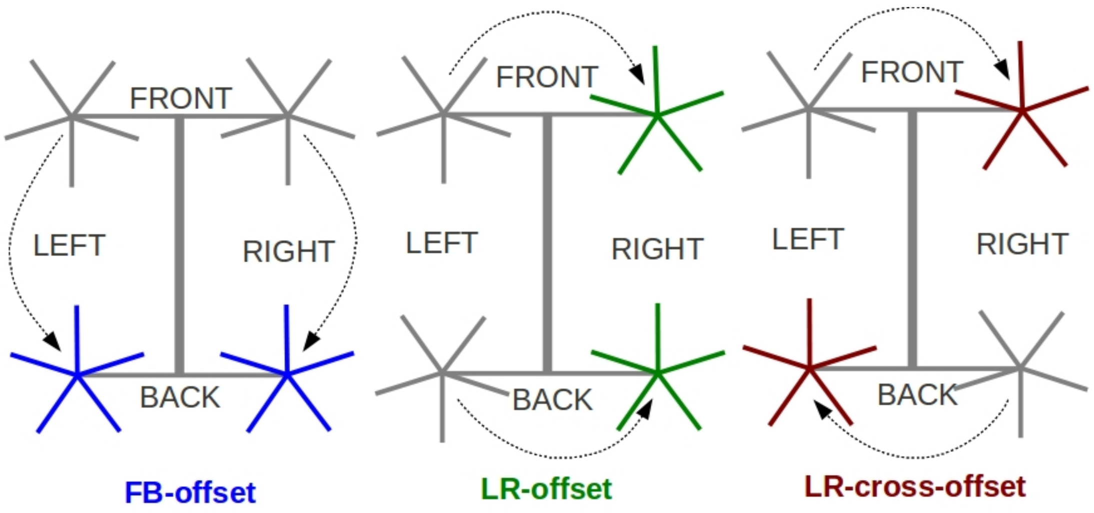


Abbildung 41: *Verschiedene Radsynchronisierungsmuster*

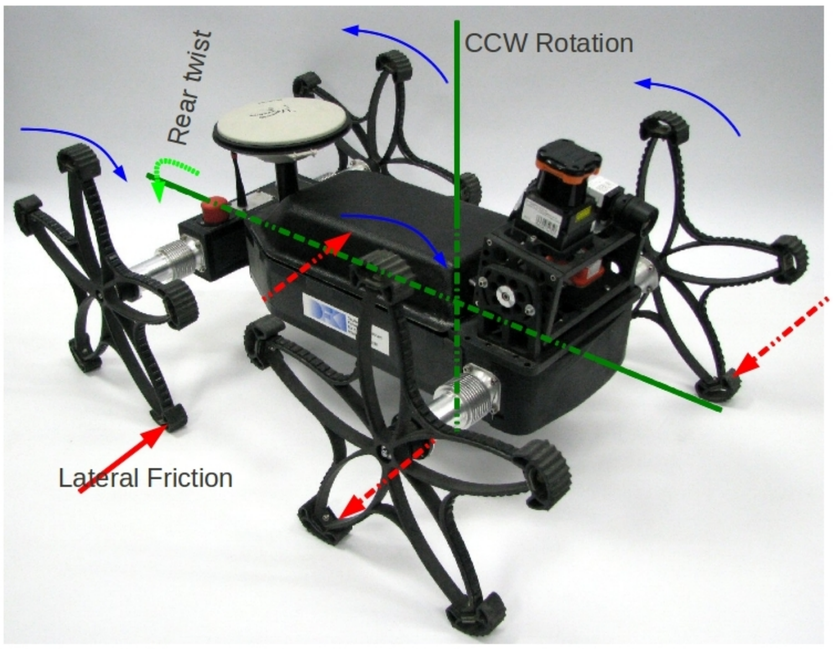


Abbildung 42: *Während einer Drehung auf den Roboter wirkende Kräfte*

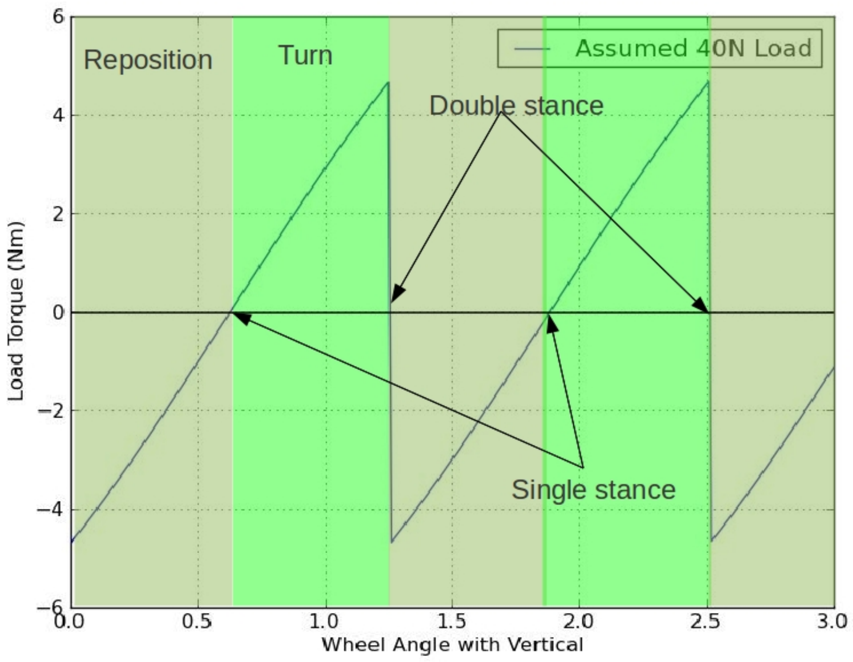


Abbildung 43: *Vorwärts-Rückwärts-Drehverhalten*

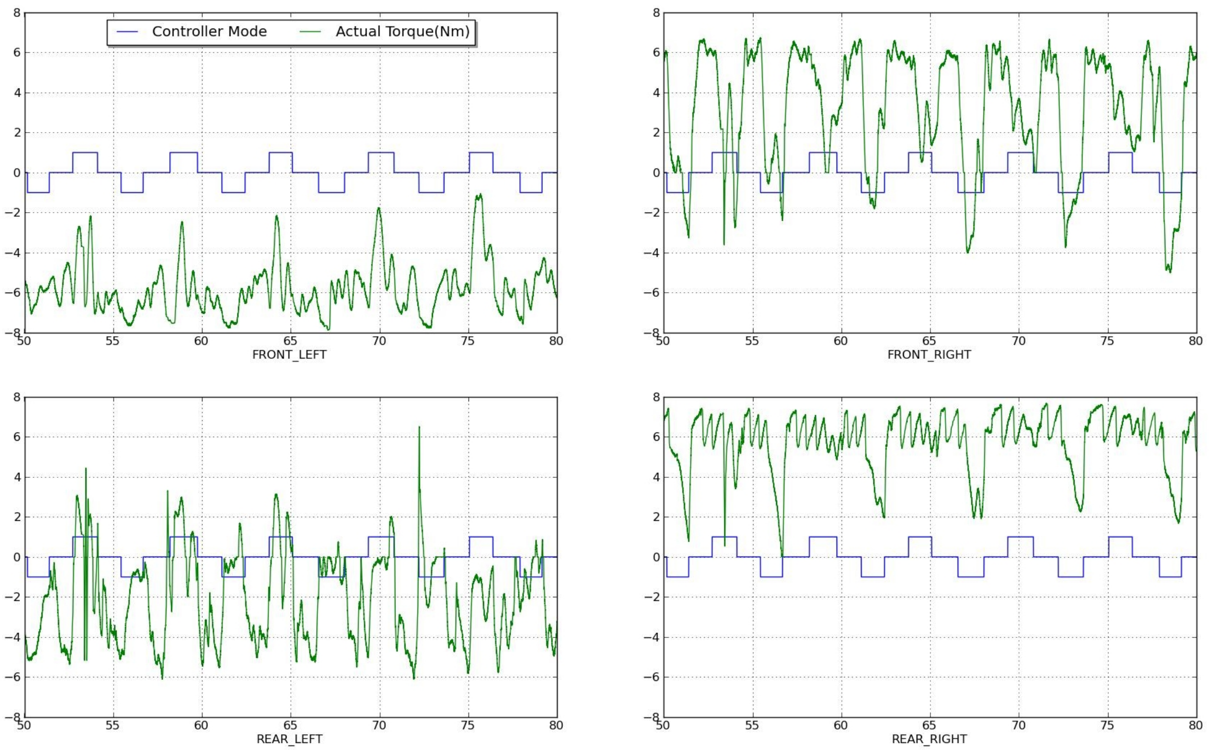


Abbildung 44: *Raddrehmomente bei Nutzung des neuen Reglers für Drehungen des Roboters*

WP3400: Rad-/Fußoptimierung

Ein neues Design für eine elastische Radkonstruktion sowie die zur Herstellung nötige Gussform wurden erstellt. Die Polyurethan-Bauteile der neuen Struktur wurden im Tiefzieh-Verfahren gefertigt (siehe Abb. 45). Anschließend wurde die Biegung der Radkonstruktion unter verschiedenen Lasten getestet (siehe Abb. 46).

Weiterhin wurden Experimente mit verschiedenen Hall-Sensoren und Magneten an den Rädern des Asguard V3 durchgeführt. Dabei bestand das Ziel darin, eine einfache, verlässliche und genaue Kontaktbestimmung und Abschätzung der wirkenden Kräfte an den Füßen zu ermöglichen.

Das Ziel dieses Arbeitspakets war es, die Vibrationen des Systems zu reduzieren und das Laufverhalten grundsätzlich zu verbessern. Dies wird durch die Modifikation des Sternrad-Designs erreicht. Federelemente in radialer Richtung und verbesserte Dämpfung an der Fußstruktur sind prototypisch getestet worden.

Ein erster Prototyp mit verschiedenen elastischen Elementen ist auf dem AsguardV2-System getestet worden.

Das beim Asguardv3 verwendete Sternrad wurde um die Fähigkeit zur Detektion des Bodenkontakts erweitert. Dazu wurden die Füße mit Sensorik ausgestattet und auf der Radnabe eine Elektronik angebracht, die die Sensorik ausliest und die Daten über eine Funkverbindung an den Roboter überträgt.

Zum Testen des Konzepts wurde zunächst nur ein Rad mit Sensorik ausgestattet. Damit das Sensorrad oder "Smart Wheel" auf dem Roboter getestet werden konnte, wurden drei weitere Räder mit den gleichen Dimensionen wie beim "Smart Wheel" erstellt. Erste Tests mit dem "Smart Wheel" wurden erfolgreich durchgeführt.

Es wurde ein Herstellungskonzept für die Integration von Sensoren in die Asguard-Füße erstellt und getestet. Eine Illustration der einzelnen Schritte des Prozesses ist in Abb. 50 dargestellt. Dabei wurden zur Evaluation des Prozesses drucksensitive Widerstände auf eine im Rapid Prototyping-Verfahren hergestellte Struktur aufgebracht. Die für die Sensoren benötigte Elektronik wird ebenfalls in der Struktur untergebracht und durch gehärtetes Polyurethan (Protocast 134 - Shore 80D) vor Umwelteinflüssen geschützt. In einem weiteren Schritt wurde die innere Struktur in einer den Asguard-Füßen angepassten Gussform ausgelegt und mit einem weicheren Polyurethan (Protoflex Shore 55A) umgeben. Die äußere Struktur dient als Schutz der Sensoren und bietet die für den Fuß nötigen Materialeigenschaften wie Flexibilität und Traktion. Innerhalb des Prozesses können die Materialeigenschaften noch durch Mischen der beiden Materialien den Ansprüchen der Anwendung angepasst werden. Zur Erzeugung der Fußstrukturen wird die Vakuumgusskammer eingesetzt.

Um den Bodenkontakt einzelner Füße zu detektieren wurden Sensorfüße mit jeweils drei Folienkraftsensoren (FSR – force sensing resistor) aufgebaut und getestet. Die innere Kontur besteht aus einem Rapid-Prototyping-Teil, an dem die Sensoren und die Elektronik befestigt werden. Eine LED an jedem Fuß zeigt den Bodenkontakt an. Zum Schutz vor Feuchtigkeit und mechanischer Belastung wurde die Elektronik mit Epoxidharz vergossen. (Abb. 51)

Im zweiten Schritt wurde der starre Innenteil in der Vakuumgussanlage mit einer weichen Außenschicht aus Polyurethan vergossen. Die äußere Schicht erhöht die Traktion der Füße und schützt die Foliensensoren vor Beschädigungen. (Abb. 52)

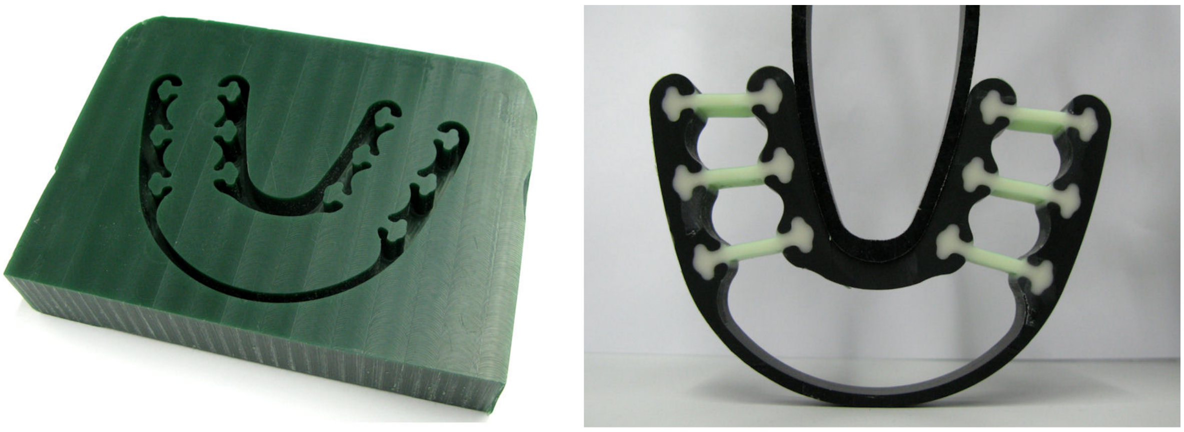


***Abbildung 45:** Gussform (l) und fertiges Bauteil (r) für die neue Fußstruktur.*

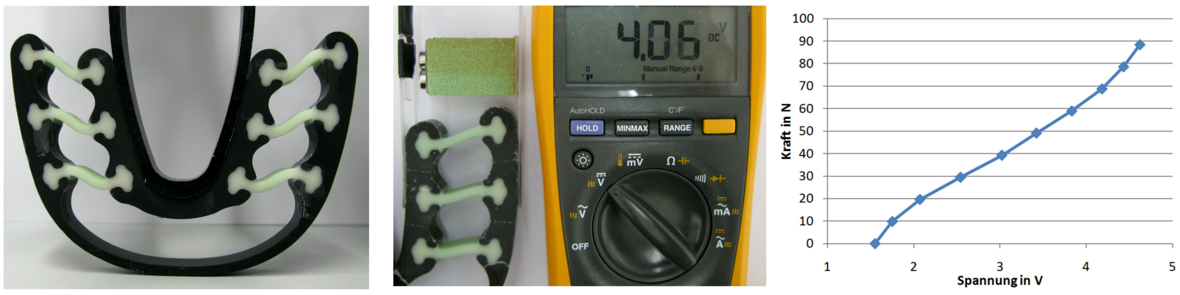


***Abbildung 46:** Versuchsaufbau und beispielhafte Ergebnisse für Belastungstests am neuen Fuß*

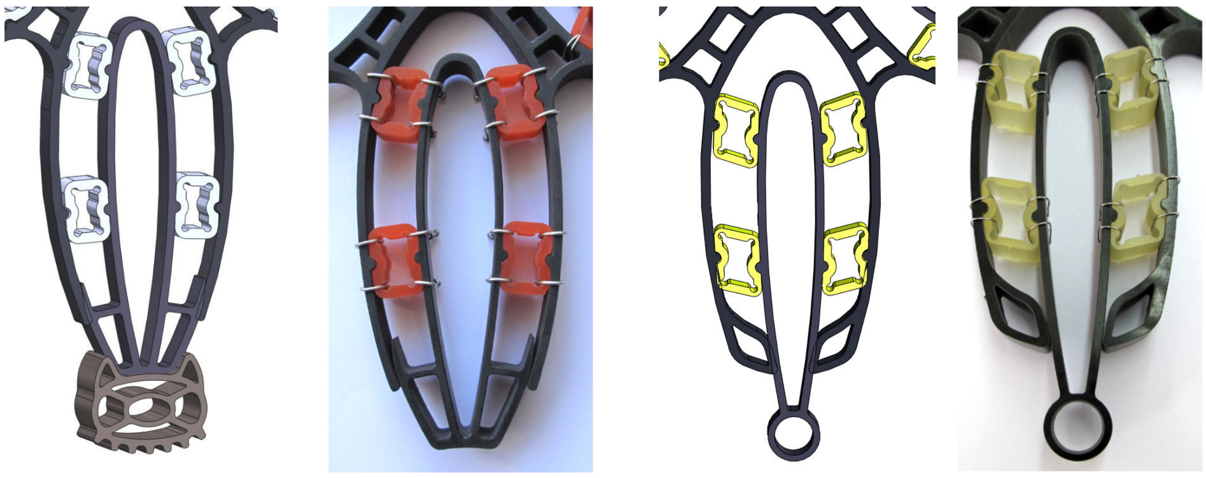


***Abbildung 47:** Konzepte zur Integration von Federelementen in das Asguard-Rad*

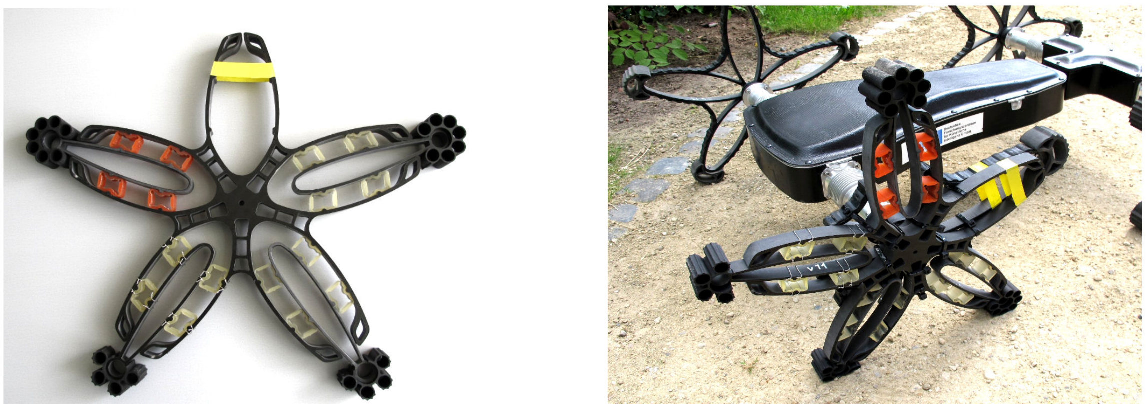


Abbildung 48: *Modifiziertes Asguard-Rad zum Testen von Federelementen*

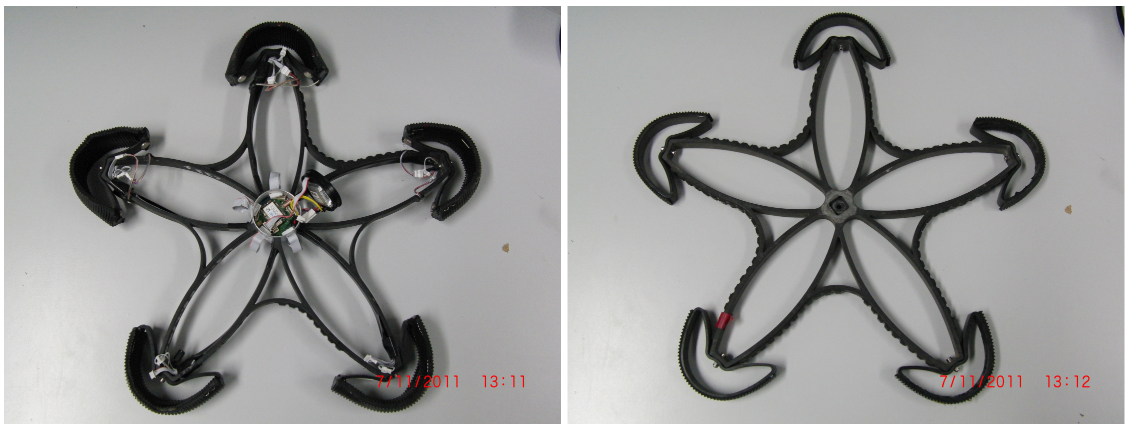


Abbildung 49: *Links das "Smart Wheel", rechts ein Rad mit den gleichen Dimensionen wie beim "Smart Wheel", jedoch ohne Sensorik*

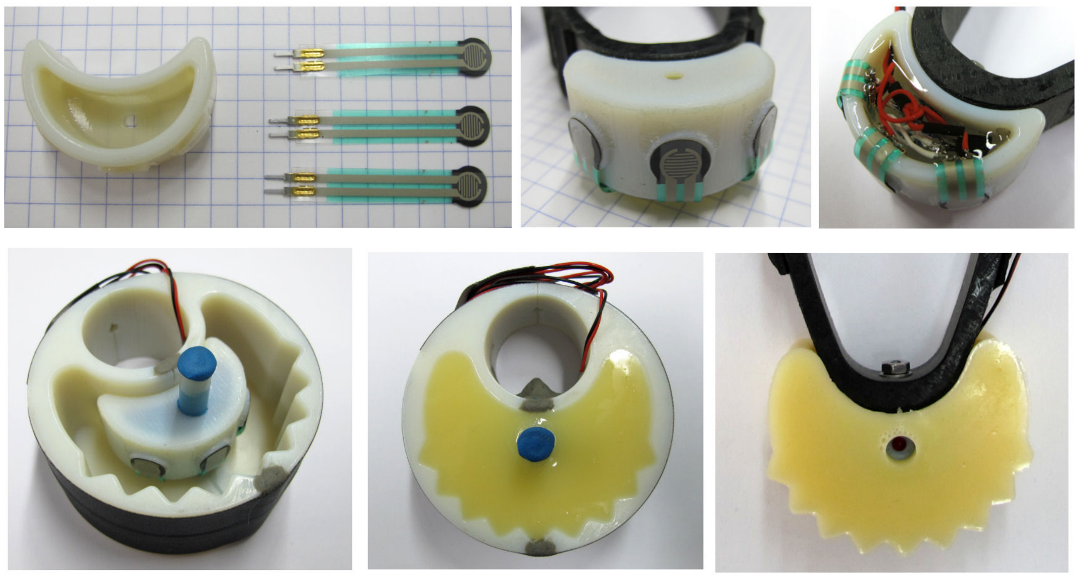


Abbildung 50: *Es wurde ein Prozess zur Einbettung von Sensoren in die Fußstruktur des AsguardV4-Systems entwickelt. Die Bilder zeigen die in Abschnitt 3.3.0.10 erläuterten Schritte, in denen unterschiedliche Materialien im Schichtverfahren zur finalen Struktur zusammengesetzt werden.*

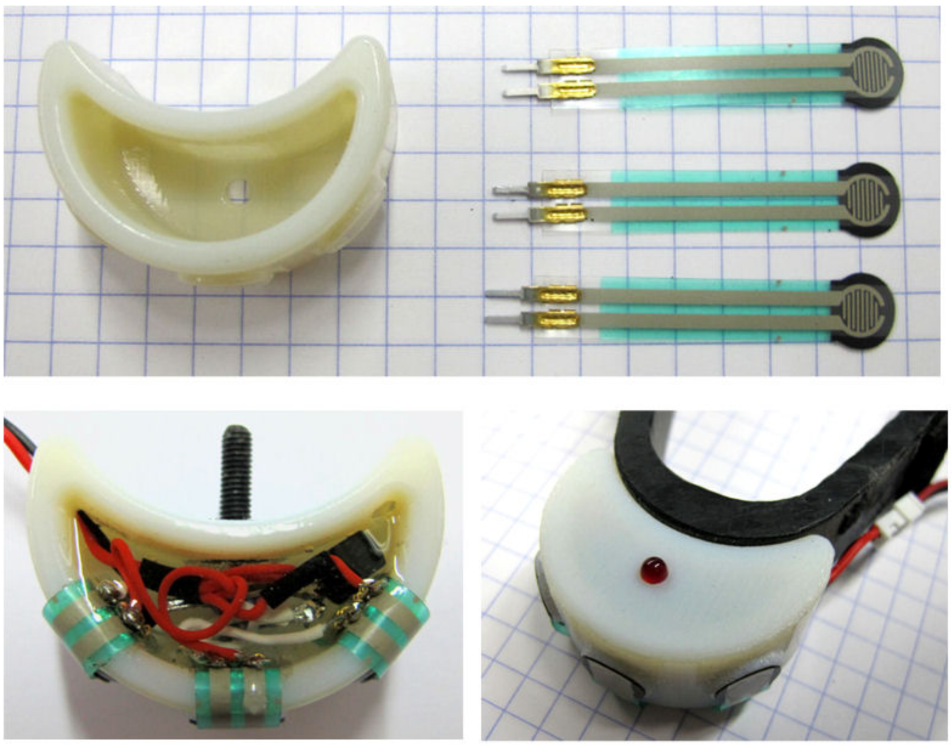


Abbildung 51: *Innenteil des Sensorfußes mit Elektronik und FSR-Foliensensoren*

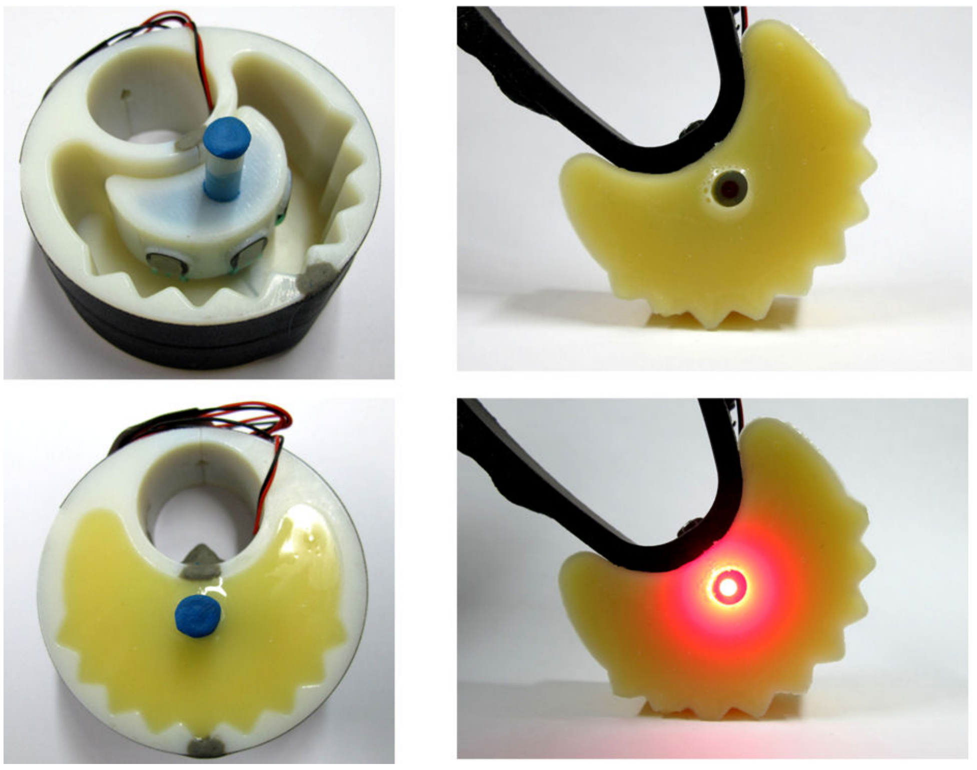


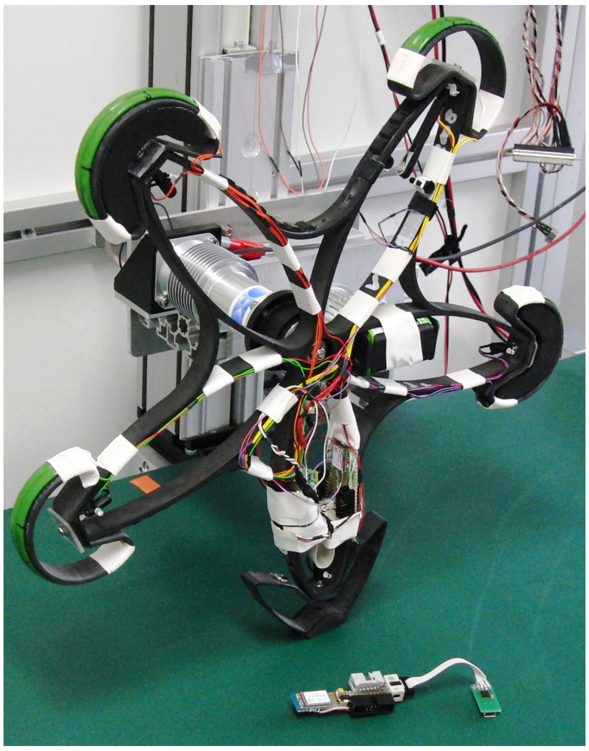
Abbildung 52: *Gussform und Sensorfuß mit Polyurethan Außenschicht*

Intelligente Radsensorik Mit Hilfe der Sensordaten aus der im Projekt entwickelten intelligenten Rad-Sensorik soll die Verbesserung des Fahrverhaltens des Roboters ermöglicht werden. Genaue Zustandsinformationen des Rades können bei der Verwirklichung einer besseren Fahrtregelung sowie Anti-Schlupf-Regelung für ASGuard hilfreich sein. Die Informationen aus der Rad-Sensorik sind gemeinsam mit anderen Sensorinformationen des Roboters unter anderem zur Bodenerkennung verwendbar. Der eingebettete Aufbau der Sensorik erlaubt eine bessere Messposition und Schutz gegen unerwünschte Störungen. Die Radsensorik nimmt die Informationen zur Erkennung der Bodenkontaktierung mit Hilfe der Fußsensoren und die Informationen zu den auf das Rad einwirkenden Kräften mit Hilfe der Kraftsensoren an den Beinen wahr (s. Abb. 53). Diese Sensordaten werden von der Sensorik des Rades direkt drahtlos an den Roboter übertragen.

Es wurden zunächst unterschiedliche Sensortypen getestet. Zu diesem Zweck wurden fünf sternförmige Testräder verwendet, um jeweils eine andere Sensorart einzubetten. Nach den Testläufen wurden für einfache Kontakterkennung der Bandschalter und für die Kraftmessung der Hallsensor gewählt. Als Kommunikationsmethode wurde ein Zigbit-Modul gewählt. Nach der Fertigstellung des Testaufbaus konnte gezeigt werden, dass das Modul für diese Anwendung tatsächlich geeignet ist. Das für die Radsensorik benötige elektronische Design wurde wie folgt verwirklicht: Beim Design der elektronischen Schaltung wurde ein zusätzlicher STM32-Mikrokontroller eingeplant, der für die Aufnahme der Sensorwerte spezialisiert ist und über die serielle Schnittstelle mit dem Zigbit-Modul kommuniziert. Für die Stromversorgung wurde ein Lithium-Polymer-Akku mit Step-Down-Regler in das Design einbezogen. Die zusätzlichen Schutzschaltungen für Batterieüberwachung, zu hohe Spannung und Kurzschluss wurden integriert. Nach dem Design der Schaltung wurde die Platine gefertigt und getestet. Dabei wurde festgestellt, dass die Akkuüberwachung einige Probleme verursacht und der Step-Down-Regler einen schlechteren Wirkungsgrad als erwartet hat. Nach der Programmimplementierung auf dem ASGuard-Framework für STM32 wurde die Radsensorik auf unterschiedliche Arten getestet. Zunächst wurde die Fußsensorik über ihre gesamte Kennlinie in der horizontalen Ebene getestet. So wurden Werte ermittelt, die zeigen, dass ein Rückschluss von den Spannungswerten der Sensoren auf die einwirkenden Kräfte möglich ist. Anschließend wurde die Radsensorik auf dem Laufband getestet (Abb. 54). Die hierbei ermittelten Ergebnisse zeigen, dass der Bandschalter eine robuste Bodendetektion bietet, dass die Hallsensoren mit einem gewissen systematischen Fehler Informationen über die einwirkende Kraft liefern und dass diese Daten über die drahtlose Verbindung problemlos zur zentralen Rechnereinheit weitergeleitet werden.

Die Sensorik wurde nach statistischen Verfahren bewertet. Dabei wurde beobachtet, dass die Messwerte leicht verrauscht sind. Schließlich wurden diese Testergebnisse in der Weka-Umgebung als Trainings-Datensatz verarbeitet. Der als Ergebnis entstandene Klassifikator zur Krafterkennung war für den entscheidungsbaumbasierten Algorithmus zu mehr als 99% und für den Algorithmus auf Basis einen neuronalen Netzes zu mehr als 91% zutreffend.

Nach diesem Schritt wurde der Klassifikator für die Messdaten eines Laufbandversuchs angewendet. Der entscheidungsbaumbasierte Klassifikator hatte das Messergebnis so klassifiziert, dass der Fußtritt in den meisten Fällen so erkannt wurde, dass sich der Kraftvektor analog zur Bewegung des Fußes bewegt und von einem Eckpunkt des Fußes zum anderen verläuft. Die Zustandsinformationen des Rades können bei der Verwirklichung einer besseren Fahrtregelung sowie einer Anti-Schlupf-Regelung für ASGuard genutzt werden.



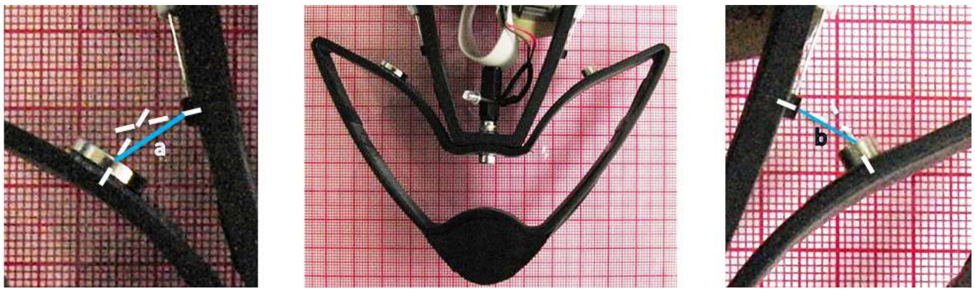
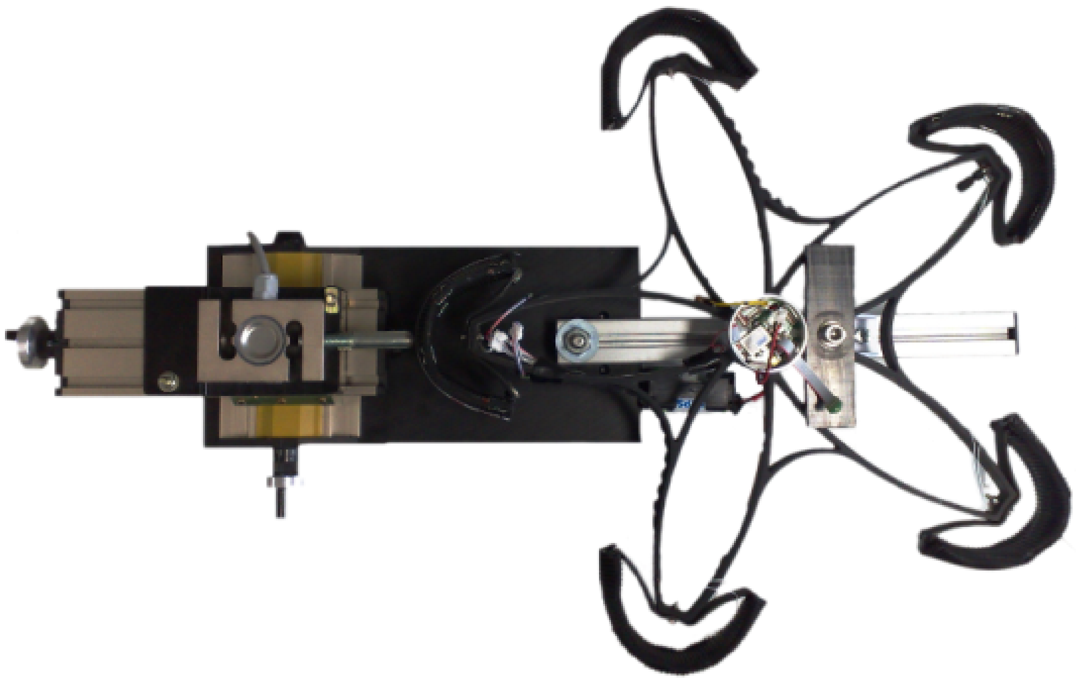


Abbildung 53: *Oben: Intelligentes Sensorrad. Unten: Detail des Kraftmess-Fußes.*



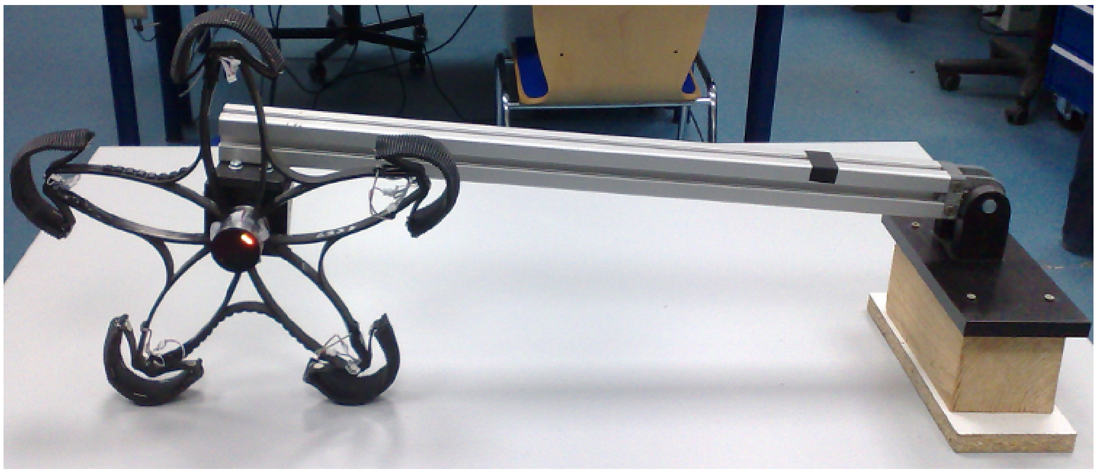
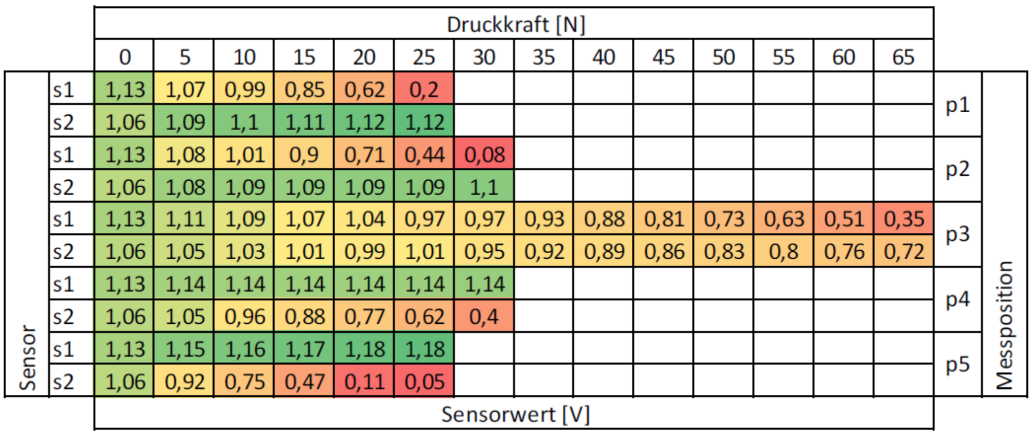


Abbildung 54: *Oben: Test-Setup für einzelne Fußsensorik. Unten: Test-Setup für Laufbandversuche mit dem vollstädigen Rad.*



Sensor		Druckkraft [N] 0	5	10	15	20	25	30	35	40	45	50	55	60	65	Messposition
	s1	1,13	1,07	0,99	0,85	0,62	0,2									p1
	s2	1,06	1,09	1,1	1,11	1,12	1,12									p1
	s1	1,13	1,08	1,01	0,9	0,71	0,44	0,08								p2
	s2	1,06	1,08	1,09	1,09	1,09	1,09	1,1								p2
	s1	1,13	1,11	1,09	1,07	1,04	0,97	0,97	0,93	0,88	0,81	0,73	0,63	0,51	0,35	p3
	s2	1,06	1,05	1,03	1,01	0,99	1,01	0,95	0,92	0,89	0,86	0,83	0,8	0,76	0,72	p3
	s1	1,13	1,14	1,14	1,14	1,14	1,14	1,14								p4
	s2	1,06	1,05	0,96	0,88	0,77	0,62	0,4								p4
	s1	1,13	1,15	1,16	1,17	1,18	1,18									p5
	s2	1,06	0,92	0,75	0,47	0,11	0,05									p5

Sensorwert [V]

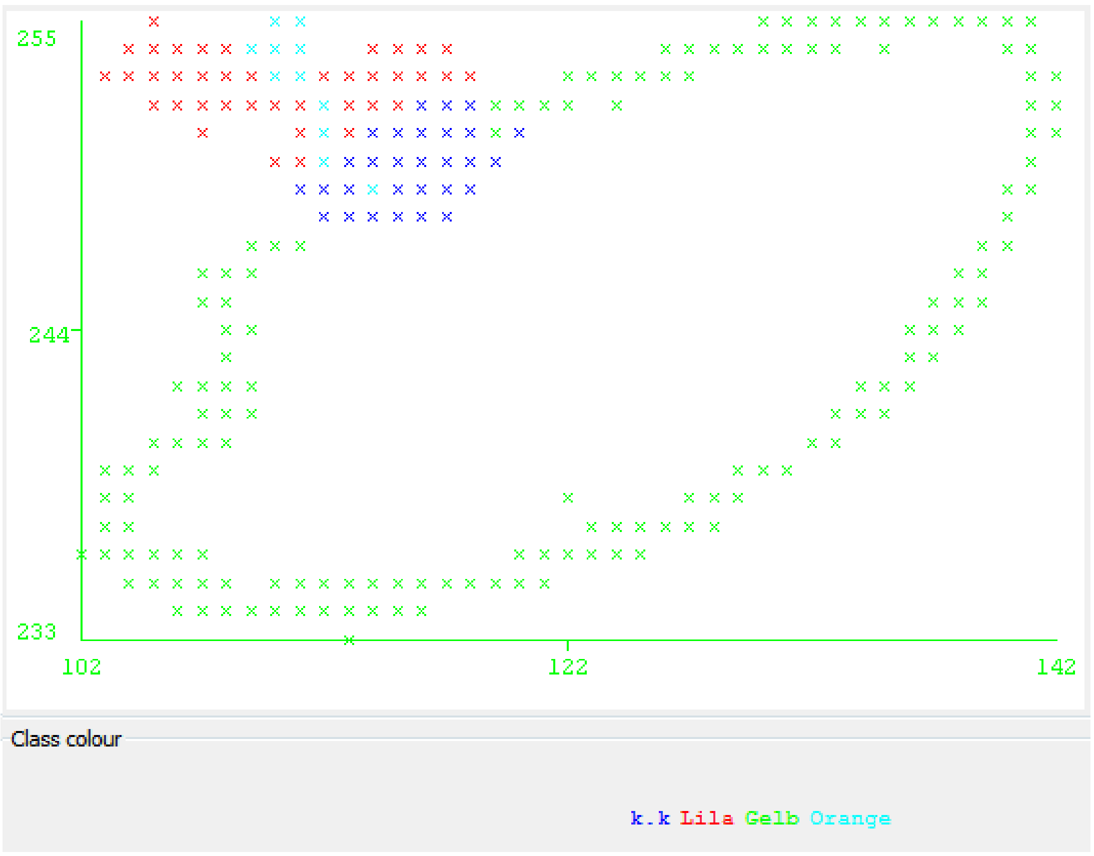


Abbildung 55: *Oben: Sensorspannungen bei unterschiedlichen Kräften und Kraftangriffspunkten. Unten: Hysteresekurve, die beim Schreitvorgang eines Fußes durchlaufen wird.*

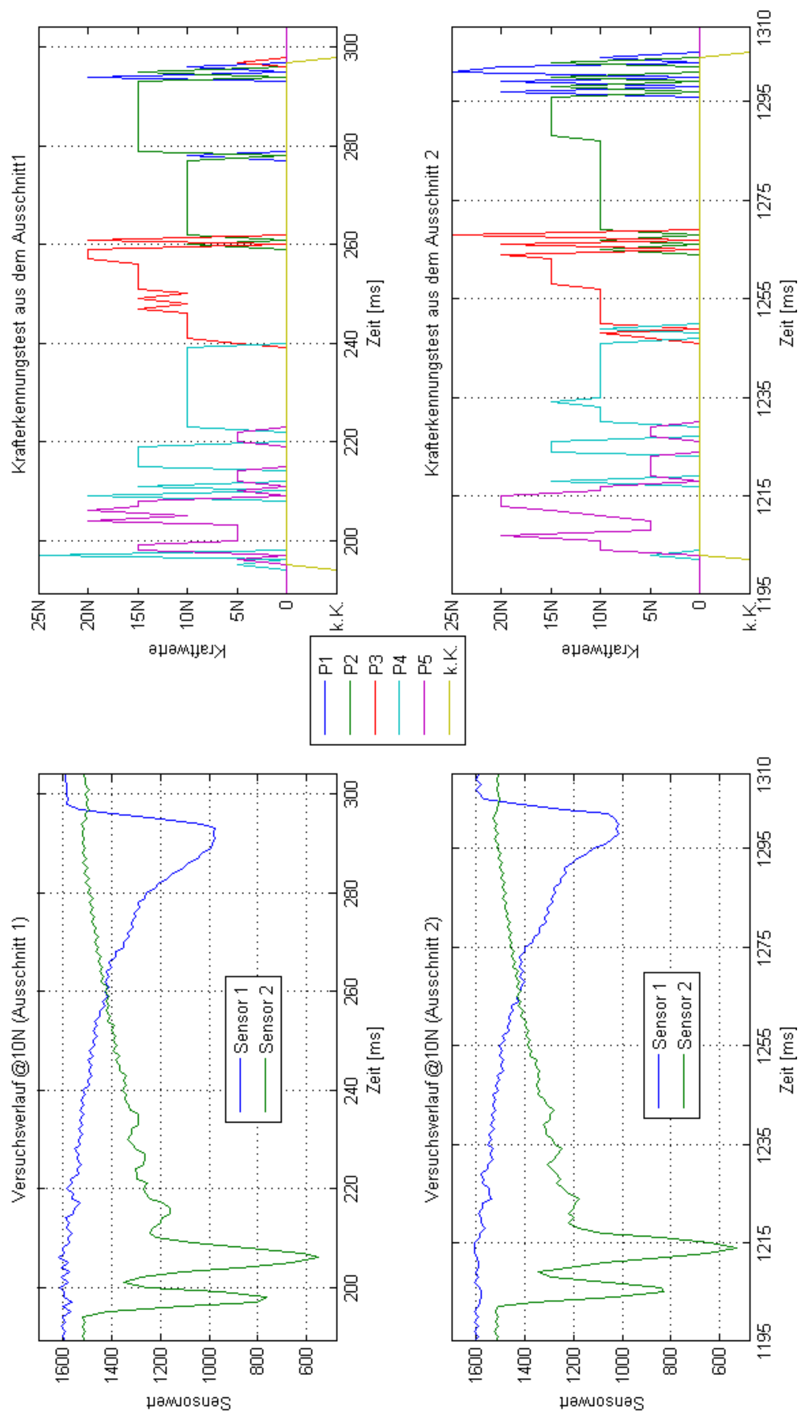


Abbildung 56: *Sensormesswerte während einen Schreitvorgangs.*

WP3500: Integration Verifikationssystem

Zunächst wurden die Anforderungen an das Verifikationssystem definiert. Auf Grundlage der Anforderungsliste wurden die Hauptkomponenten für das Verifikationssystem ausgewählt. Anschließend wurde ein mechanisches Konzept erstellt (siehe Konstruktionszeichnung 63). Dieses Konzept sieht vor, dass der Roboter in mehrere funktionale Module unterteilt wird: Ein Trägermodul, ein PC-Modul, ein Sensormodul, ein DGPS-Modul sowie zwei Batteriemodule. Jede dieser Komponenten wird mechanisch sowie elektrisch gekapselt und wasserdicht sein.

Für jedes der Module wurde spezifiziert, welche elektronischen Komponenten verbaut und wie diese verschaltet werden. Des Weiteren wurden die elektrischen Schnittstellen zwischen den Modulen definiert.

Um die benötigte Motorleistung für das Verifikationssystem abschätzen zu können, wurden mit dem bestehendem AsguardV3-System Tests durchgeführt. Dabei wurde die Stromaufnahme für die Szenarien *Schnelle Fortbewegung*, *Treppensteigen* sowie *Punktdrehung* bestimmt. Die Tests haben gezeigt, dass die Motoren des AsguardV3 nahe und teilweise über dem Maximum ihrer spezifizierten Leistung arbeiten. Da das Verifikationssystem auf Grund der Kapselung schwerer sein wird als das V3-System, sind neue Motormodule ausgewählt und getestet worden (siehe Abb. 58). Um die Motoren vor thermischen Belastungen zu schützen werden im neuen Design Temperatursensoren aufgebracht. Hierzu sind Tests durchgeführt worden, um die optimale Position der Sensoren festzustellen (siehe Abb. 59). Das Ergebnis der Untersuchungen zeigt, dass ein Sensor an der Rückseite des Motors genügt, da die zeitliche Verzögerung der Temperaturmessung nur gering ist und nur unwesentlich von der Maximaltemperatur in der Mitte des Gehäuses abweicht.

Um die leistungsfähigeren Motoren des V4-Systems zu steuern, ist eine neue Elektronik entwickelt, integriert und getestet worden (siehe Abb. 60). Dazu wurde die Firmware, die derzeit auf der Motorsteuerungsplatine (H-Bridge-Module) verwendet wird, auf die neue Hardware portiert. Erste Ergebnisse zeigen inbesondere ein reduziertes Rauschen bei der Strommessung.

Die Stereokamera auf dem aktuellen AsguardV3-System hat eine Baseline von 25 cm (Abstand zwischen den Kameras). Dies hat in der Vergangenheit zu Problemen gerade im Nahbereich geführt. Die große Baseline hat hier zu starken perspektivischen Verzerrungen geführt, so dass der Stereoalgorithmus negativ beeinflusst wurde. Um eine optimale Baseline für das V4-System zu ermitteln, sind systematische Tests der Baseline durchgeführt worden (siehe Abb. 61). Als Ergebnis der Tests wird jetzt eine Baseline von 15 cm bis 20 cm favorisiert.

Die Motormodule für AsguardV4 wurden gefertigt, integriert und getestet (siehe Abb. 62). Die neuen Motormodule bestehen aus einer äußeren Hülle, die über den Motor, das Getriebe, eine elastische Wellenkupplung sowie eine Encoder geschoben wird. Um die kompakte Bauform realisieren zu können, wurden der Motor und das Getriebe leicht angefräst, um Raum für ein Flachbandkabel vom vorderen Encoder zur Hinterseite des Motors zu schaffen. Für den vorderen Encoder sowie für die Hinterseite des Motors wurden Platinen gefertigt, um den Encoder über das Flachbandkabel anzubinden und alle Kabel einheitlich vom Motor wegzuführen.

Eine neue Zentralplatine für die Motorelektronik wurde gefertigt und integriert. Des Weiteren wurde die gesamte Elektronik der Basisplattform des AsguardV4 auf einer Konstruktionszeichnung des AsguardV4 aufgebaut und in Betrieb genommen. Hierdurch soll einerseits die Elektronik getestet werden, bevor die mechanische Plattform gefertigt wird. Anderseits soll geprüft werden, ob es Platzprobleme bei der Elektronik gibt, die noch vor der Fertigung der mechanischen Plattform behoben werden müssen.

Konzept und Design des AsguardV4 wurden weiter ausgearbeitet und verfeinert (siehe Abb. 63). Dazu wurden alle Elektronikkomponenten modelliert und ihre Position sowie ihre Anschlüsse innerhalb des Roboters definiert.

AsguardV4 wird aus drei Hauptmodulen bestehen: Mobile Plattform, PC-Modul und Batteriemodul. Der am PC-Modul angebrachte Sensorkopf soll per Hand auf vordefinierte Einrastpositionen verstellbar sein und über einen Motor verfügen, der den oberen Teil neigen kann. Der obere Teil des Sensorkopfes soll mit einem Laserscanner, einer Stereokamera sowie LED-Scheinwerfern ausgestattet werden. Der gesamte Roboter soll wasser- und staubdicht sein.

Das Verifikationssystem (Asguard v4) ist ein modulares Konzept, bestehend aus einer mobilen Basisplattform, wasserdichten Akkumodulen sowie einem Aufbau, der alle PC- und Sensorkomponenten enthält. Die Basisplattform ist mechanisch und elektronisch vollständig entwickelt, integriert und getestet. (Abbildungen 64, 65, 66 und 67)

Die PC- und Sensorkomponenten sind vollständig entwickelt und getestet (Abbildungen 68 und 69).

Abbildung 70 zeigt das fertig integrierte Asguard-v4-System. Die Sensorausstattung ist mit der des Asguard v3 identisch. Das System hat ein Gesamtgewicht von 16 kg, und ist vor Staub und Spritzwasser geschützt. Zusätzlich ist der v4 mit einem Beleuchtungssystem ausgestattet, das auch Kameraaufnahmen bei schlechten Lichtverhältnissen ermöglicht.

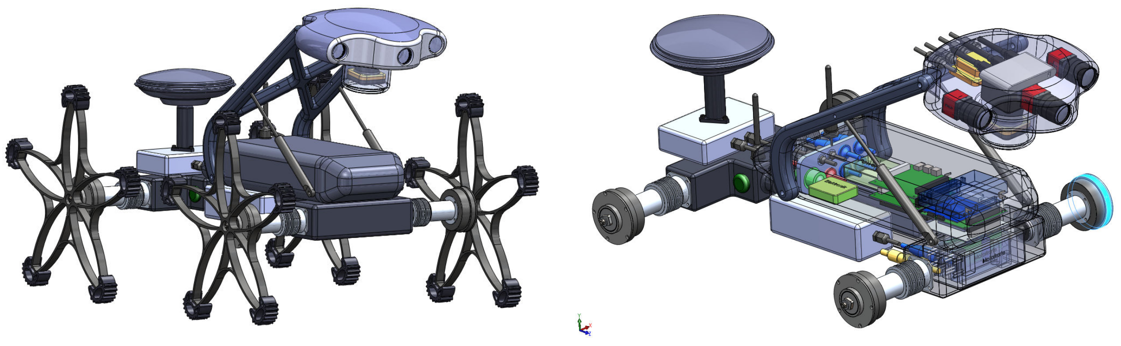


Abbildung 57: *Vorläufige Konstruktionszeichnungen für die vierte Generation des Asguard-Roboters*

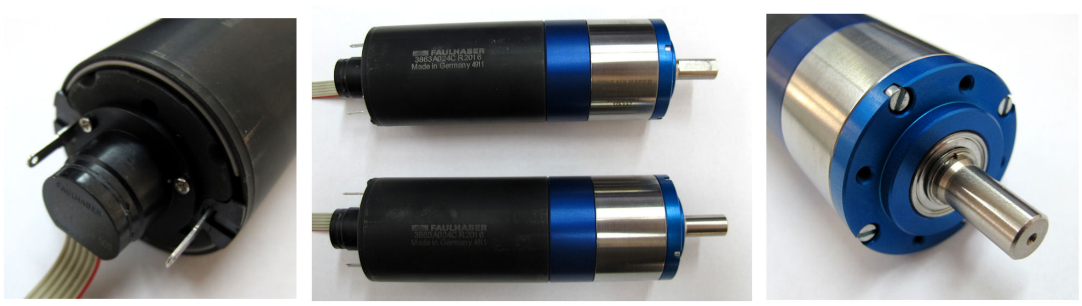


Abbildung 58: *Die für das AsguardV4-System ausgewählten Motoren haben eine höhere Leistung und arbeiten in einem besser geeigneten Temperaturbereich als die AsguardV3-Motoren.*

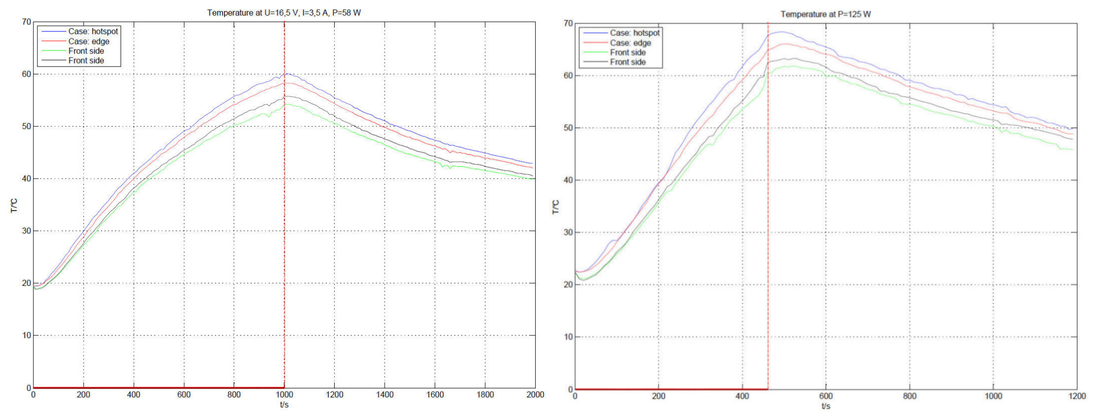


Abbildung 59: *Messung der Temperaturentwicklung an verschiedenen Messpunkten des Motors.*

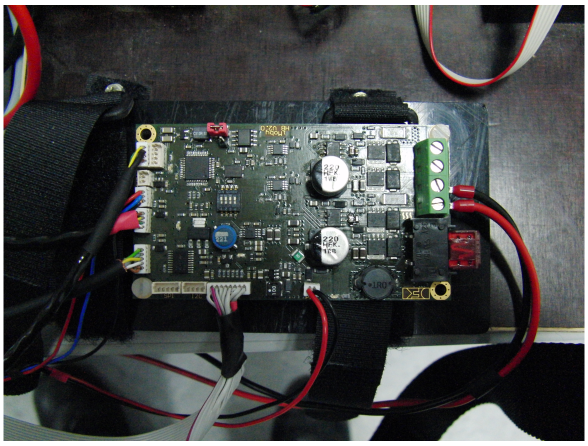


Abbildung 60: *Die neu entwickelte Steuerungselektronik für die leistungsfähigeren V4-Motoren.*

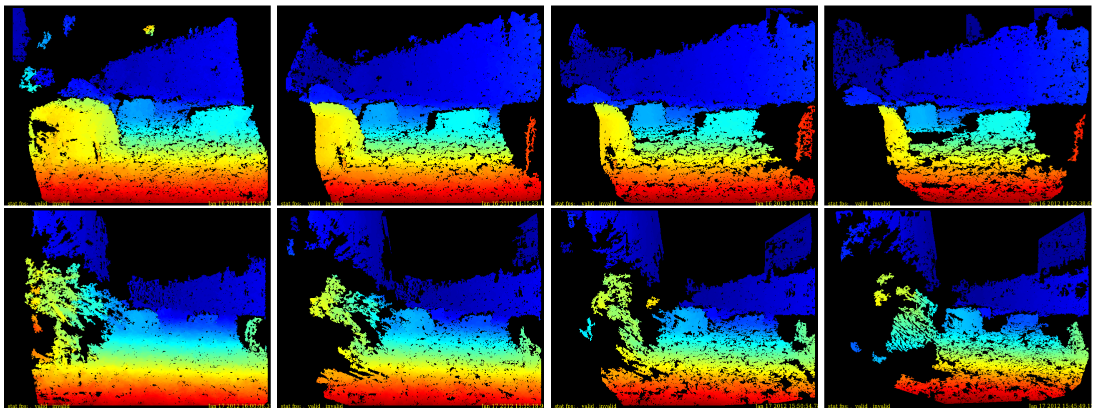


Abbildung 61: *Tiefenbild für Standardobjektiv (obere Reihe) und Weitwinkelobjektiv (untere Reihe), bei Basisabständen von 10, 15, 20 und 25 cm (von links nach rechts)*

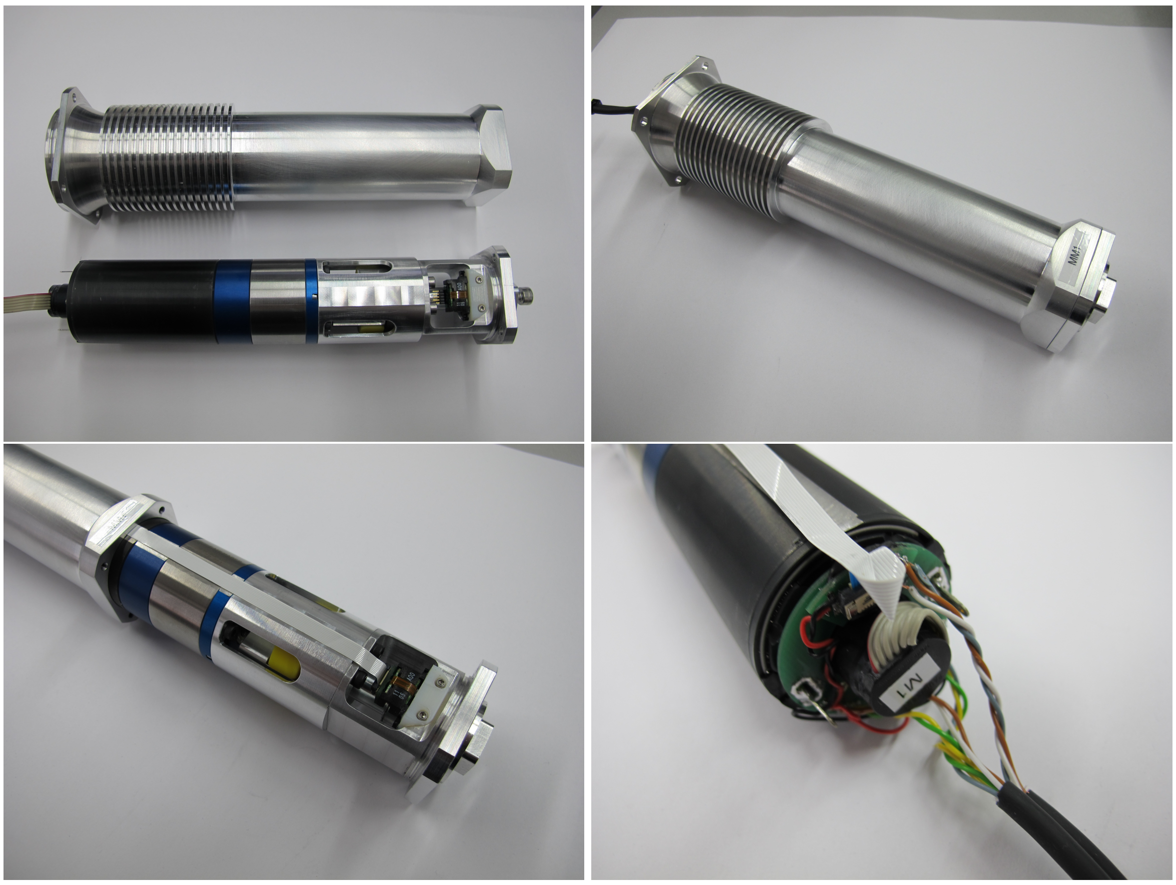


Abbildung 62: Motormodule des AsguardV4

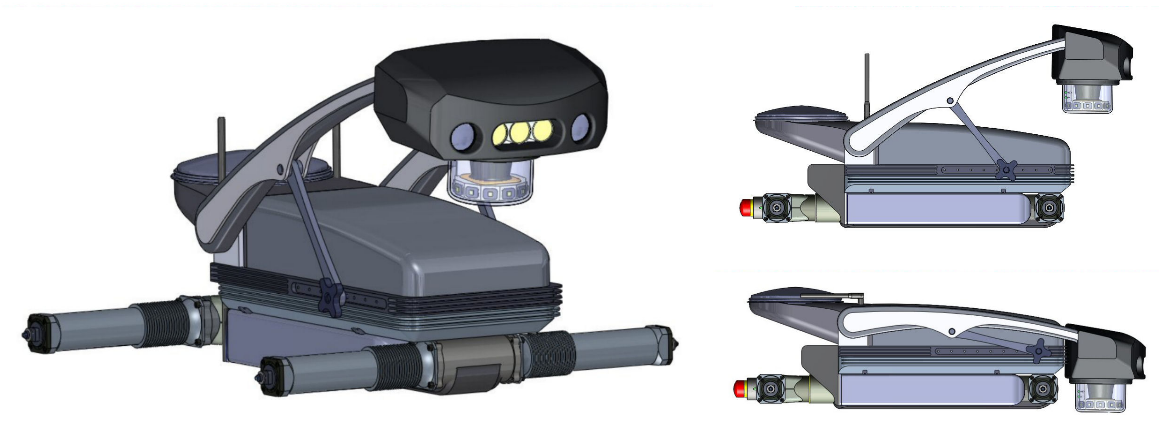


Abbildung 63: Vorläufige Konstruktionszeichnungen für die vierte Generation des Asguard-Roboters

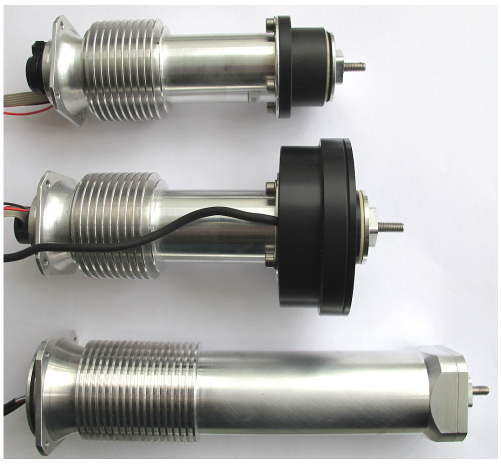


Abbildung 64: *Motormodule Asguard v2, v3 und v4*

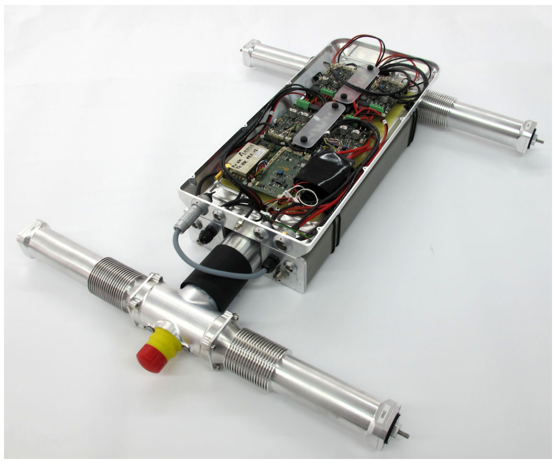


Abbildung 65: *Mobile Basisplattform Asguard v4*

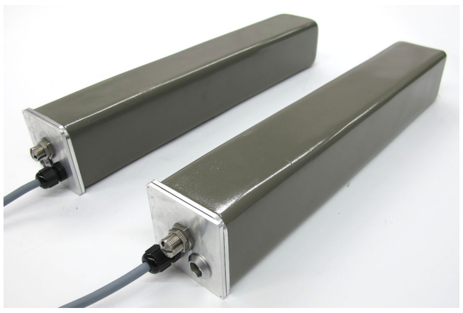


Abbildung 66: *Akkurohre*

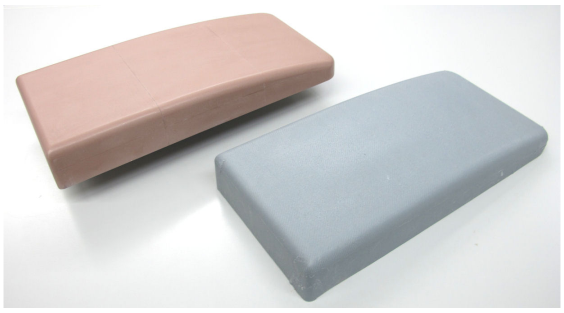


Abbildung 67: *Fiberglasabdeckung und Form*

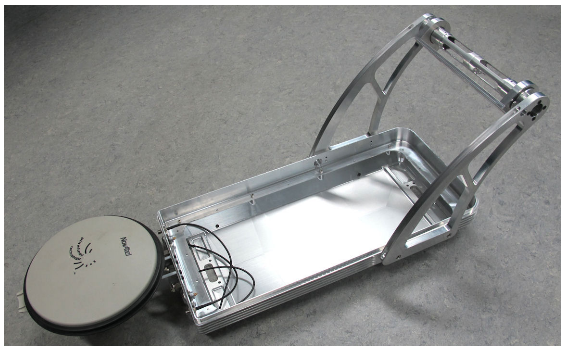


Abbildung 68: *Mechanische Komponenten PC-Modul*

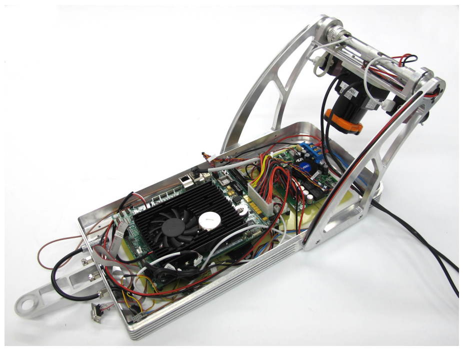


Abbildung 69: *PC-Modul Asguard v4*

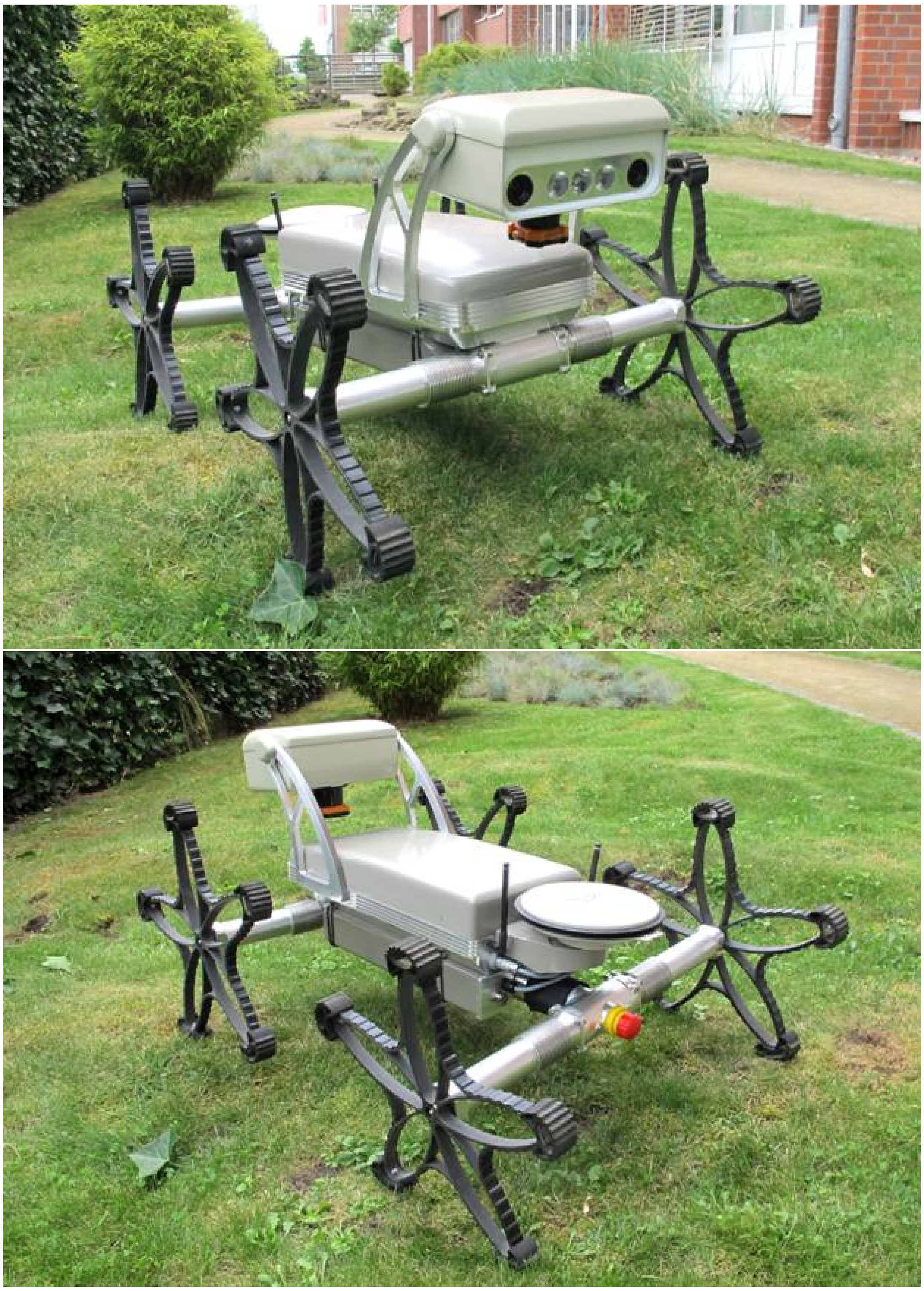


Abbildung 70: *Das fertig integrierte Asguard-v4-System hat ein Gesamtgewicht von 16 kg.*

3.4 AP4000: Embodied SLAM

In diesem Arbeitspaket soll ein hybrider Ansatz aus embodied SLAM und visuellem SLAM entwickelt werden. Der „embodied"-Teil dieses Ansatzes stellt dabei eine dramatische Verbesserung gegenüber dem reinen visuellen SLAM-Ansatz dar. Wir werden daher in diesem Arbeitspaket eine anspruchsvolle Umgebungsrepräsentation erzeugen (AP4100), die den eSLAM-Algorithmus unterstützt. Die beiden Aspekte des eSLAM-Ansatzes werden dann implementiert und untersucht: Der Einsatz propriozeptiver Daten als direktes Mittel zur Lokalisation (WP4200) sowie der Einsatz einer propriozeptiven Geländeklassifikation als Mittel zur Selbstlokalisation und zur Kartierung (AP4300). Beide Arbeitspakete müssen in engem Zusammenhang mit den Arbeiten zur Mobilität (AP3000) durchgeführt werden, da die Mittel zur Interpretation von propriozeptiver Information von AP3000 bereitgestellt werden. AP4400 strebt schließlich die Vereinigung von visuellen SLAM-Methoden mit dem im vorangegangenen AP entwickelten Ansatz für embodied SLAM an.

WP4100: Repräsentation der Umgebung

Das Ergebnis dieses Arbeitspakets bildet die Grundlage für verschiedene andere Projekte, weshalb es in der Arbeitsgruppe „Navigation und Planung" weiter vorangetrieben wird. Ziel ist dabei, von möglichen Synergieeffekten mit anderen Projekten profitieren zu können.

Zudem soll die Integration in andere Projekte vorangetrieben werden, die dann ihre relevanten Repräsentationen und Algorithmen für ihre Anwendungsfälle integrieren können.

Das in Kooperation mit der Arbeitsgruppe „Navigation und Planung" entwickelte Modell einer **Umgebungsrepräsentation** (Environment Representation; ER) wurde prototypisch implementiert. Die aktuelle Realisierung unterstützt den Umgang mit **multimodalen Kartentypen**. Ziel ist es, die ER-Realisierung als generisches Modul anzubieten, das für verschiedene Problemstellungen einsatzfähig ist. Ebenfalls prototypisch implementiert wurde auf Basis des ER-Frameworks eine Anwendung zu Darstellung von Laserscans (Abbildung 71).

Eine geeignete Umgebungsrepräsentation wird unter anderem für die **Pfad- und Bewegungsplanung** (Path/Motion Planning) sowie für **Lokalisierung und Kartenbildung** (Localisation and Mapping) benötigt. Derartige Strukturen zur Verwaltung von Sensordaten sind grundlegend für mobile Systeme, um autonom agieren zu können (Weltmodell). Ein generischer ER-Ansatz liefert hierbei nicht nur einen Beitrag zur Wiederverwendbarkeit, sondern auch zur generellen Verwendbarkeit einer solchen Datenstruktur.

Um die Umgebung visualisieren zu können, wurde ein Softwaretool implementiert, das Daten aus der entwickelten Umgebungrepräsentations-Bibliothek *envire* darstellt: *enview*. Es ist in der Lage, folgende Daten aus *envire* zu visualisieren:

- Die Position des Roboters und die Ungenauigkeit der Positionsschätzung

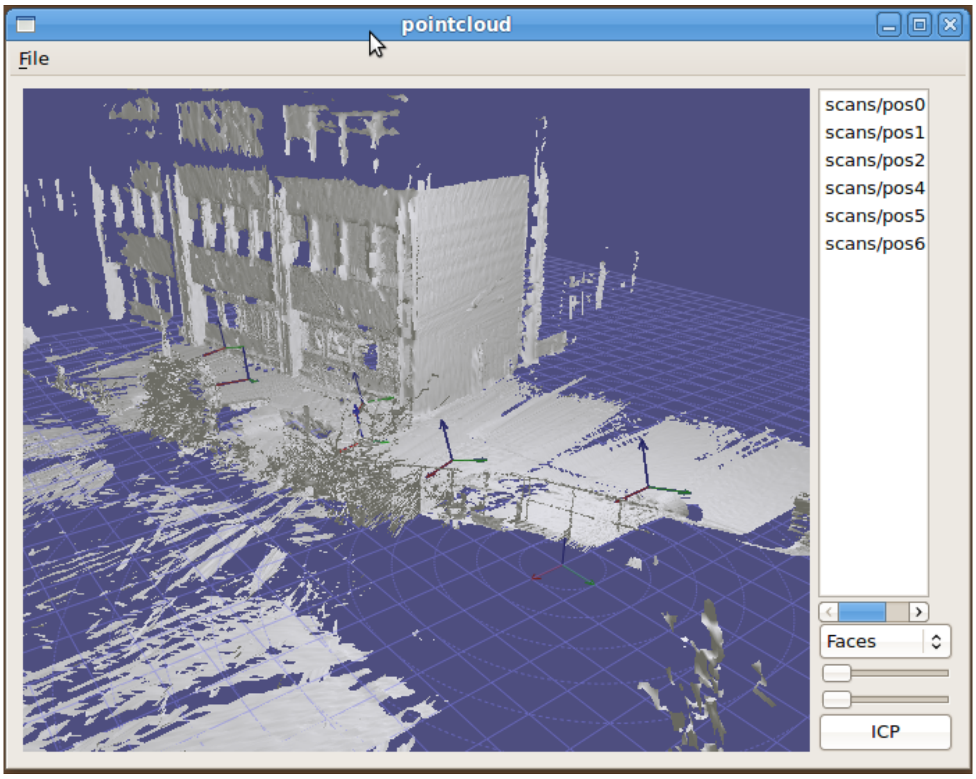


Abbildung 71: *Point Cloud Viewer*

- Nächster Wegpunkt des Roboters
- Einzelne Scans aus der Umgebungserfassung
- Der aktuell bearbeitete Befehl
- Die gefahrene Strecke (Abb. 34)

Es wurde ein Posenschätzer entwickelt, der GPS, Odometriedaten und IMU-Messwerte über einen Kalman-Filter zu einer Pose fusioniert.

Ein generisches Kamerainterface für FireWire- (AVT) und Ethernet-Kameras (Prosilica) wurde entwickelt. Für die AVT-Guppy-Kameras wurde die Synchronisierung über den FireWire-Bus implementiert und getestet. Um die optimalen HDR-Parameter der Kameras zu finden, wurden mehrere Testreihen bei Tageslicht bzw. Schatten aufgenommen. Abbildung 72 zeigt Beispiele für schlechte (links) und gute Parametereinstellungen (rechts).

Erste Tests zur visuellen Odometrie anhand von SURF-Features wurden durchgeführt (sparse depth-from-stereo).

EnView kann nun verschiedene env-Items mittels einer Plug-In-Architektur darstellen. Zur Zeit sind Meshes, LaserScans und FrameNodes (Transformationen) implementiert (Abbildung 74). Es ist jetzt möglich, FrameNodes über EnView zu ändern. Das Modell von AsguardV2 kann nun in EnView verwendet werden.

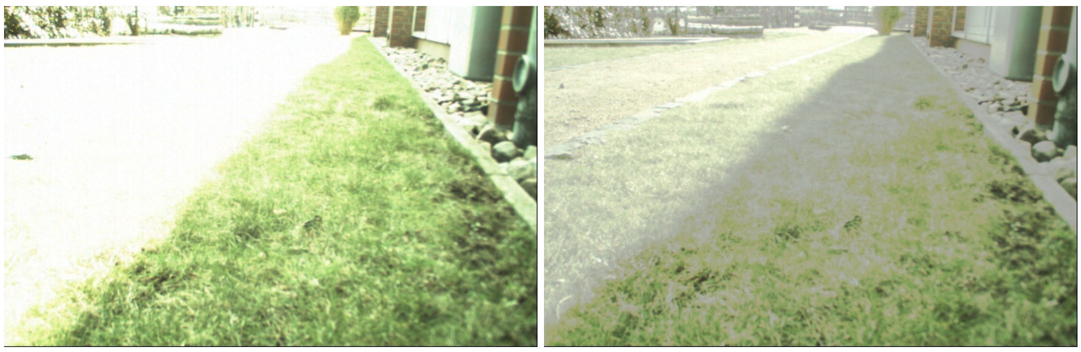


Abbildung 72: *Vergleich schlechter (links) und guter (rechts) HDR-Parameter (Kamera: AVT Guppy)*

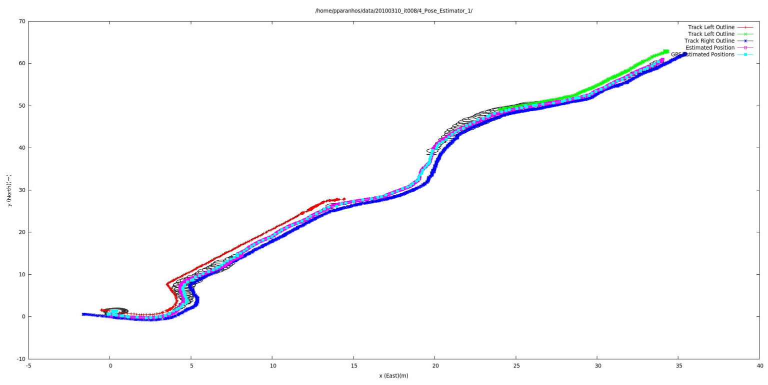
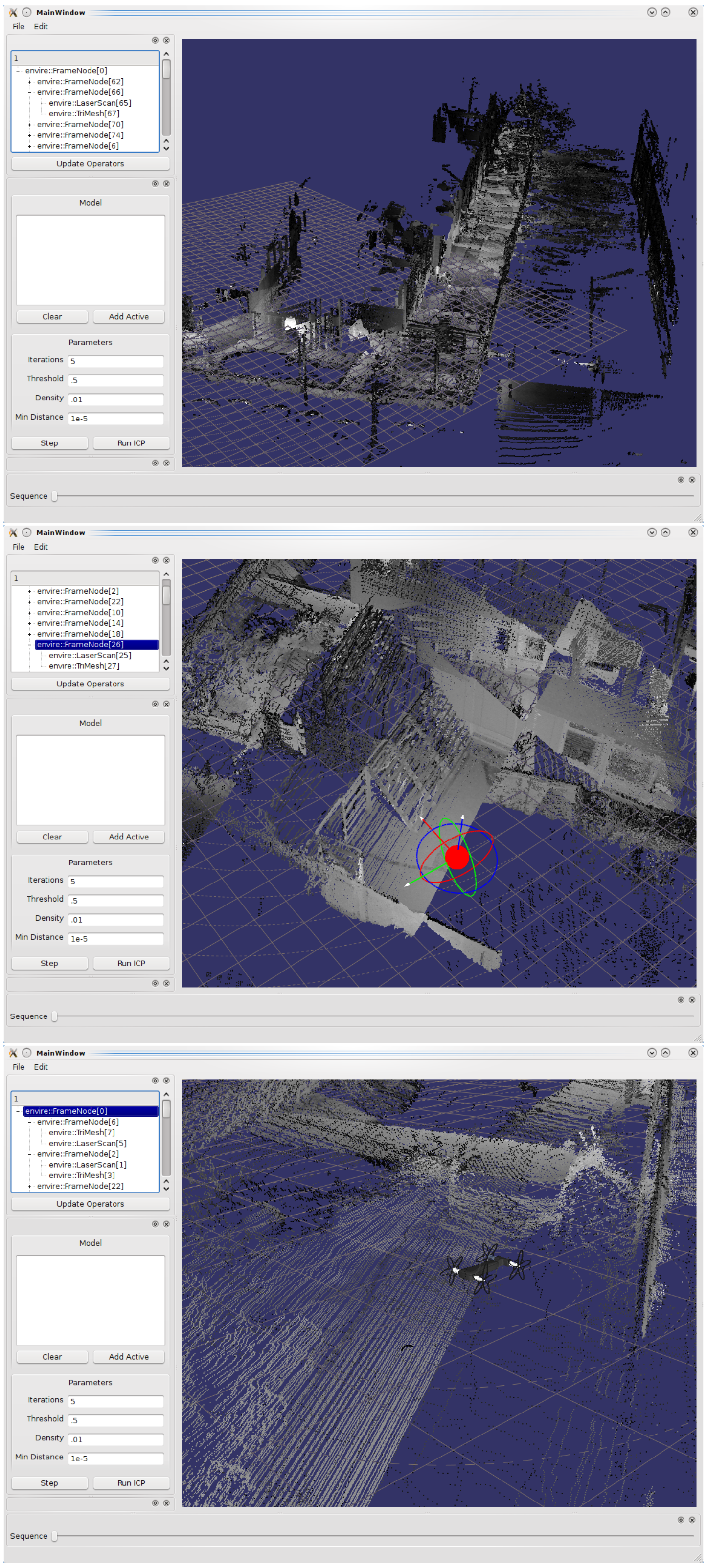


Abbildung 73: *Posenschätzer*



```
1
- envire::FrameNode[0]
  + envire::FrameNode[62]
  - envire::FrameNode[66]
      envire::LaserScan[65]
      envire::TriMesh[67]
  + envire::FrameNode[70]
  + envire::FrameNode[74]
  + envire::FrameNode[6]

Iterations    5
Threshold     .5
Density       .01
Min Distance  1e-5
```

```
1
  + envire::FrameNode[2]
  + envire::FrameNode[22]
  + envire::FrameNode[10]
  + envire::FrameNode[14]
  + envire::FrameNode[18]
  - envire::FrameNode[26]
      envire::LaserScan[25]
      envire::TriMesh[27]

Iterations    5
Threshold     .5
Density       .01
Min Distance  1e-5
```

```
1
- envire::FrameNode[0]
  - envire::FrameNode[6]
      envire::TriMesh[7]
      envire::LaserScan[5]
  - envire::FrameNode[2]
      envire::LaserScan[1]
      envire::TriMesh[3]
  + envire::FrameNode[22]

Iterations    5
Threshold     .5
Density       .01
Min Distance  1e-5
```

Abbildung 74: *EnView*

Die Verarbeitung von Multi-Level Surface Maps wurde implementiert (siehe Abb. 75) und stellt eine zusätzliche Möglichkeit dar, Terraininformation zu repräsentieren.

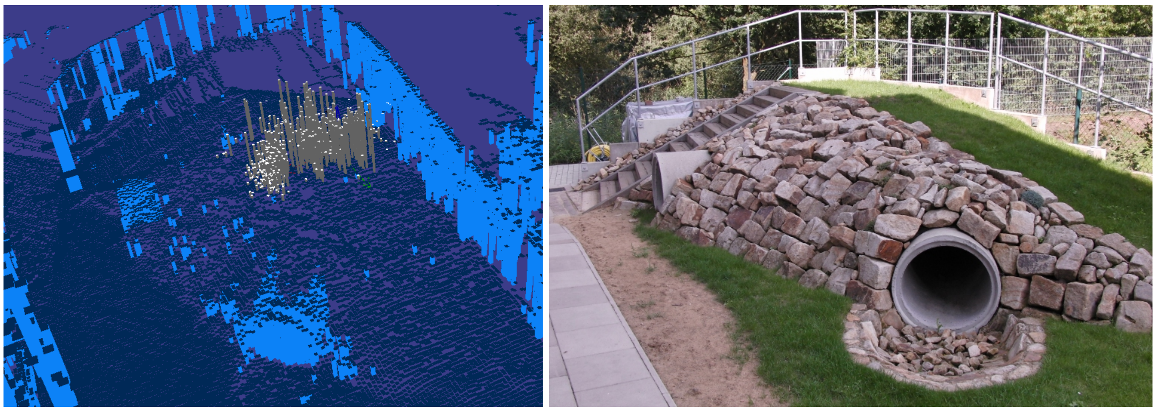


Abbildung 75: *Links: Multi-Level-Surface-Repräsentation der DFKI-Roboter-Teststrecke (Dunkelblau: Horizontale Bereiche. Hellblau: Vertikale Bereiche. Grau: Verteilung der Position des Roboters. Höhe der Balken entspricht relativer Wahrscheinlichkeit der Position).*
Rechts: Foto des repräsentierten Bereichs.

In Sektion 5.6 ist eine Präsentation zu der Struktur der in diesem Arbeitspacket entwickelten Softwarebibliothek angehängt.

WP4200: SLAM mit direkten propriozeptiven Daten

Es wurde ein Kalmanfilter implementiert, der aus den Daten der IMU (Inertial Measurement Unit) und den GPS-Positionsdaten eine Schätzung der aktuellen Position, Geschwindigkeit und Orientierung vornimmt (Abb. 76).

Es ist ein abstrakter Partikelfilter implementiert worden, inklusive eines einfaches Testfalles. Weiterhin wurde eine Implementierung des für den direkten propriozeptiven SLAM-Algorithmus erforderlichen Partikelfilters begonnen. Hier wurde das Odometriemodell aus WP3300 verwendet und in den Partikelfilter integriert. Weiterhin wurde die Visualisierung dahingehend erweitert, dass sie in der Lage ist, sequentielle Logdaten (in diesem Fall den Zustand des Partikelfilters) anzuzeigen sowie zeitlich zu positionieren (siehe auch Abb. 77).

Ein Algorithmus zur Berechnung einer visuellen Odometrie anhand von Stereokamerabildern wurde entwickelt und getestet. Abbildung 78 zeigt die aus den Kamerabildern extrahierten Features und die gefundenen Zuordnungen zwischen linker und rechter Kamera.

Erste Versuche zur propriozeptiven Lokalisierung wurden in der Simulation durchgeführt (s. Abb. 79).

Die mechanische, elektronische und softwareseitige Integration der externen Radsensormodule (Radenkoder) wurde durchgeführt. Damit können nun Radpositionen und Drehmomentwerte auf dem Roboter AsguardV3 ermittelt werden.

Zur Datenaufzeichnung auf der Teststrecke wurden Soft- und Hardware vorbereitet, um die Simulationsergebnisse des direkten eSLAM-Ansatzes zu verifizieren.

Die Implementierung der visuellen Odometrie wurde restrukturiert, sodass sie nun leichter erweiterbar ist. Es wurden erste Tests mit synchronisierten Stereobildern, die vom Roboter aufgenommen wurden, durchgeführt.

Die *Probabilistic Direct Embodied Localisation* wurde implementiert und die Resultate zur ICRA 2011 eingereicht. Abbildungen 80 und 81 zeigen einen Vergleich zwischen Odometrie und direkter Lokalisation auf dem Hügel der Roboter-Teststrecke bzw. auf der gesamten Teststrecke.

Es zeigt sich, dass *Direct Embodied Localisation* insbesondere auf Untergründen, auf denen die Räder leicht durchrutschen (wie z.B. Stufen), gute Ergebnisse liefert.

Die Berechnung der Odometrie wurde überarbeitet, um eine Kompensation von vertikalen Lageveränderungen der Körpermitte des Roboters zu erreichen, die durch die Radgeometrie verursacht werden.

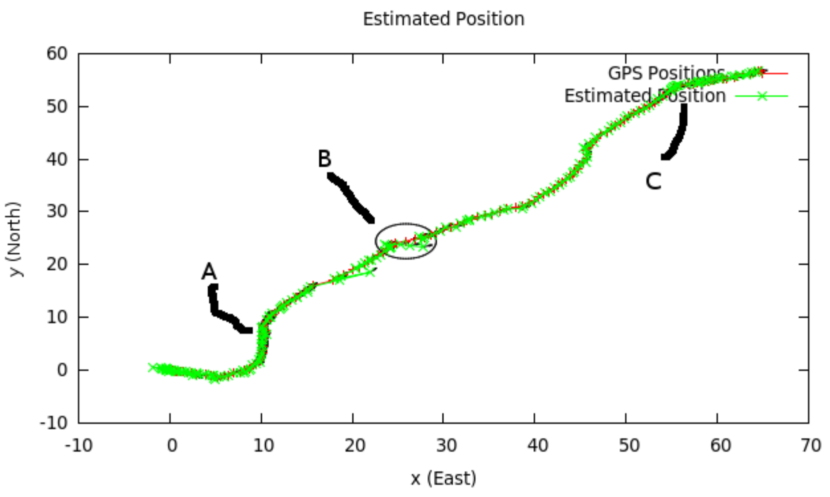


Abbildung 76: *Ergebnisse der Positionsschätzung mittels eines Kalman-Filters*

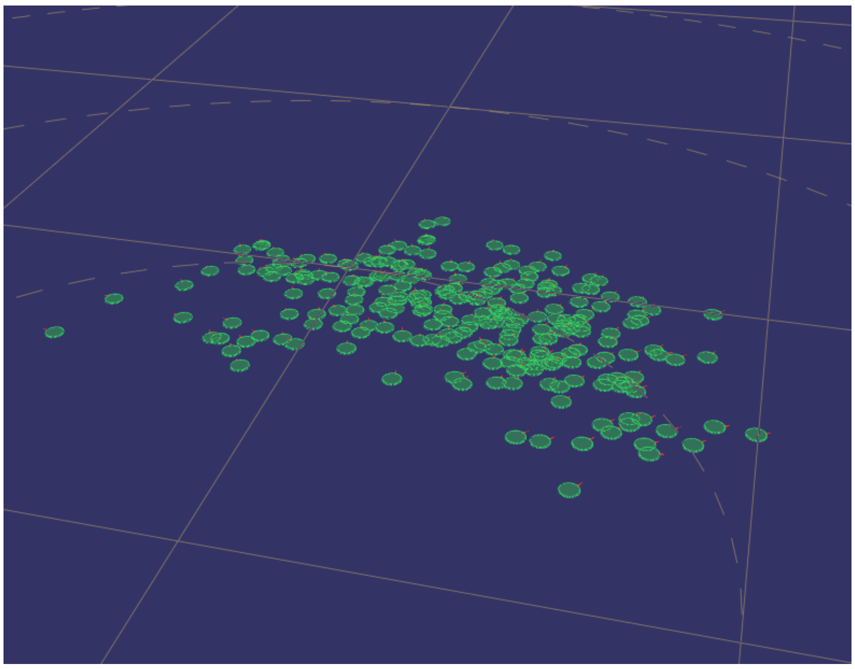


Abbildung 77: *Partikelfilter*

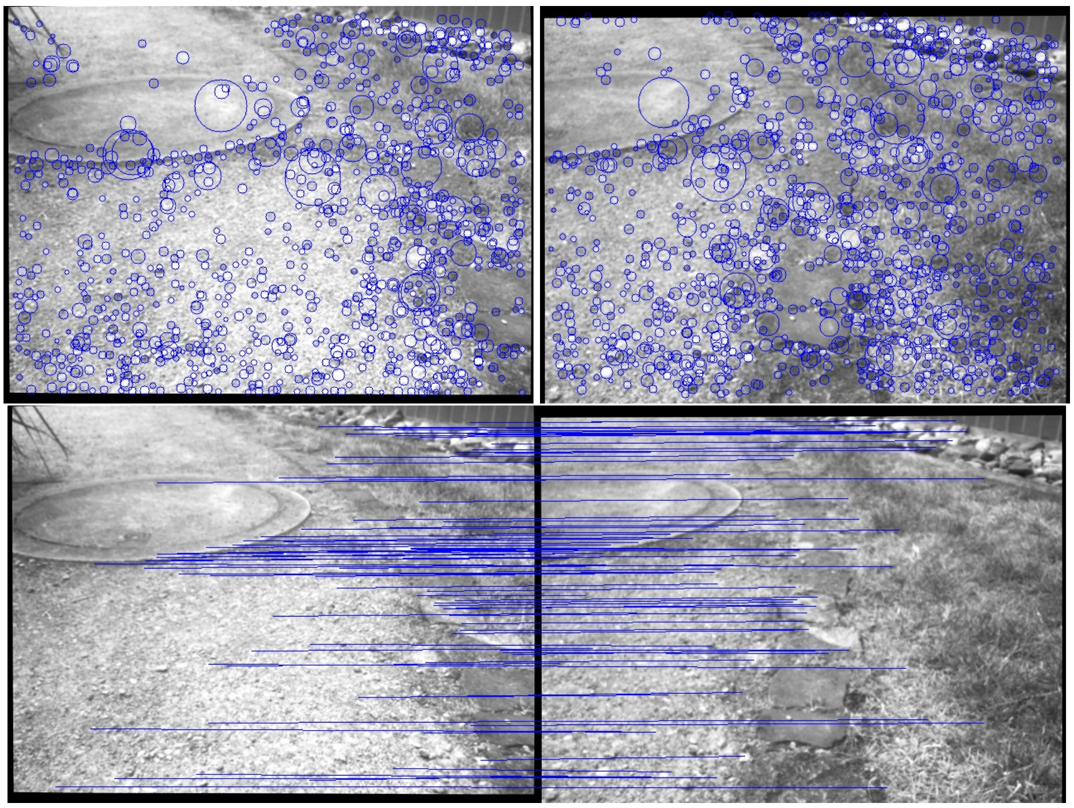


Abbildung 78: *Visuelle Odometrie: SURF-Deskriptoren (oben) und die gefundenen Zuordnungen zwischen linker und rechter Kamera (unten)*

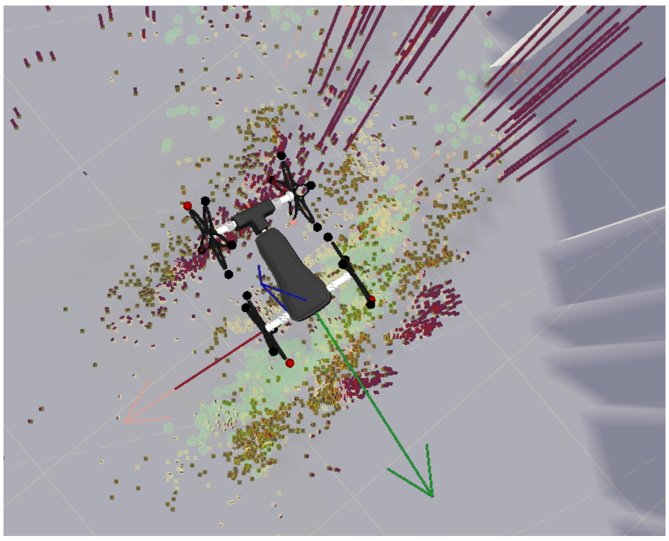


Abbildung 79: *Propriozeptive Lokalisation in der Simulation*

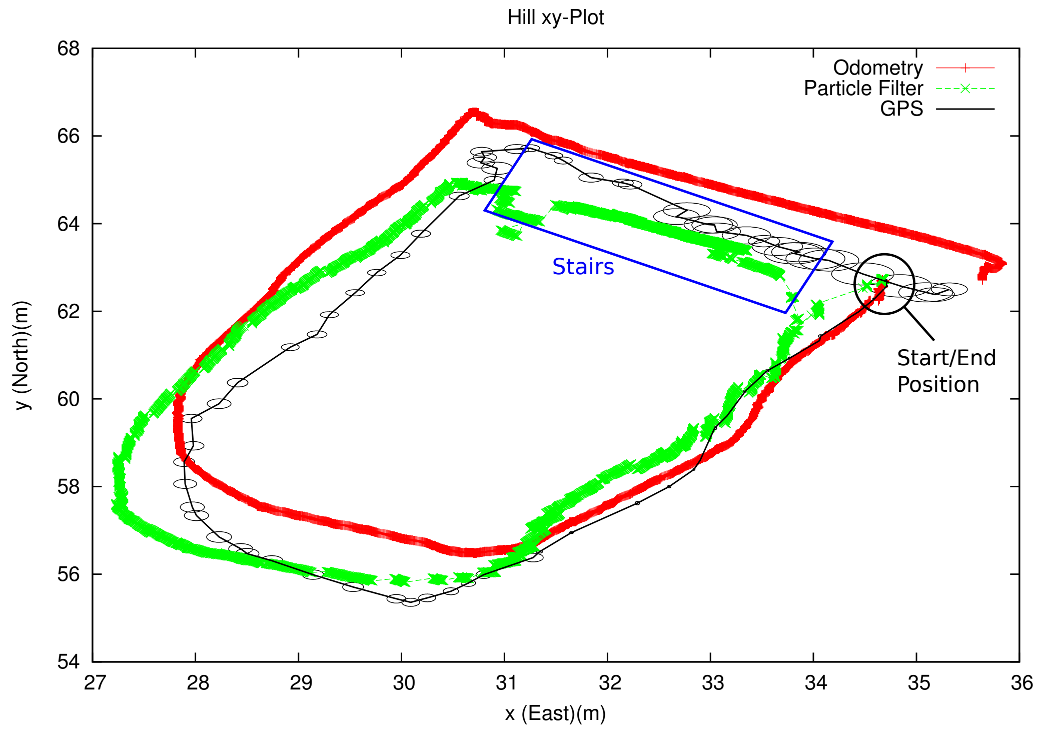


Abbildung 80: *Vergleich der Odometriedaten mit den Ergebnissen der* Probabilistic Direct Embodied Localisation

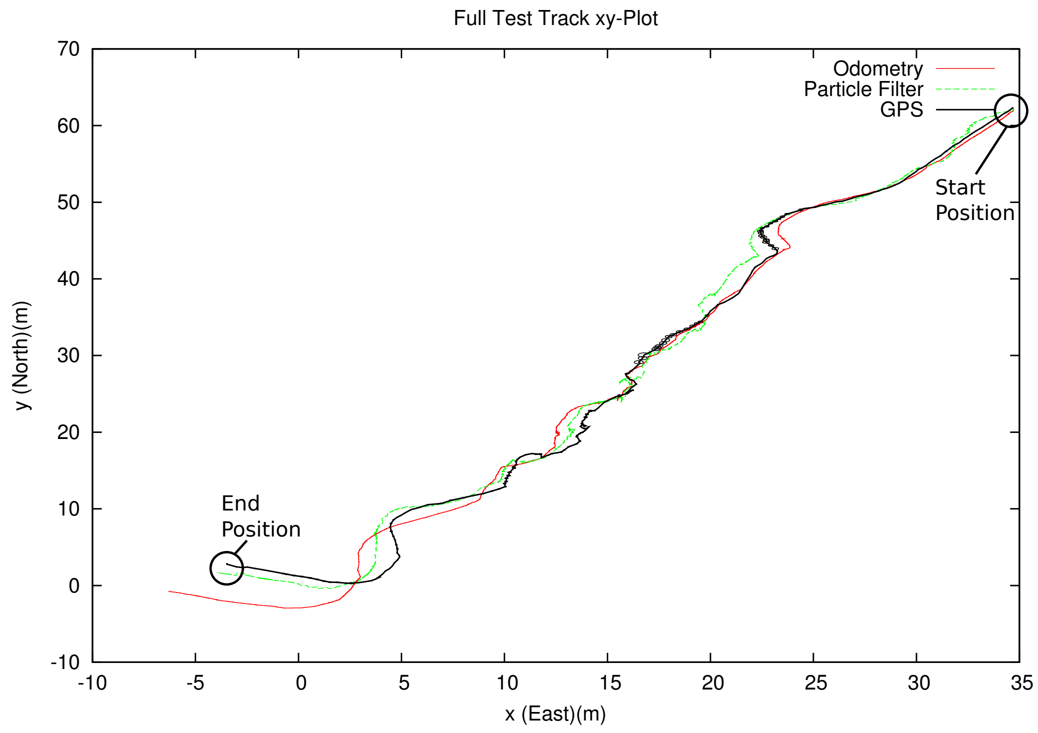


Abbildung 81: *Ergebnisse der direkten Lokalisation auf der gesamten Roboter-Teststrecke*

Die Publikation zur Lokalisation auf Basis von propriozeptiven Daten in Sektion 5.7, sowie die Verbindung von propriozeptiven und visuellen Daten in Sektion 5.9 gibt nähere Details über die entwickelten Methoden.

WP4300: SLAM auf Basis von Terrainklassifikation

Es wurden verschiedene Ansätze zur visuellen Terrainklassifikation implementiert. Verfahren, die auf Farbräumen operieren, werden mit texturbasierten Methoden verglichen. Einer der implementierten Algorithmen, der Daten im HSV-Farbraum (Farbwinkel, Sättigung und Helligkeit) verarbeitet, liefert das in Abb. 82 gezeigte Ergebnis.

Um beim Asguard-Roboter in der Lage zu sein, das tatsächlich auf die Räder wirkende Drehmoment zu bestimmen, muss die Deflektion der elastischen Achsenkupplung in das Drehmoment umgerechnet werden. Da die Übertragungsfunktion von Deflektion zu Drehmoment unbekannt ist, wurden Arbeiten zur Bestimmung dieser Funktion durchgeführt. Dazu wurde ein Versuchsaufbau erstellt, in dem eine definierte externe Last an die Kupplung angelegt werden kann, während das externe Drehmoment gemessen wird. Die Auswertung der Versuchsdaten ergab einen nichtlinearen Zusammenhang zwischen Deflektion und Drehmoment. Daher wurde als Annäherung ein Hysteresemodell nach Bouc, Wen, Baber und Noori implementiert. Die Parameter dieses Modells wurden unter Zuhilfenahme des Levenberg-Marquardt-Algorithmus bestimmt. Hierbei wurden die Einflüsse von Getriebespiel und statischer Reibung berücksichtigt und herausgerechnet. Die resultierende Übertragungsfunktion ist in Abb 83 dargestellt.

Die Bestimmung des Drehmoments ermöglicht es, ein Durchrutschen der Räder aufgrund wechselnder Bodenbeschaffenheit zu detektieren sowie die Güte der Odometrie zu verbessern. Des Weiteren können die Drehmomente der Räder als Eingabe für das dynamische Bewegungsmodell des Roboters dienen, wodurch die Regelung des Roboters verbessert werden kann. Die Integration in das dynamische Modell wurde abgeschlossen und Tests wurden durchgeführt.

Ein *Expectation-Maximization*-Algorithmus für *Gaussian Mixture Models* wurde als Partikel-Clusterer implementiert. Da der Partikelfilter multiple Hypothesen in Bezug auf den Systemzustand nutzt, können durch das Clustering mehrere Partikel zu einer Hypothese verschmolzen werden. Dies ermöglich bzw. vereinfacht die Weiterverwendung der Ergebnisse in nachfolgenden Berechnungen.

Die existierenden Werkzeuge zur Anzeige der Umgebung sowie des Zustandes der SLAM-Komponenten wurden verbessert. Dadurch ist es nun möglich, größere Datensätze zu verarbeiten, ohne dass die Performance bei der Anzeige einbricht.

Es wurden Verbesserungen an der Interpretation des Zustandes des SLAM durch die Anwendung von Clustering-Methoden und einem *Gaussian Mixture Model* vorgenommen. Dadurch ist es möglich, den Zustand des SLAM-Ansatzes, der aus Hunderten von Positionshypothesen besteht, auf wenige Hypothesen-Cluster zu reduzieren.

Die Klassifikation von Terrainarten durch Nutzung interner Sensorinformationen basiert auf der Tatsache, dass Untergründe verschiedene Traktionsprofile in Interaktion mit dem Roboter aufweisen. Eine wichtige Eigenschaft ist dabei, dass nur eine Unterscheidung möglich ist, wenn die zum Terrain tangentiale Kraft größer ist als die maximale Haftreibungskraft zwischen Fuß und Terrain und das Rad somit rutscht (slip). Die zum Terrain tangentiale Kraft kann über die orthogonale Kraft und das Drehmoment des Rades abgeschätzt werden. Zur Klassifizierung des Terrains werden die Messwerte der tangentialen Kraft beim Durchrutschen zusammengefasst und mit Hilfe einer *Support Vector Machine* (SVM) bewertet. Die SVM muss für die jeweilige Situation trainiert werden. Abbildung 85 zeigt die Schritte zur Klassifizierung.

Da die auf dem HSV-Farbraum basierende Klassifikation verschiedener Terrainarten (siehe Fig. 86) z.T. sehr anfällig gegenüber Beleuchtungsänderungen ist, wurde der Hokuyo-Laserscanner als weitere Informationsquelle in Betracht gezogen. Dabei sollen nicht die Distanzmesswerte, sondern die Remissionswerte verwendet werden. Diese repräsentieren die Intensität, mit der die künstlich beleuchtete Umgebung Laserlicht einer Wellenlänge von 905 nm reflektiert. Zur erfolgreichen Nutzung dieser Daten müssen die Distanz zum Untergrund sowie der Einfallswinkel des Laserlichts einbezogen werden.

Eine Literaturrecherche und Testaufnahmen haben gezeigt, dass die Remissionswerte als Differenzierungskriterium zwischen einigen Terrainklassen geeignet sind. Zur Sensorfusion zwischen Kameras und Laserscanner ist eine Kokalibrierung der Sensoren erforderlich.

In diesem Arbeitspaket sind die Ergebnisse der Terrainklassifizierung auf Basis von Radschlupf (Slip) sowie der Klassifizierung durch visuelle Merkmale zusammengeführt worden. Das Funktionsprinzip ist in Abb. 87 dargestellt. Es werden dabei visuell erstellte Klassifikationen in eine Karte projiziert und mit schlupfbasierten Events verglichen (siehe Abb. 88). Dabei kommt eine Wahrscheinlichkeitsfunktion zum Tragen, die die kombinierte Wahrscheinlichkeit von beiden Messungen zusammenführt und somit die Wahrscheinlichkeit einzelner Partikel in einem Partikelfilter beeinflusst. Die Funktion ist auf Logdaten des AsguardV3-Roboters auf der DFKI-Teststrecke verifiziert worden. Eine beispielhafte Karte ist in Abb. 89 dargestellt. Die Ergebnisse der Messungen sind mit unterschiedlichen Messmodellen verglichen worden (sie Abb. 90). Dabei wurde ein Verringerung des Fehlers durch den Einsatz der vorgeschlagenen Methode festgestellt. Die Methode und Ergebnisse sind zu einer wissenschaftlichen Publikation zusammengefasst und bei der IROS 2012 eingereicht worden. Die Ziele des Arbeitspakets sind somit erreicht worden und die Aktivitäten in diesem AP beendet.

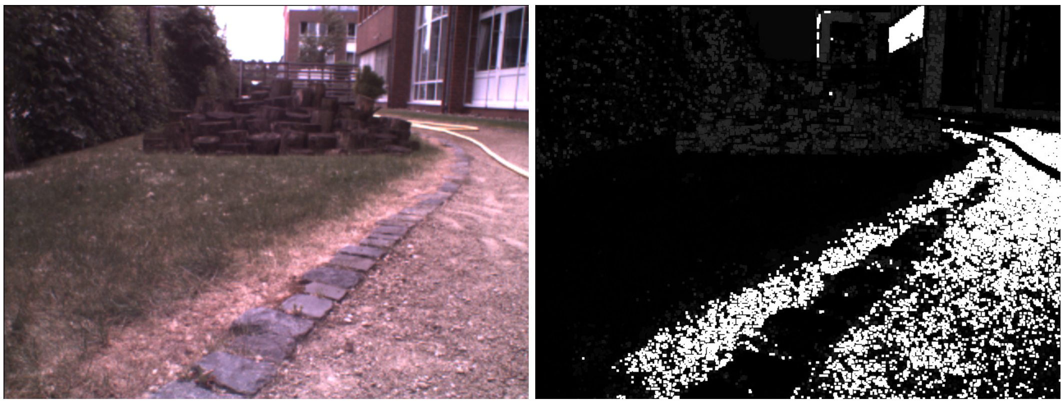


Abbildung 82: *Ergebnis der visuellen Terrainklassifikation.Links: Input-Bild. Rechts: Terrainklasse "Pfadïn weiß.*

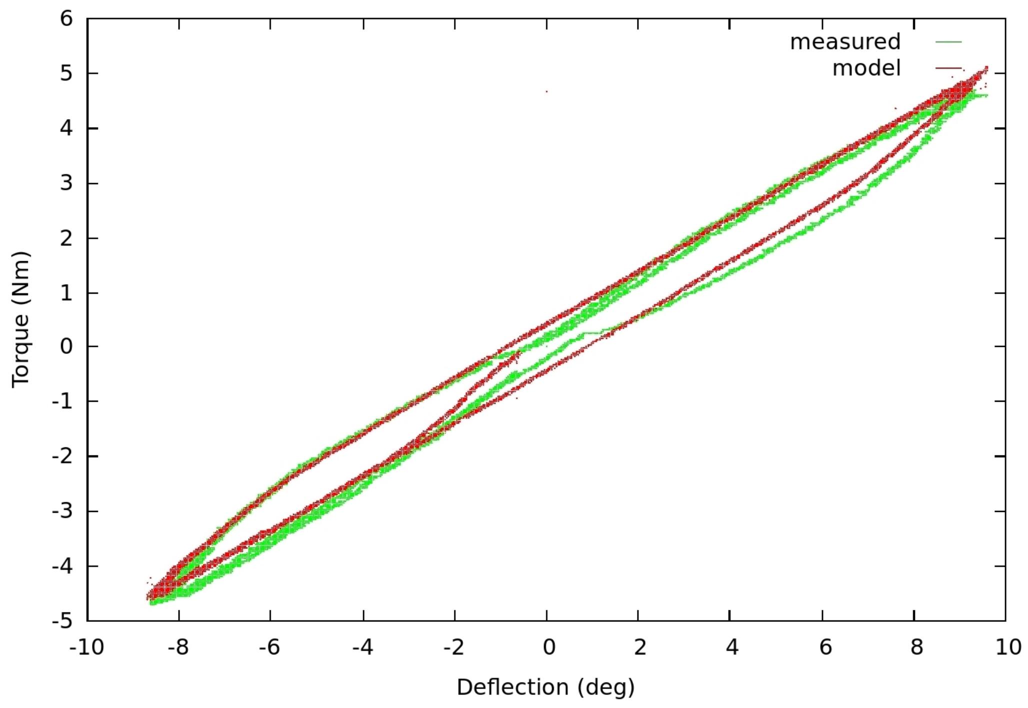


Abbildung 83: *Hysterese: Vergleich zwischen erzeugtem Modell und Messdaten eines externen Drehmomentsensors. Die x-Achse zeigt die Deflektion der elastischen Kupplung in Grad, die y-Achse das Drehmoment in Nm.*

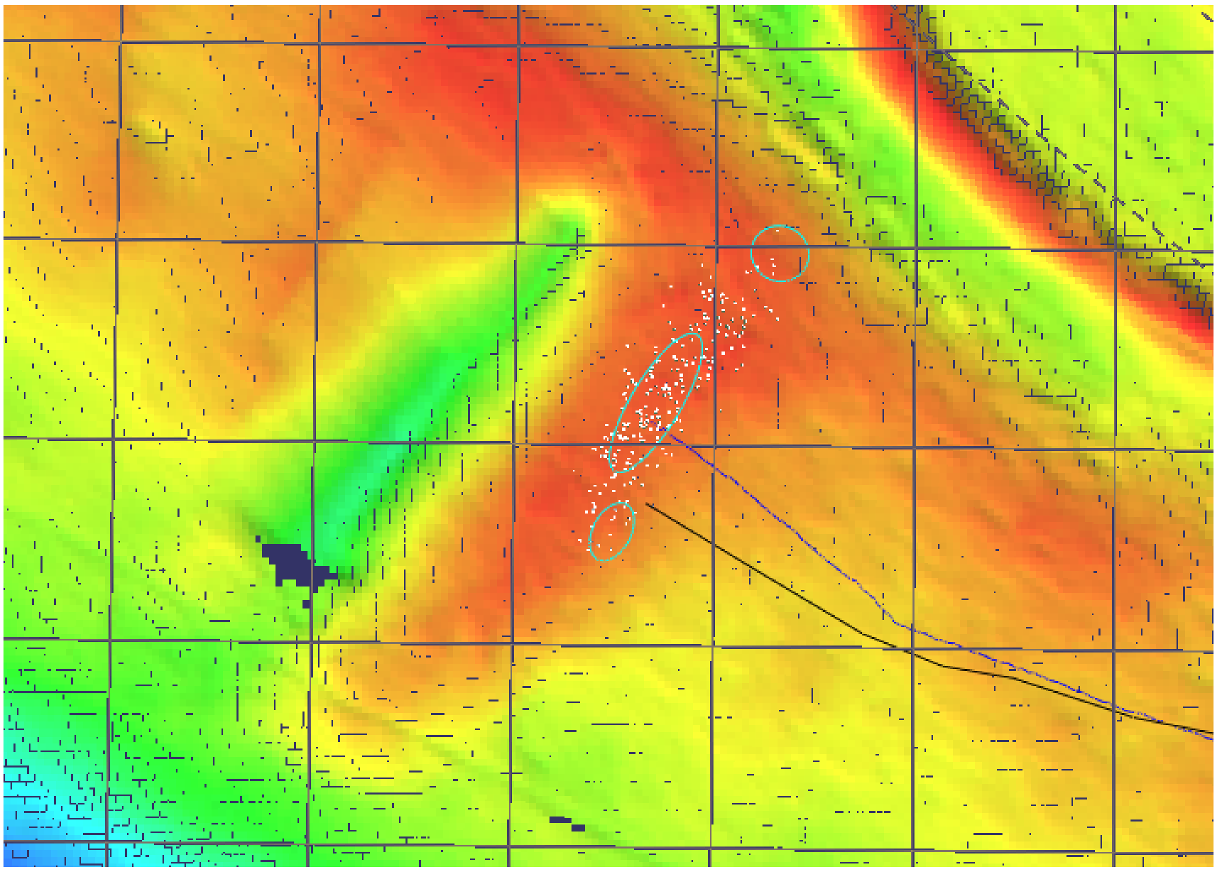


Abbildung 84: *Die Position des Roboters wird durch Partikel repräsentiert. Unter Anwendung eines* Gaussian Mixture Model *und einer* Maximum Likelihood*-Analyse werden die Partikel zusammengefasst (cyanfarbene Ellipse).*

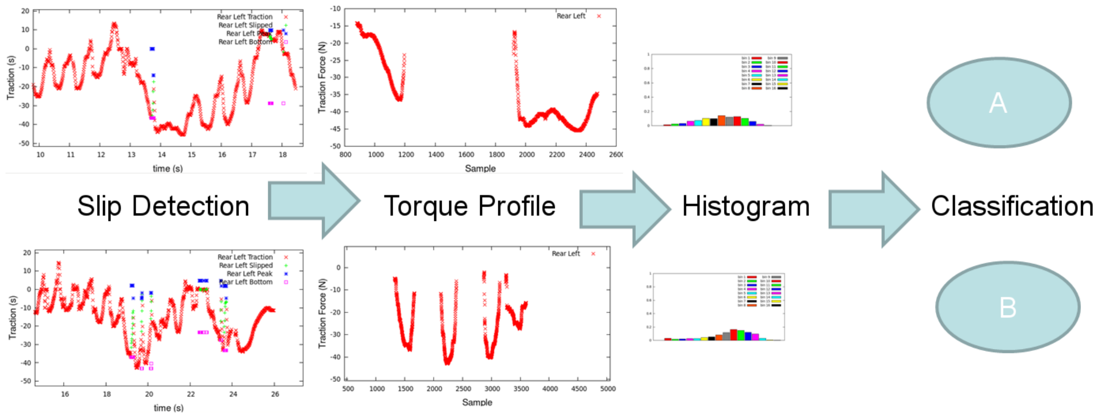


Abbildung 85: *Die Klassifizierung des Terrains basiert auf Drehmomentmessungen und der Detektion des Durchrutschens eines Rades. Intervalle, in denen ein Durchrutschen auftritt, werden mit Hilfe einer* Support Vector Machine *(SVM) einer Terrainklasse zugeordnet.*

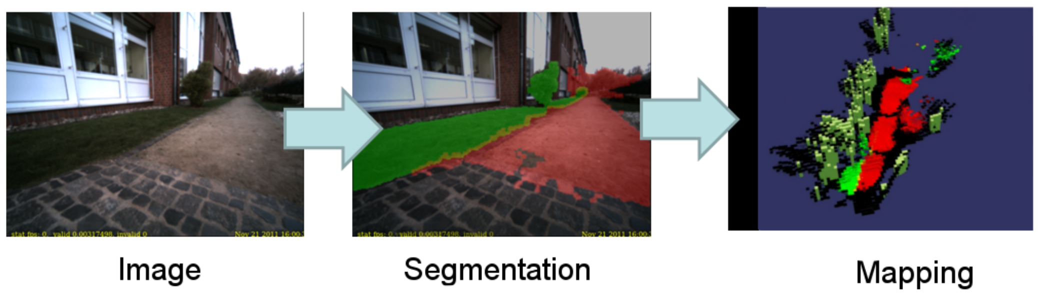


Abbildung 86: *Die visuelle Terrainklassifikation nutzt Farbbilder der Kameras, um zwischen einzelnen Terrainarten visuell zu unterscheiden. Anschließend wird das Klassifikationsergebnis über eine Rückprojektion in die aktuelle Karte eingetragen.*

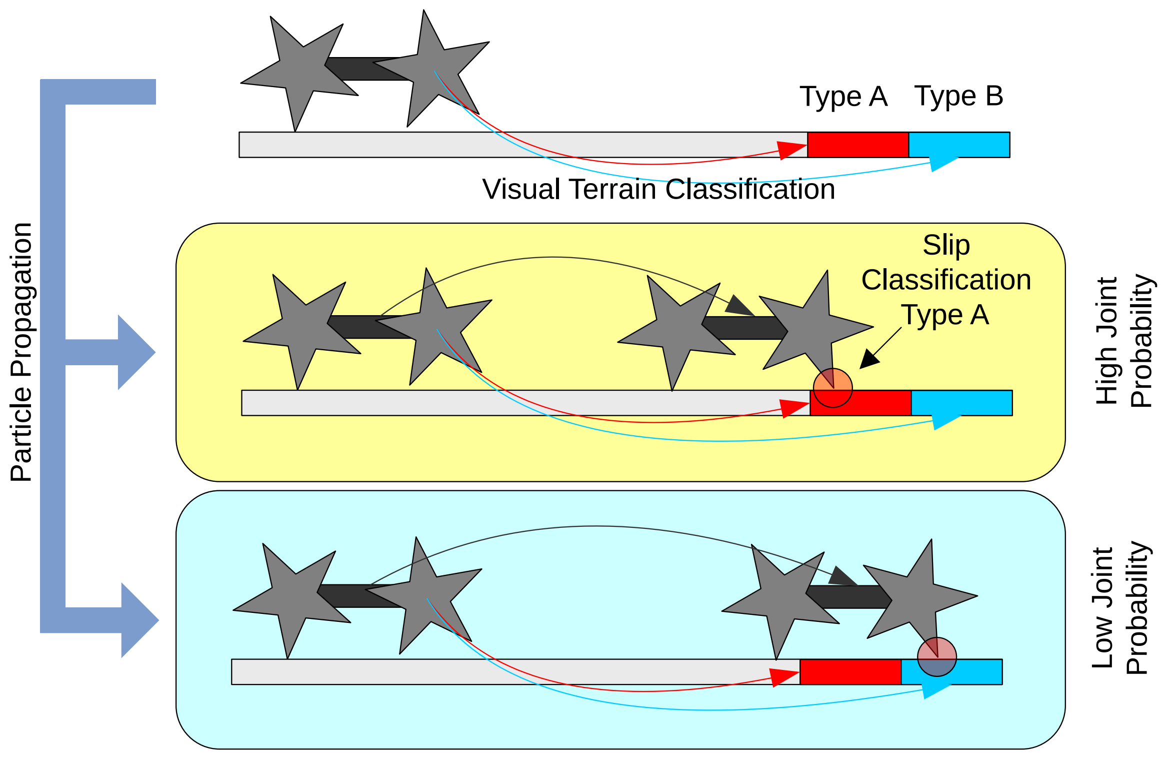


Abbildung 87: *Das Konzept der simultanen Kartierung und Lokalisierung auf Basis von* indirect embodied data *ist hier dargestellt. Durch eine visuelle Klassifizierung (oben) werden Terraintypen in eine Karte eingezeichnet. Eine Klassifizierung auf Basis von Radschlupf wird mit dieser Information kombiniert und kann so zu einem hohen (Mitte) oder niedrigen (unten) Gewicht der einzelnen Partikel des Filters beitragen.*

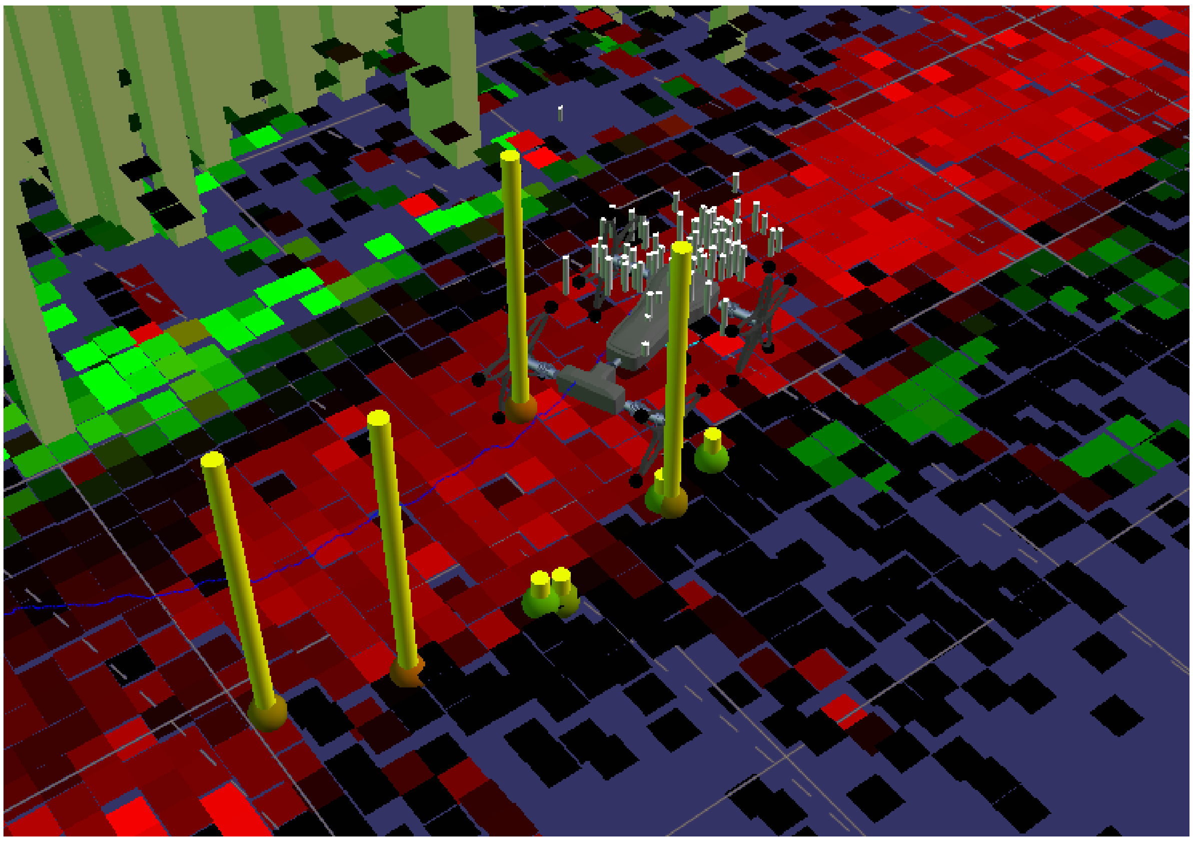


Abbildung 88: *Visualisierung des internen Zustandes des Filters für Logdaten des AsguardV3-Roboters auf der DFKI-Teststrecke. Die Partikel sind als graue Balken dargestellt, deren Höhe proportional zu ihrem Gewicht ist. Die gelben Balken zeigen slip-basierte Messungen an. Deren Höhe ist proportional zu der kombinierten Wahrscheinlichkeit von visueller und slip-basierter Messung.*

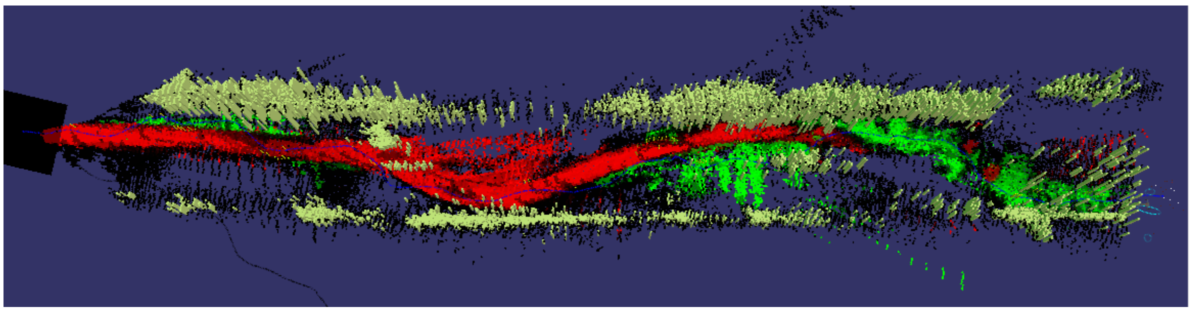


Abbildung 89: *Resultierende Karte der DFKI-Teststrecke. Visuell klassifizierte Bereiche sind rot (Weg) und grün (Gras) markiert. Schwarze Bereiche konnten nicht klassifiziert werden. Wände werden durch gelbe Balken angezeigt.*

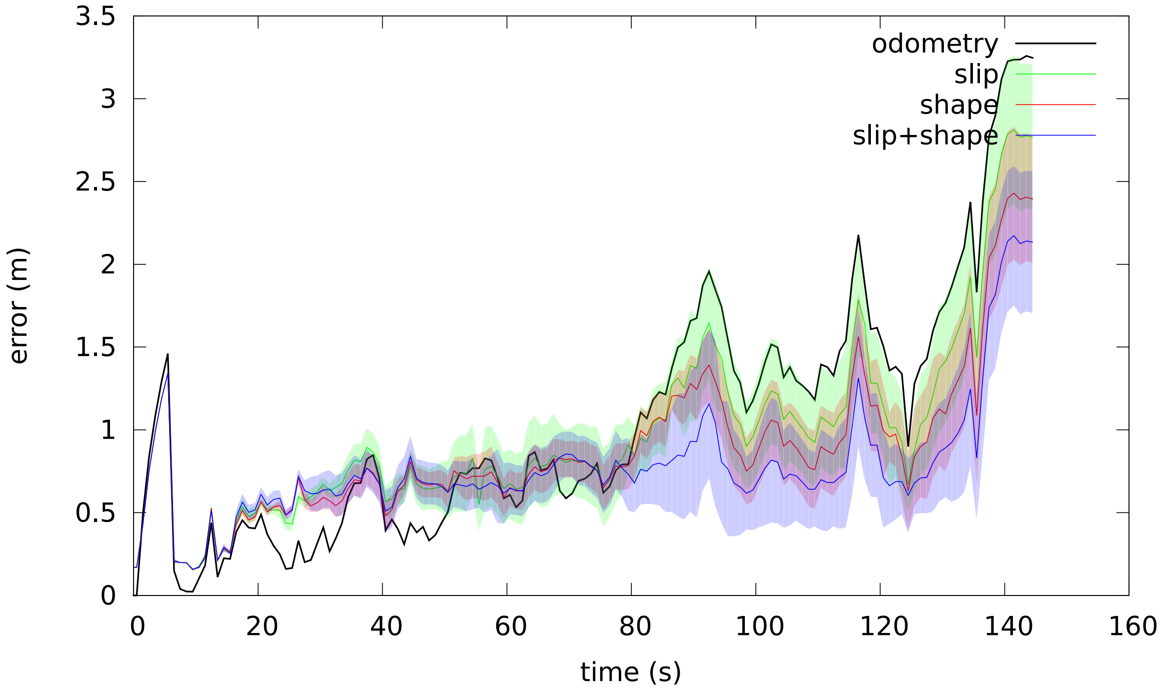


Abbildung 90: *Fehler im Vergleich zur Referenztrajektorie. Dabei wurden einzelne Messmodelle miteinander verglichen. Für jede Konfiguration wurden fünf Durchläufe genutzt und der Mittelwert sowie die Standardabweichung angezeigt. Es zeigt sich, dass beide Messmodelle (AP 4200 und AP 4300) das Verhalten in Bezug auf die Odometrie verbessern und in Kombination noch bessere Ergebnisse liefern.*

In Sektion 5.8 ist das Paper zur Selbstlokalisation auf Basis von indirekten propriozeptiven Daten eingefügt, welches die Inhalte dieses WP genauer beschreibt.

WP4400: Visual SLAM und eSLAM Integration

Visuelle Daten wurden in ein kombiniertes SLAM-Framework integriert. Zur Selbstlokalisierung des Roboters wird ein *Extended Kalman Filter* verwendet, der Daten von der Odometriesensorik, die GPS-Position und ICP-Daten aus dem Laser-Scan-Matching integriert. Um zu detektieren, ob Sensoren falsche oder stark rauschende Daten liefern, die verworfen werden sollten, wird ein statistischer Chi-Quadrat-Test verwendet. Dies erhöht die Robustheit des Filters gegenüber falschen Sensordaten und steigert somit die Güte der Selbstlokalisierung. Die Ergebnisse der Posenschätzung zeigt Abb. 91. Hier sind ungültige Sensordaten blau, gültige Messungen rot dargestellt.

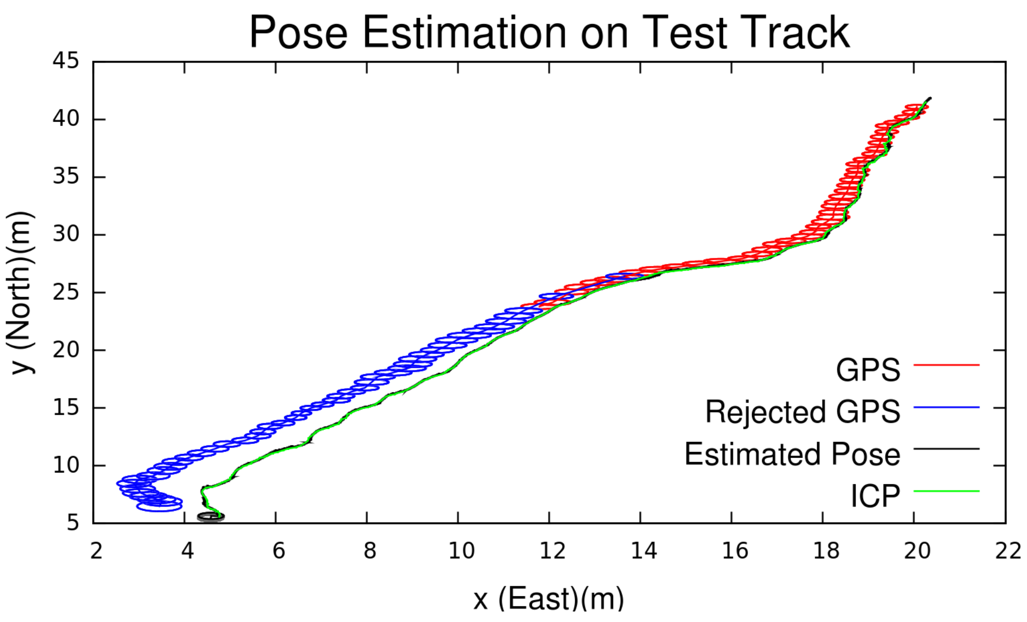


Abbildung 91: *Ergebnis der Integration von Odometrie, GPS und ICP-Daten in einem* Extended Kalman Filter

Zur Erstellung einer Höhenkarte der Umgebung können u.a. aus den Stereokameradaten berechnete Tiefenbilder genutzt werden. Daher wurden verschiedene *Dense-Stereo*-Algorithmen getestet und die Ergebnisse verglichen. Hierfür wurden Kamerabilder von der Roboterteststrecke verwendet. Eines der Ergebnisbilder zeigt Abb. 92.

Ein Verfahren zu Berechnung von Entfernungsinformationen aus Stereokamerabildern wurde mit alternativen Ansätzen verglichen und aufgrund seiner guten Funktionsfähigkeit ausgewählt. Dieses Modul wurde in das bestehende Softwareframework integriert. Tests mit Aufnahmen der onboard-Kameras wurden durchgeführt. (Abbildung 93)

Um alle bildbasierten Module mit hinreichend guten Daten zu versorgen, wurde ein Unschärfedetektor implementiert. Durch die Nutzung dieses Softwaremoduls soll die Ergebnisgüte nachfolgender Algorithmen verbessert werden.

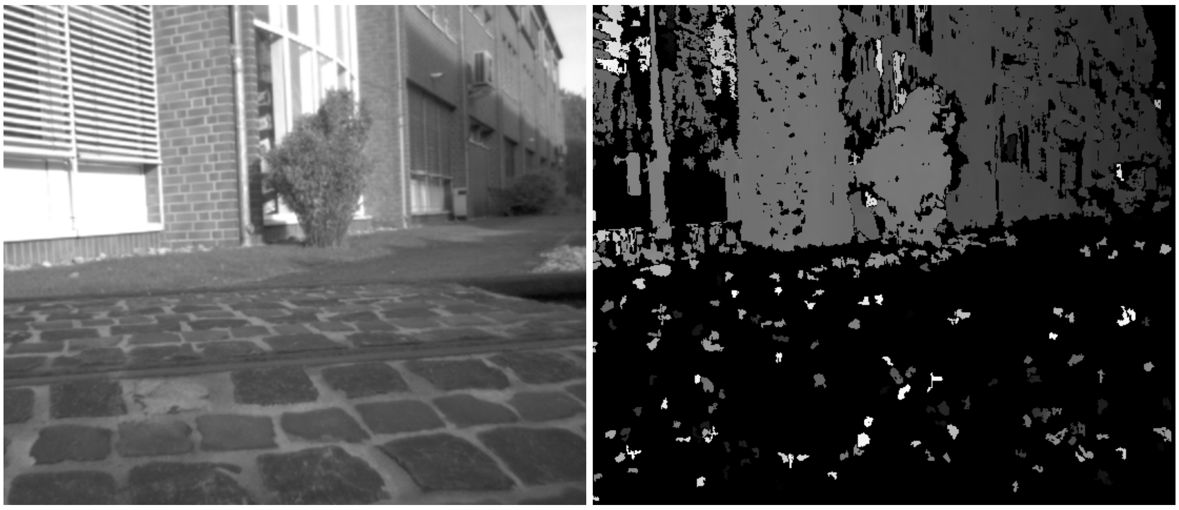


Abbildung 92: *Ergebnis der Dense-Stereo-Tests. Links: Eines der Input-Bilder. Rechts: Berechnetes Tiefenbild. Schwarz: Keine Tiefeninformation extrahierbar, dunkle Farben entsprechen weiter entfernten Objekten, helle dem Nahbereich.*

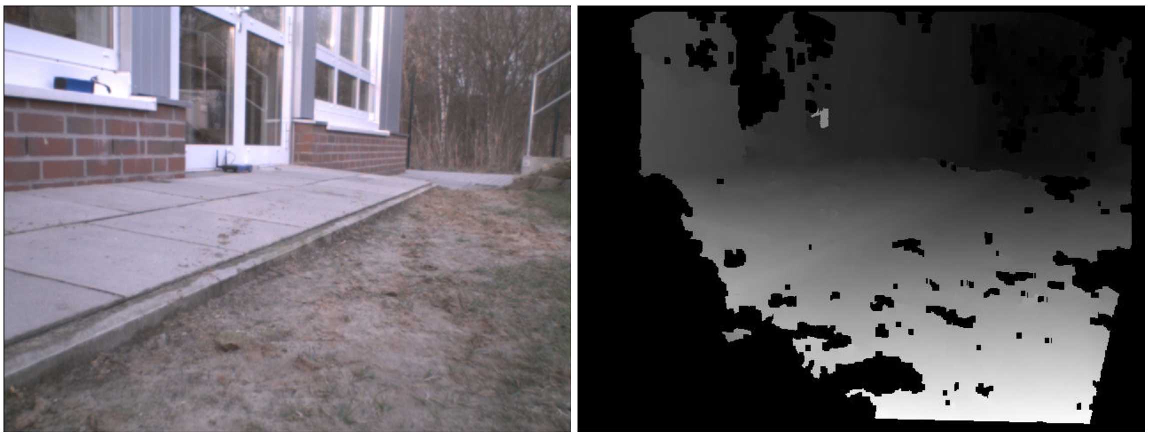


Abbildung 93: *Kamerabild und aus dem entsprechenden Stereopaar errechnetes Tiefenbild.*

Es wurden Arbeiten zur Integration der Ergebnisse der Stereovision in dem Kartenerstellungsprozess durchgeführt.

Es wurden verschiedene verfügbare SLAM-Backends auf Basis von GraphSLAM-Ansätzen evaluiert. Als besonders geeignet hat sich das *HOG-Man*-Framework herausgestellt. Es ist in das verwendete Softwareframework **Rock** integriert worden.

Für einen SLAM-Algorithmus müssen verschiedene Posen des Roboters in Zusammenhang gebracht werden. Um dies zu ermöglichen, ist ein *sparse stereo*-Ansatz gewählt worden. Basierend auf Ergebnissen aus dem Projekt CUSLAM wurden *sparse feature extractors* auf Stereobilder angewandt und die Korrelation zwischen Bildpaaren berechnet (siehe Abb. 96). Desweiteren wurde ein RANSAC-Algorithmus implementiert, der mit Hilfe eines geometrischen Constraints echte Korrelationen von „Ausreißern” unterscheidet. Die so gefilterten Daten können in einem weiteren Schritt in 3D-Feature-Wolken umgerechnet werden (Abb. 97). Diese Wolken können dann jeweils zueinander korreliert werden.

Die Implementierung des visuellen SLAM-Ansatzes mit dem Hog-Man-Framework ist abgeschlossen worden. Als Resultat werden sowohl dichte Stereobilder als auch extrahierte 3D-Features in eine gemeinsame Karte eingetragen. Abb. 98 zeigt, wie die einzelnen Kartenmodalitäten zusammengeführt und mit Texturinformationen annotiert werden.

Weiterhin kann nun durch die Integration des Hog-Man Frameworks ein *Loop Closing* durchgeführt werden. Dabei werden Bereiche, die bereits kartiert worden sind, bei einem erneutem Überfahren als solche identifiziert und mit den alten Positionen assoziiert. So ist es möglich, Karten zu erstellen, die keine globalen Inkonsistenzen aufweisen. Dies wäre z.B. der Fall, wenn ausschließlich die Odometriedaten verwendet werden.

Die Arbeiten an der Integration von visuellem SLAM und embodied SLAM sind abgeschlossen worden. Dabei wurde die bereits vorhande Integration und Kapselung des Graph-Optimierers Hog-Man erweitert und generalisert. Somit ist es möglich geworden, den partikelfilterbasierten Ansatz des eSLAM durch die Schnittstelle der Umgebungsrepräsentation aus WP4100 mit dem Graph-Optimierer zu verbinden.

Dabei erzeugt der Partikelfilter einzelne Kartenabschnitte, die jeweils alle möglichen Karten der einzelnen Partikel enthalten, und speichert diese als Segmente ab. Diese Segmente sind miteinander verknüpft. Einerseits gibt es eine Verknüpfung, die auf der Tatsache beruht, dass die Segmente aufeinander folgend erstellt worden sind. Die aufsummierten Fehler aus der Odometrie bilden somit einen räumliche Beziehung inklusive einer Schätzung des Fehlers. Weiterhin werden die visuellen Methoden verwendet, um einen möglichen Kreisschluss zu erkennen. Ist dies der Fall, so kann der immer weiter wachsende Fehler durch eine Korrelation mit einem früher gebildeten Segment reduziert werden. Dies ist bereits für rein visuelle Methoden gezeigt worden. Die Neuerung in der Integration ist die Ausrichtung der einzelnen Kartensegmente. Durch den Graph-Optimierer werden die Beziehungen zwischen den Segmenten so optimiert, dass der gesamte Fehler minimal ist (Abb.100). Dies führt dazu, dass in einem lokalen Segment auch andere Partikel als die mit der höchsten Gewichtung für die globale Karte genutzt werden können. Daraus ergeben sich wesentlich besser passende Übergänge zwischen einzelnen Kartensegmenten (siehe Abb.101) im Vergleich mit der reinen visuellen Methode, die bereits implementiert worden ist.

Die Ergebnisse des Arbeitspaketes sind in einer Publikation mit dem Titel “Map Segmentation based SLAM using Embodied Data” zusammengefasst und bei der IEEE MFI Konferenz eingereicht worden. Mit diesen Arbeiten ist das Arbeitspaket abgeschlossen. Die Publikation ist in 5.9 zu finden.

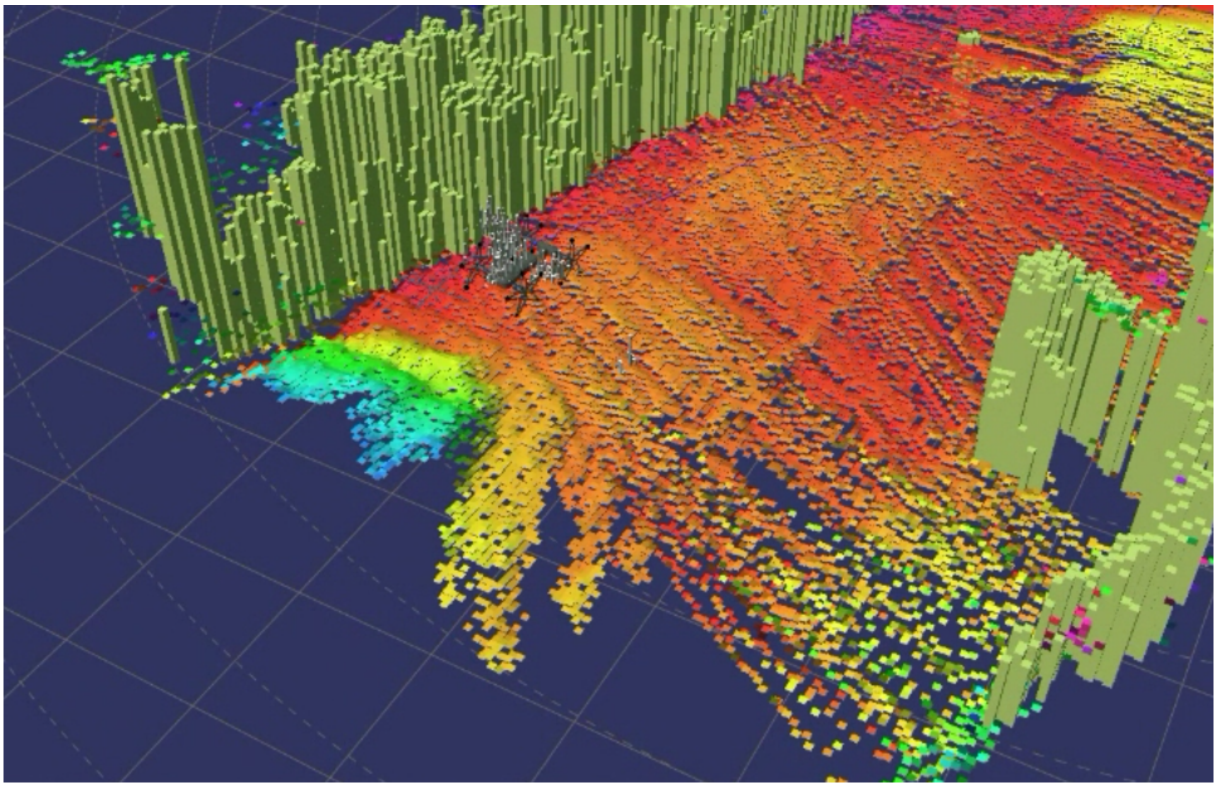


Abbildung 94: *Integration der* Stereo Vision-*Daten in die Kartenerstellung. Das Stereobild liefert ein grobes Tiefenbild des Geländes, das anschließend durch die Informationen des Laserscanners verfeinert wird.*

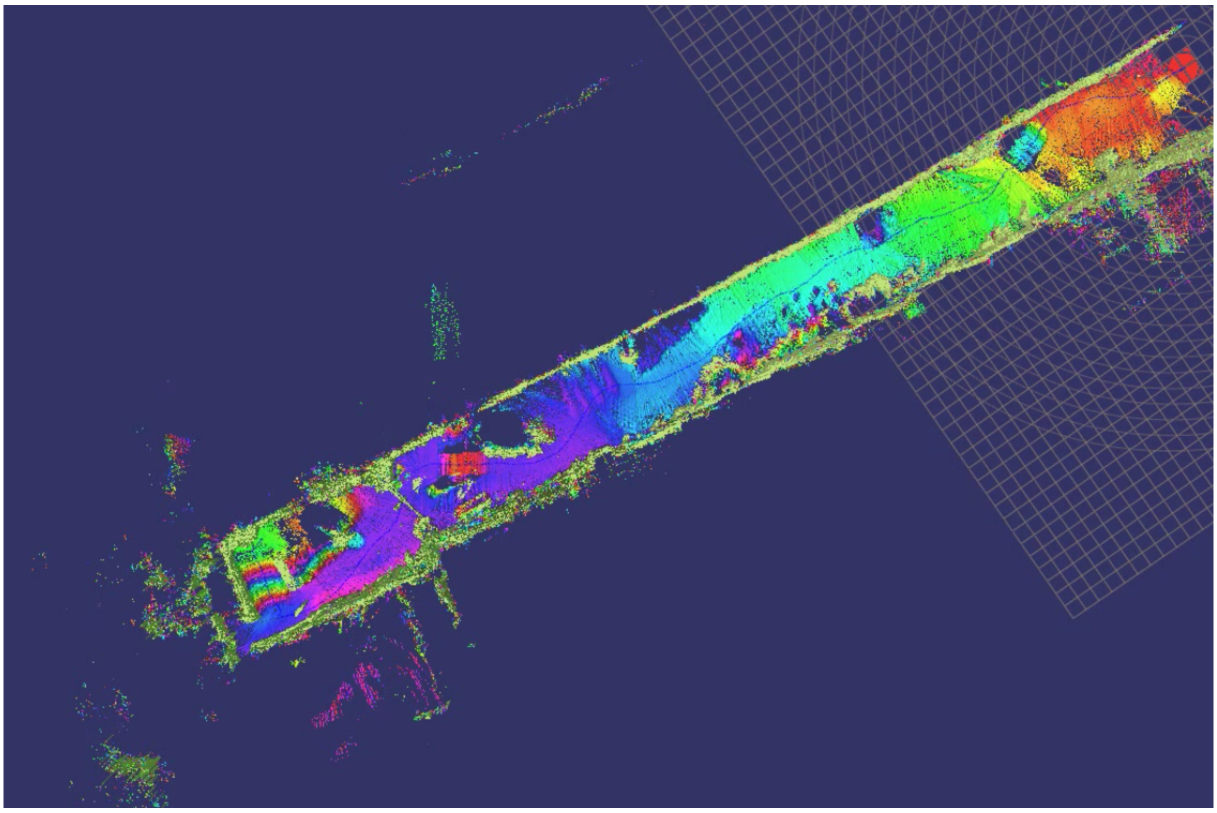


Abbildung 95: *Die Integration der Stereo- und Laserscanner-Daten erlaubt das Generieren von vollständigen Umgebungskarten. Beispiel: Karte des DFKI-Testgeländes aus der Vogelperspektive*

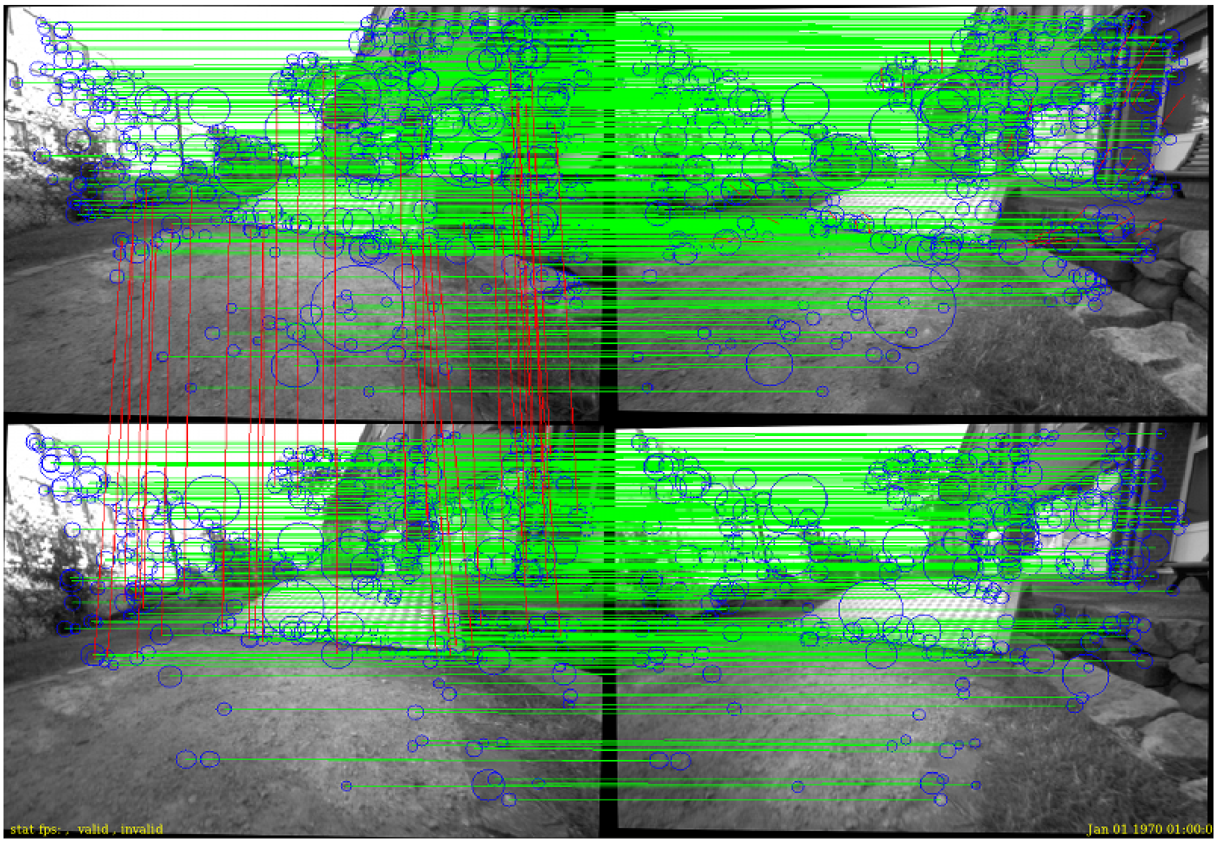


Abbildung 96: *Zwei Stereo-Bildpaare, deren markante Bildpunkte (Features) durch geometrische Constraints korreliert sind. Die grünen Linien basieren auf den epipolaren Constraints des Stereosystems. Die roten linien beschreiben die Korrelation von Bildpaaren aus unterschiedlichen Zeitschritten und somit Roboterpositionen.*

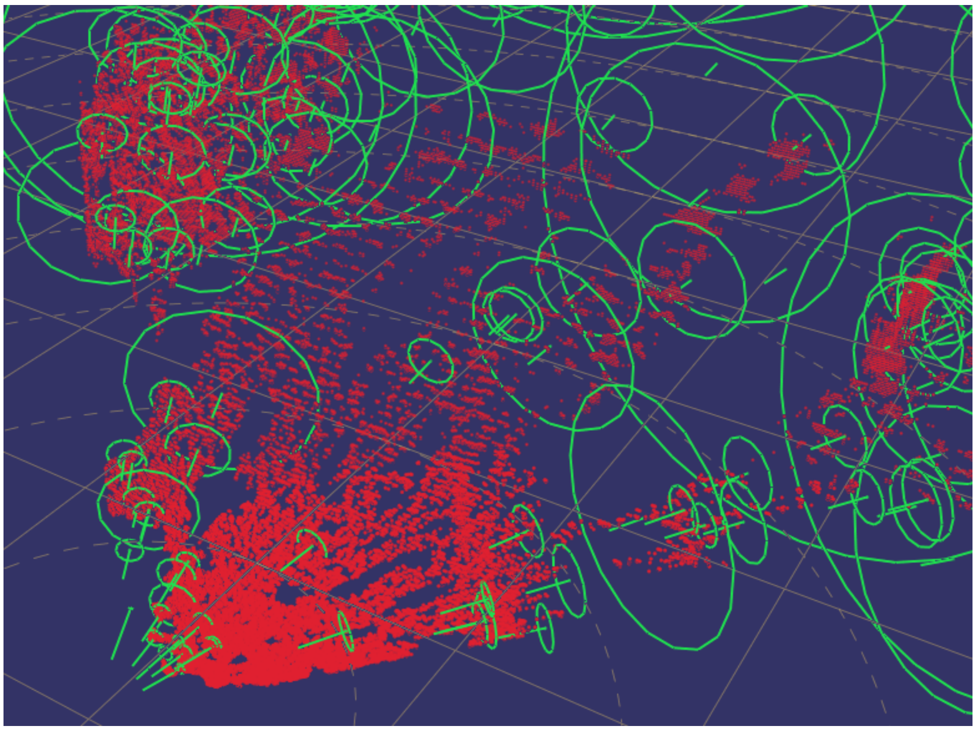


Abbildung 97: *Durch die Kalibrierung der Stereokamera kann aus korrelierten Stereofeature-Paaren eine 3D-Position der* sparse features *errechnet werden (in Grün dargestellt). Die roten Punkte sind durch eine dichte Stereobild-Berechnung erzeugt worden.*

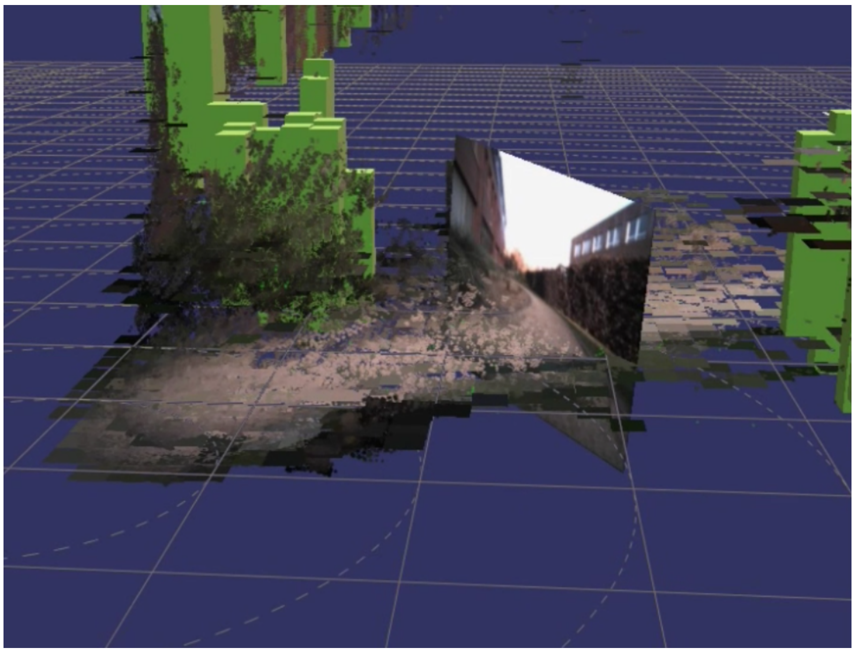


Abbildung 98: *Zusammenführung verschiedener Kartenmodalitäten und Erweiterung um kamerabasierte Texturinformationen*

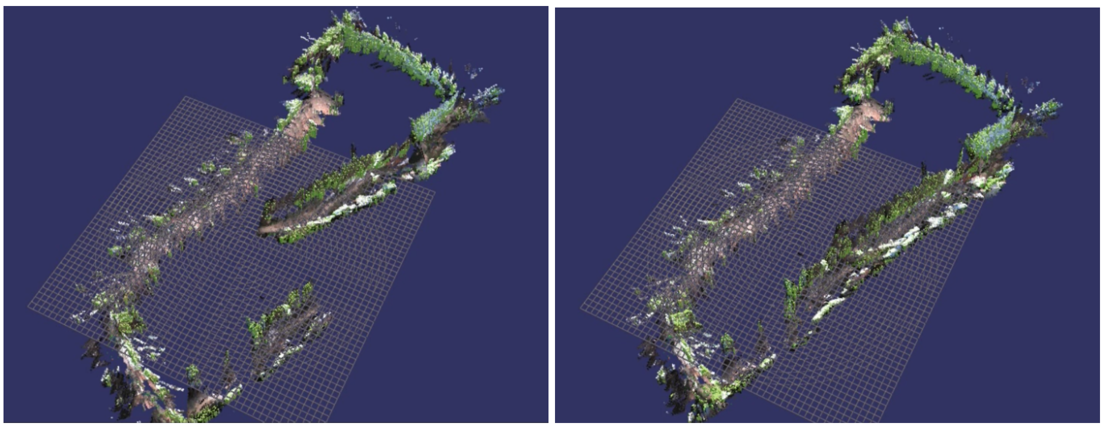


Abbildung 99: *Umrundung des DFKI-Gebäudes vor und nach dem* loop closing*, bei dem ein vorher kartierter Bereich wiedererkannt wird.*

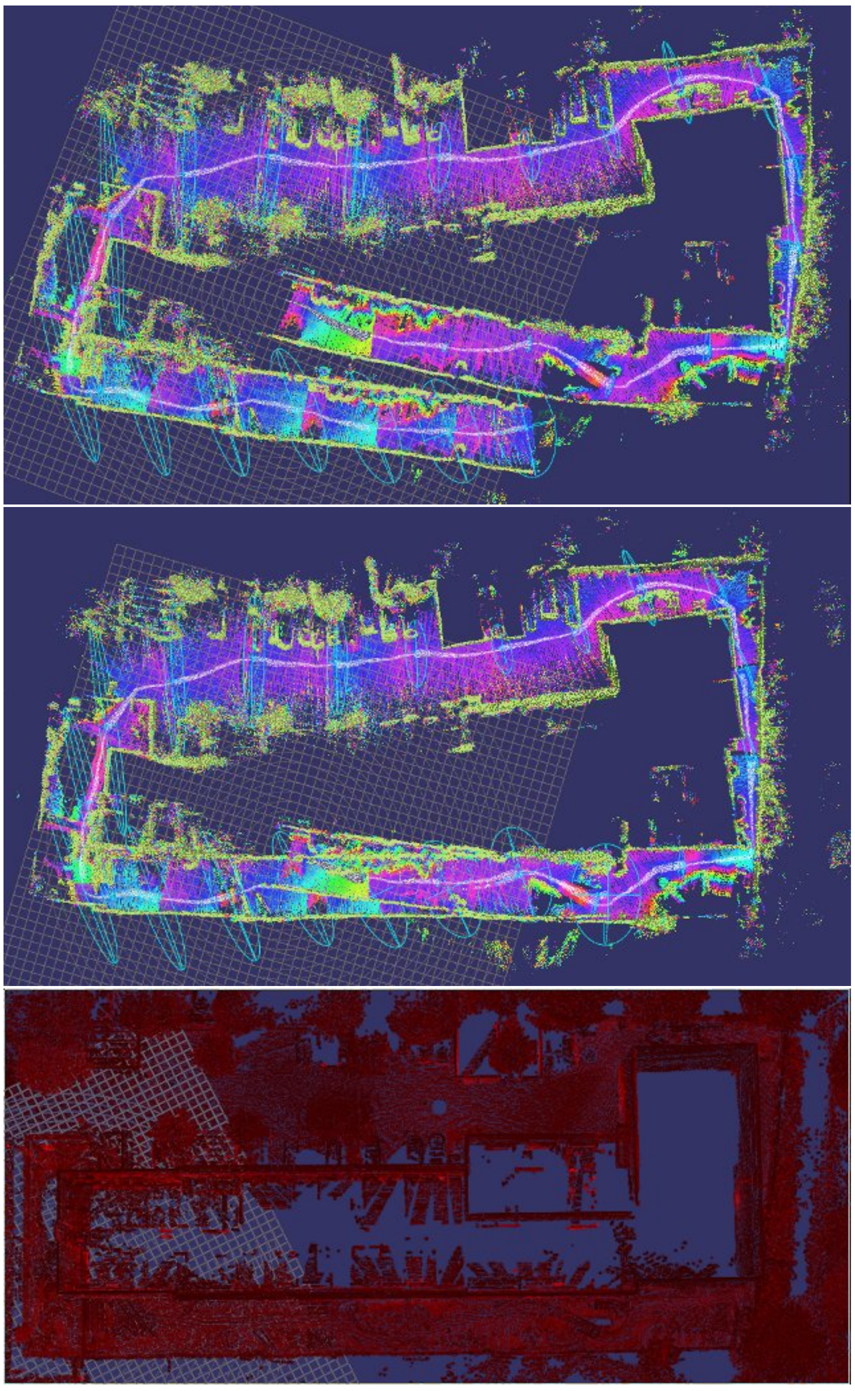


Abbildung 100: *Loop around the DFKI building before (top) and after (center) loop closing. Loop closing is performed by matching stereo image features. When compared to a scan of the building (bottom) shows that the optimized map is locally consistent, but contains distortions.*

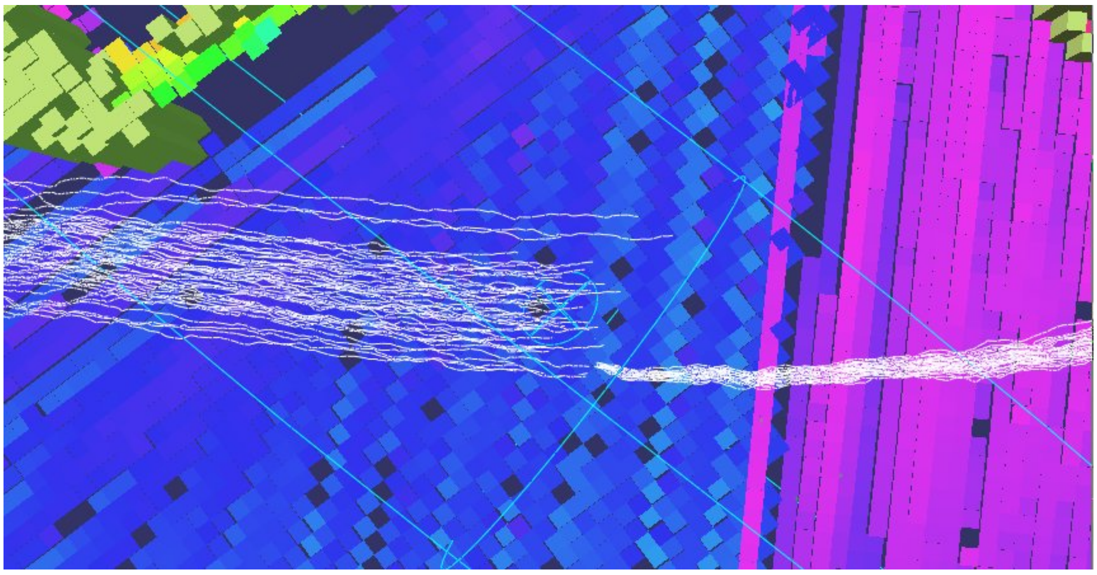


Abbildung 101: *Enlarged top view of a transition between two segments of the globaly optimized map. The particle trajectories of the previous segment (left) are connected to the trajectories on the following segment (right). Map parts are selected from the individual segments after optimization in such a way that the robot path is kept smooth. This ensures smooth map transitions and globally matching maps.*

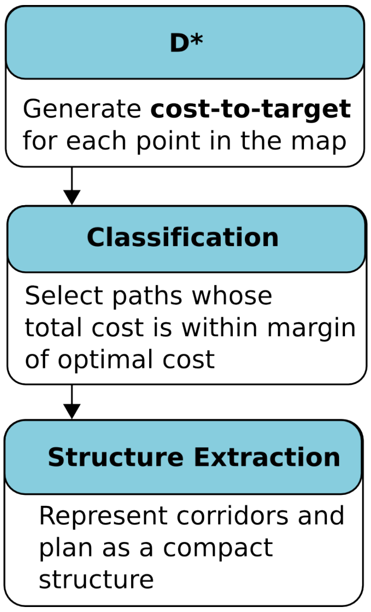

D*
Generate **cost-to-target** for each point in the map
Classification
Select paths whose total cost is within margin of optimal cost
Structure Extraction
Represent corridors and plan as a compact structure

Abbildung 102: *Generelle Funktionsweise der qualitativen iMoby-Pfadplanung*

3.5 AP5000: Autonome Navigation

In diesem Arbeitspaket werden Definition und Implementierung der Autonomie-Aspekte behandelt: Pfadplanung und Pfadausführung. Zunächst wird ein Ansatz zur Pfadplanung entwickelt, der in der Lage ist, Informationen aus der Analyse der Plattform, insbesondere die Wahl der Bewegungsmodalitäten, zu nutzen. Dieser Pfadplanungsansatz wird in das System eingebunden, wodurch die Interaktion von Aufgaben- und Pfadplanung möglich wird (AP5200). Dadurch wird es möglich, Entscheidungsfindungsprozesse in Echtzeit durchzuführen, um Navigationsfehler in Verbindung mit den im AP3300 (Diagnostik) gebauten Bewegungsmodellen und der durch eSLAM verfügbar gemachten Informationen zu beheben. Am Ende des Projektes wird ein echter Missionsplaner in das System eingebunden, um verbesserte Fähigkeiten für die Entscheidungsfindung bereitzustellen (AP5400).

WP5100: Qualitative Pfadplanung

Die Planungsebene eines Robotersystems muss mit verschiedenen Belangen und Detailgraden von Planungsaufgaben umgehen. Auf der höchsten Ebene steht die langfristige, strategische Planung des Systems, die nicht unbedingt lokale Eigenschaften der Umgebung in die Entscheidungsfindung einbezieht. Die Pfadplanung beschäftigt sich damit, auf welchen Wegen sich ein System durch seine Umgebung bewegen kann. Die detaillierte Bewegungsplanung geht direkt auf die lokalen Eigenheiten der Umgebung und des Untergrundes ein. Die Bewertung der besten Strategie erfolgt hierbei meist auf Basis von Kostenfunktionen. Diese lassen sich jedoch oft nur schwer miteinander kombinieren.

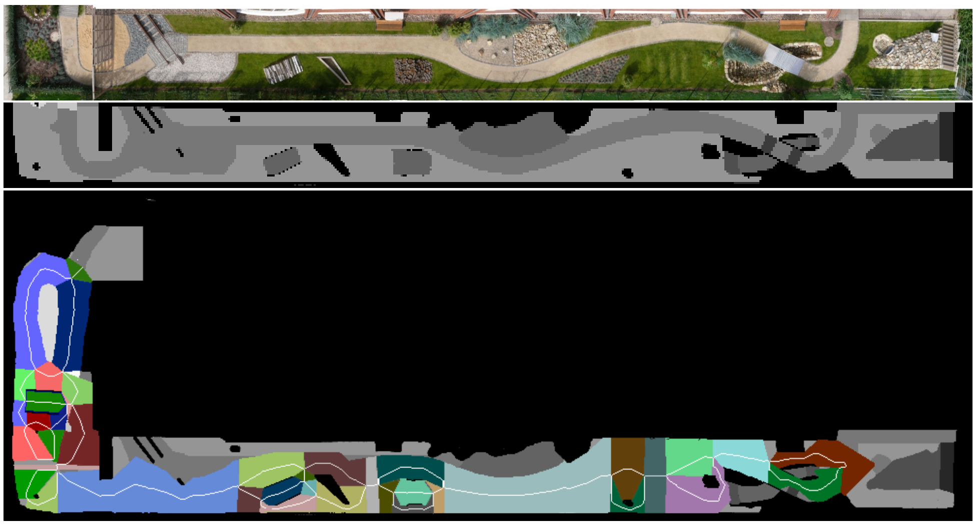


__Abbildung 103:__ Ein Durchlauf des Planungsalgorithmus: Testgelände am DFKI aus der Vogelperspektive, manuell generierte Klassifizierung, Darstellung der Korridore

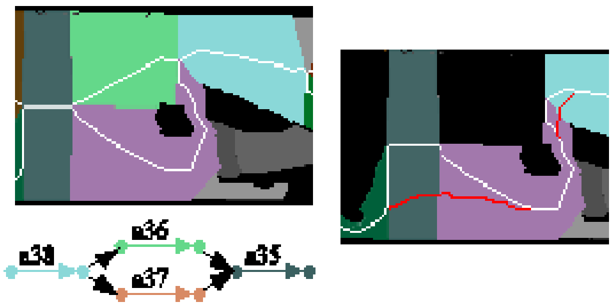


__Abbildung 104:__ Ein Beispiel der topologischen Darstellung, wo alternative Pfade erkennbar sind. Der Roboter kann entweder den Pfad $a_{38} \rightarrow a_{36} \rightarrow a_{35}$ oder den Pfad $a_{38} \rightarrow a_{37} \rightarrow a_{35}$ wählen.

Der im Projektantrag vorgeschlagene Algorithmus zur qualitativen Pfadplanung wurde implementiert und experimentell evaluiert. Die generelle Idee des Ansatzes zeigt Abbildung 102. Der Ansatz ist dabei, dass meistens verschiedene, *qualitativ* vergleichbare Lösungen für ein Problem zur Verfügung stehen. Durch die Angabe eines Korridors von Möglichkeiten kann der Roboter mit Hilfe eines Planmanagementsystems eine geeignete Lösung finden. Abbildung 103 zeigt beispielhaft einen Durchlauf des Planungsalgorithmus.

Ein Artikel zum hier realisierten Ansatz wurde geschrieben und zur Veröffentlichung angenommen.

Eine genauere Darstellung der entwickelten Methoden ist in Sektion 5.10 zur Planer/-Planer integration, sowie Sektion 5.11 zur qualitativen Pfadplanung zu finden.

WP5200: Integration mit ausführendem Aufgabenplaner

In diesem Arbeitspaket wurde das *Orocos Component Layer* an das Plan-Management angebunden. Das Ziel hierbei war die einfache Spezifizierung und Manipulation des

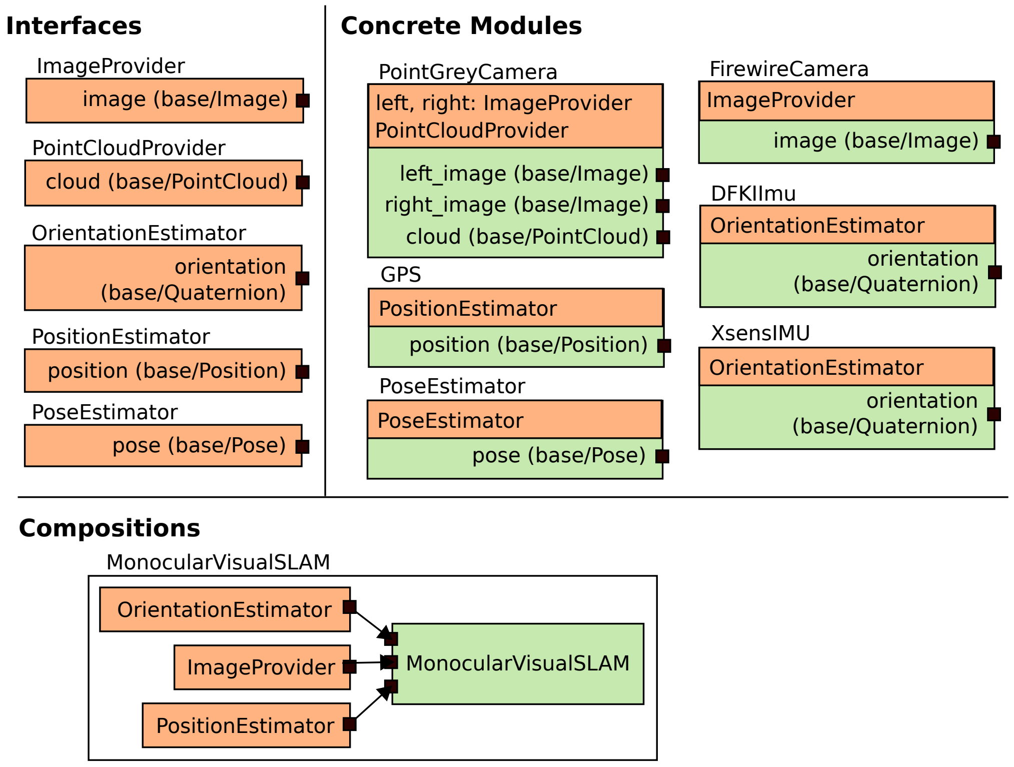


Abbildung 105: *Spezifikation von Interface, Komponente und Komposition.*

Systems. Modellbasierte Integration erlaubt die Validierung des Komponentennetzwerks, bevor die Software auf dem Roboter genutzt wird, was ein wichtiger Schritt zur Fehlervermeidung ist. Wichtig hierbei ist das Interface. Es stellt die Generalisierung eines einzelnen Komponentenblocks dar und erlaubt es, die funktionale Schicht unabhängig von den konkreten internen Funktionen einzelner Module aufzubauen. Kompositionen fassen Interface und konkrete Module zusammen und bilden die tatsächliche Funktionalität. In Abb. 105 ist die Komposition *MonocularVisualSLAM* dargestellt, welche einen Orientierungs- und einen Positionsschätzer sowie als Eingabe eine Bildquelle verwendet. Durch die Nutzung von Interfaces kann diese Komposition ohne größere Anpassungen auch auf andere Systeme übertragen werden. Bei der Ausführung werden einzelne Module (Abb. 106) ausgewählt, die die benötigten Interfaces implementieren. Auf Basis dieser Information wird eine Abhängigkeitsstruktur im Plan-Manager erstellt (hier nicht dargestellt), welche die Überwachung und Adaption zur Laufzeit möglich macht.

Es wurde eine Fehlerbehandlung für von der Hardware gemeldete Fehler implementiert. (H-Brücken-Module, Sensoren, etc.)

Genauere information zu Spezifikation und Online-Anpassung der Komponentenebene können in Sektion 5.12 nachgelesen werden.

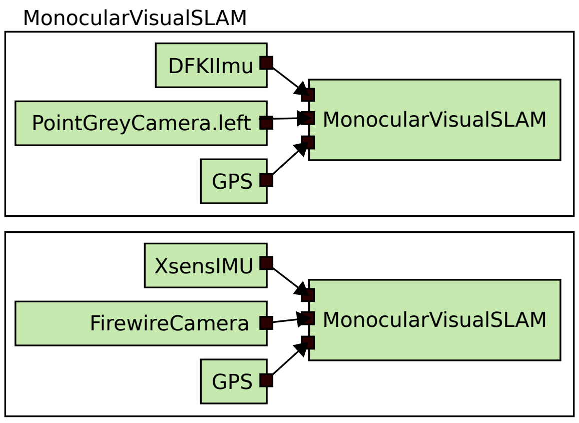


Abbildung 106: *Zur Laufzeit werden Kompositionen durch Selektion konkreter Module instanziiert.*

WP5300: Fehlerbehandlung in der Navigation

Fehlermeldungen auf der hardwarenahen Softwareschicht (low-level) werden nun erfasst und abgefangen. Es können sowohl sensorerzeugte Fehlermeldungen (Sensorfehlfunktionen) als auch sicherheitsrelevante Fehler (Verbindungsverlust zur *Operator Control Unit*) detektiert werden. Eine Fehlertoleranz ist gegeben, d.h., das System kann Daten fehlfunktionierender redundanter Sensoren ignorieren und die Navigation nach einem Verbindungsabbruch und einem erneuten Verbindungsaufbau zur *Operator Control Unit* wiederaufnehmen.

Ein Algorithmus zur lokalen Hindernisvermeidung wurde implementiert.

Es wurde eine statistische Analyse durchgeführt, um ein Fehlermodell für das *Iterative-Closest-Point*-Verfahren (ICP) aufzustellen. Diese Analyse dient einem doppelten Zweck: Auf der einen Seite sollen fehlerhafte Ergebnisse des Lokalisierungsalgorithmus erkannt werden, auf der anderen Seite soll das System in der Lage sein, nach Lokalisierungsfehlern zu einer korrekten Positionshypothese zurückzufinden.

Algorithmen zur lokalen Bewegungsplanung wurden implementiert, die es erlauben, mit Fehlern in der Karte umzugehen. Hindernissen, die nicht in der globalen Karte enthalten sind, kann das System nun lokal ausweichen. (Abbildung 107)

Es wurde ein Verfahren zur Verifikation der Ergebnisse des ICP-Moduls entwickelt.

Das ICP-Modul wird genutzt, um die eigene Position in Relation zu einer Karte zu berechnen. Dazu wird eine Reihe von Laserscans verschoben und gedreht, bis sie mit minimiertem Fehler in die Karte passt. Dabei kann der ICP-Algorithmus allerdings gegen ein lokales Minimum konvergieren. Um zu prüfen, oder dies der Fall ist, wurde eine *Support Vector Machine* (SVM) trainiert, die Güte der Einpassung der Laserscans in die Karte zu ermitteln.

Auf Basis der ICP-Verifikation wurde ein Verfahren zur Relokalisierung entwickelt. Dabei wird zufällig über die gesamte Karte verteilt ein 3D-Scan mittels ICP eingepasst. Wird das Ergebnis von der SVM als geeignet klassifiziert, so werden anschließend die

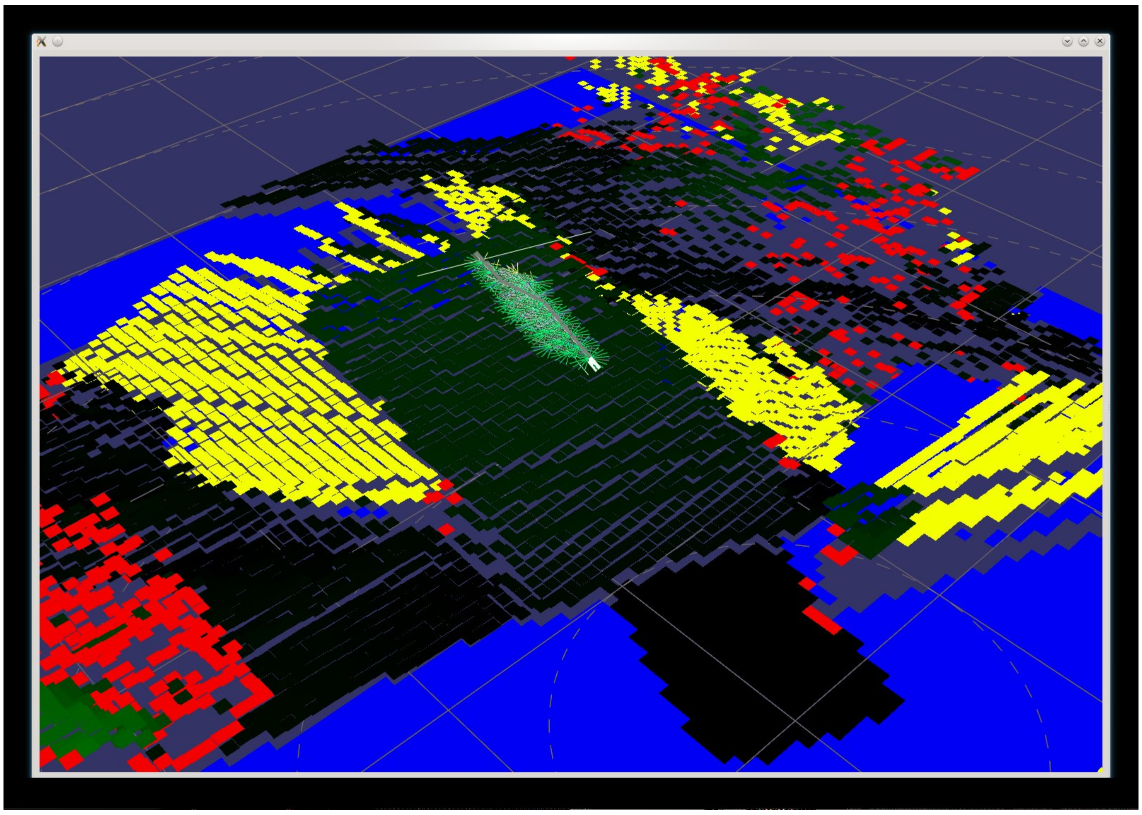


Abbildung 107: *Lokale Pfadplanung*

Stabilität der Konvergenz und die ermittelte Position getestet. Fallen diese Tests positiv aus, wird davon ausgegangen, dass die Relokalisierung erfolgreich war.

Außerdem wurde ein Verfahren zu Überwachung der Navigationsschicht entwickelt. Eines der Probleme bei der Überwachung besteht darin, zu entscheiden, ab welcher Unsicherheit der Positionsschätzung von einem Fehler ausgegangen werden muss. Um dieses Problem zu lösen, wird ausgenutzt, dass die Navigationsschicht hier nicht auf Basis von Trajektorien, sondern von Korridoren arbeitet. Ein Korridor ist dabei als eine Fläche definiert, auf der der Roboter sicher navigieren kann. Liegt die Positionsschätzung innerhalb eines Korridors, wird angenommen, dass die Navigationsschicht korrekt arbeitet. Erst wenn die geschätzte Position den Korridor verlässt, liegt ein Fehlerfall vor. Dann wird eine Relokalisierung ausgeführt.

WP5400: Integration von Missions- und Aufgabenplanung

Die Integration von Techniken zur Aufgabenplanung in das Planmanagementverfahren, das entwickelt wurde, wurde untersucht. Das Ziel war dabei die Erhöhung der Robustheit der erzeugten Pläne und die Möglichkeit, aufwändigere Aufgabenplaner – wie z.B. High-Level-Missionsplaner von Bodenkontrollstationen – zu integrieren.

Zur Durchführung von WP5400 wurde eine Übersicht über existierende und verfügbare Software zur Missions- und Aufgabenplanung erstellt. Anschließend wurden die verfügbaren Softwarepakete in Hinsicht auf ihre Einsetzbarkeit innerhalb des Projektes evaluiert. Die Evaluation ergab, dass zwei Missions- und Aufgabenplaner existieren, die prinzipiell für das Projekt geeignet sind.

Der erste ist der Europa2-Planer, der von Nasa Ames veröffentlicht wurde. Hierbei handelt es sich um ein Planungssystem, das für Applikationen aus dem Weltraumbereich entwickelt wurde. Da in solchen Applikationen meist Ressourcenknappheit herrscht, hat der Planer einen starken Fokus auf Zeit- und Ressourcenplanung. Dies ist eine für das Projekt interessante Charakteristik, allerdings ist der Europa2-Planer gleichzeitig sehr komplex und aufwändig in das bestehende Softwaresystem zu integrieren.

Beim zweiten evaluierten Planer handelt es sich um ein generisches Planungswerkzeug, das von der französischen Luftfahrtforschungsanstalt (ONERA) entwickelt wurde. Der Planer stellt dabei Möglichkeiten zur klassischen sowie zur wahrscheinlichkeitsbasierten Planung bereit. Des Weiteren ist der Planer mit erheblich weniger Aufwand in das bestehende Softwaresystem zu integrieren, da er auf dem gleichen Framework aufsetzt.

In Diskussion mit den Projekten RIMRES und IMPERA wurde PDDL als gemeinsame Grundlage für eine Zusammenarbeit festgelegt. Diese Entscheidung basiert darauf, dass es eine Reihe von Planungstools gibt, die diese Sprache verstehen, sowie auf der Tatsache, dass das IMPERA-Projekt diese Sprache nutzen möchte. In weiterer Zusammenarbeit ist das generelle Interface zum Planer entwickelt worden. Außerdem ist evaluiert worden, wie die effektive Nutzbarkeit der entwickelten Interfaces verifiziert werden kann. Da das RIMRES-Projekt und iMoby dieselbe Planmanagementschicht verwenden, und in iMoby selbst kein ausreichend komplexes Szenario vorliegt, wurde die Verifikation in Kollaboration mit dem RIMRES-Projekt durchgeführt.

Der eSLAM-Algorithmus wurde in die Regelschleife des Roboters integriert. Dadurch wird es möglich, in unbekanntem Gelände zu navigieren und somit den dritten Meilenstein zu absolvieren.

Der Planer des Roboters ist wie folgt aufgebaut:

Der Roboter bekommt ein Kommando in der Form einer relative Positionsangabe. Daraufhin wird die aktuelle Umgebungskarte des Roboters von der eSLAM-Komponente abgefragt und an die Komponente für die Befahrbarkeitsanalyse weitergeleitet. Diese produziert eine annotierte Umgebungskarte, in der eine Abschätzung über die Befahrbarkeit der Umgebung hinterlegt ist.

Diese Karte wird von der Komponente zur Korridorplanung genutzt, um Korridore von gleich gut befahrbarem Gelände zu bilden, hierbei wird zwischen verschiedenen Abstufungen der Befahrbarkeit sowie unbekanntem Gelände unterschieden. Den Korridoren werden anschließend spezielle Fahrverhalten des Roboters, die auf dem jeweiligem Gelände am besten funktionieren, zugeordnet.

Hierdurch ergibt sich folgende Sequenz:

- Fahre auf gut befahrbarem Gelände mit hoher Geschwindigkeit bis zur Position X.
- Fahre dann auf dem schlechter befahrbaren Gelände mit einem vorsichtigen Verhalten bis Y.
- Im anschließenden unbekannten Gelände nutze das Explorationsverhalten um das Ziel zu erreichen.

Tritt während der Ausführung der Sequenz ein Fehler auf, wird die aktuelle Sequenz verworfen und der gesamte Planungsvorgang neu ausgeführt.

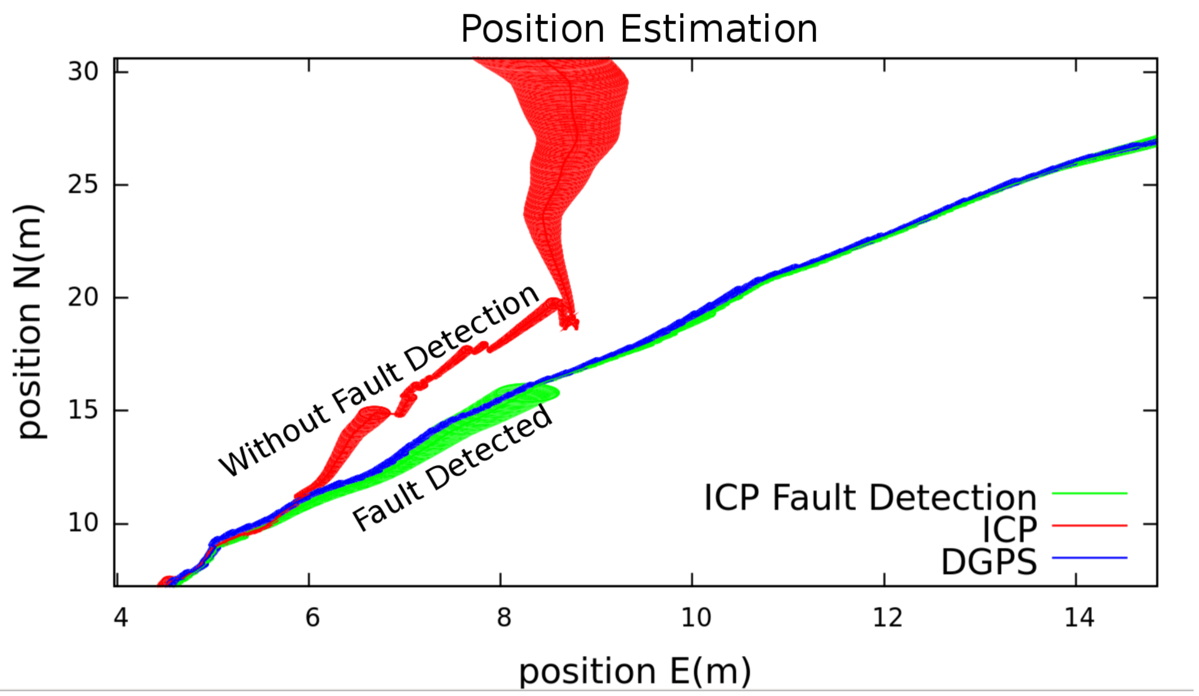


Abbildung 108: *Posenschätzung mit Fehlerdetektion*

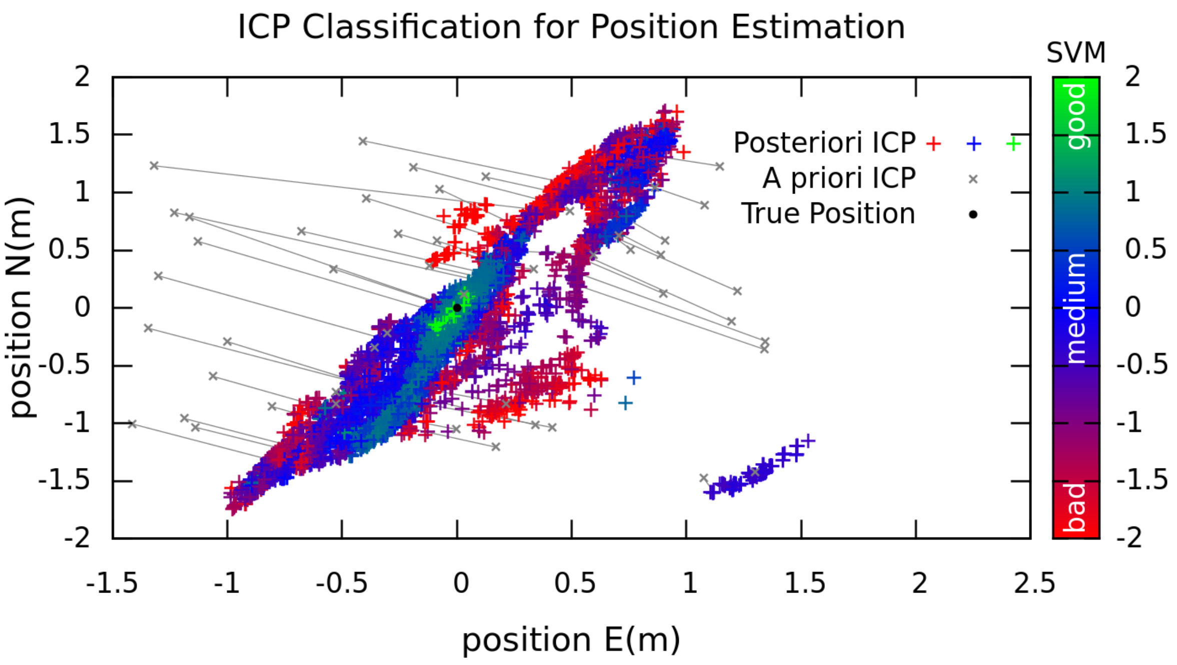


Abbildung 109: *Relokalisierung mittels ICP*

4 Fazit und Ausblick

In einem zusammenfassenden Rückblick kann festgestellt werden, dass sowohl die technischen als auch die nicht-technischen Ziele des Projektes erfüllt wurden.

Es konnte im Projekt gezeigt werden, dass die Asguard-Mobilitätsplatform auch zur Nutzung in autonomen Szenarien geeignet ist. Weiterhin wurde für die Asguard-Plattform ein weiterer Iterationsschritt durchgeführt, der eine weitere Verbesserung der Technologie im Bezug auf Konzept und Materialien mit sich brachte. Die Einbettung der Sensoren in die Strukturelemente ist durchgeführt worden, hat sich aber nicht so vielversprechend wie erhofft gezeigt. Der Mehraufwand und Nutzen hat sich gegenüber von der Software geforderten sensorischen Informationen für ein hybrides System als nicht ausreichend vorteilhaft herausgestellt. Weitherhin wurde die Idee eines passiv gedämpften Sensorkopfes nicht in das finale System eingebunden. Hier waren die Ergebnisse auch mit ungedämpften Sensorköpfen ausreichend für die Anforderungen.

Die Arbeiten auf dem Gebiet der Einbeziehung von körperbezogenen Daten für die Navigation hat gute Ergebnisse hervorgebracht. Es ist gezeigt worden, dass es möglich ist, sich auch ohne visuelle Sensoren in einer a-priori-Karte bis auf 1.5 m genau zu lokalisieren. Im Verbund mit visueller Sensorik konnte auch die Simultane Lokalisierung und Kartenbildung (SLAM) gezeigt werden. Es ist außerdem eine Methode entwickelt worden, die es möglich macht, graphenbasierte Ansätze mit Partikelfiltern für SLAM-Lösungen einzusetzen. Die SLAM-Ansätze konnten auf dem Asguard-System erfolgreich verifiziert werden.

Im Bereich der Softwaremodule für autonome Navigation ist eine Reihe von Tools für die effektivere Nutzung des Orocos Realtime Toolkit erzeugt worden. Weiterhin sind Softwaretools erzeugt worden, die eine Ausführungsschicht darstellen. Damit ist es möglich, Konfigurationen von Modulen zur Laufzeit zu überwachen und in Fehlersituationen anzupassen. Dies macht eine effektivere Entwicklung von Softwaremodulen möglich, da Fehlersituationen in einer höheren Schicht abgehandelt werden. Weiterhin ist eine Reihe von Softwaremodulen für die Navigation entwickelt worden und am Asguard-System demonstriert worden. Es ist möglich, in unbekanntem Terrain autonom einen Wegpunkt anzufahren und wieder zum Ausgangspunkt zurückzukehren. Dabei ist auch ein "near-optimal" Planer entwickelt worden, der eine Verbindung zwischen der symbolischen und der Kartenschicht schafft.

Das DFKI hat durch das Projekt iMoby weitreichende praktische Erfahrungen in der autonomen Navigation von Außenräumen sammeln und demonstrieren können. Die innovativen Themen, die dabei angegangen worden sind, konnten auf zahlreichen Fachkonferenzen und in Fachjournalen präsentiert werden. Weiterhin sind die Projektergebnisse auch durch Medien und Fachbesucher beim DFKI erfolgreich verbreitet worden.

5 Appendix

Classifying Autonomous Systems
Intelligent Mobility Report D2.1

DFKI Robotics Innovation Center
Förderkennzeichen: 50RA0907
Projektleiter: Sylvain Joyeux
Laufzeit: 1.4.2009 – 31.03.2012

1. June 2010

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

German Research Center for Artificial Intelligence
Robotics Innovation Center
Prof. Dr. Kirchner
Robert-Hooke-Str. 5
28359 Bremen, Germany

Contents

1 Introduction **3**
1.1 Defining Autonomy . . . 3
1.2 Why do we need autonomy levels ? . . . 4

2 Review **4**
2.1 Existing schemes . . . 4
2.2 Existing Systems . . . 6

3 Method **7**
3.1 Designing and Interpreting Autonomy Levels . . . 7
3.2 Proposed Aspects . . . 7

4 Classification of Existing Systems **9**

5 Testing Autonomy **9**

6 Conclusion **11**

Abstract

Robotic exploration missions to celestial bodies in our solar system provide us with the necessary knowledge to prepare for human presence on the moon or other planets, and could have the ability to give vital information about our own origins. Direct control over the mobile robot systems is only possible in particular cases, and the ability of the systems to take decisions autonomously increased in importance with more complex mission scenarios and targets further away. The ability to classify and benchmark autonomy is important for the design, evaluation and comparison of such systems. One of the problems is that existing schemes to classify autonomy are either too generic, or don't fit the requirements for mobile space exploration systems. The existing schemes for space engineering and other domains are discussed. Further, existing and planned robotic exploration missions are analysed towards their autonomy content. A new scheme for the classification of autonomy in the given domain is drafted and the existing systems classified. The evaluation of performance levels for autonomy is discussed, and initial ideas on how this could be feasible given. The paper is considered a starting point for further work in the area of classification and benchmarking of autonomy for mobile space exploration robots.

1 Introduction

The exploration of unknown places has always been an important part of our history. Traditionally this was the domain of brave man and woman often risking their lives to venture into the unknown. Today, we have the possibility to send machines in our stead, taking the physical risk for us.

For an exploration mission to be successful, we have to acquire new knowledge from the places we visit, and relate it to the things that are familiar to us. Judging which actions to take in order to gain the most insight is often a hard task. Up until now, human operators were responsible to handle that part of the mission. An alternative solution is to let the robotic system decide for itself what courses of action to take, in order to extend the information known about the environment it is in. In such a scenario only information that constitutes new knowledge is transmitted back to the operator. As exploration missions go further away from Earth, this alternative solution might become more important to pursue.

There are usually two parts in any exploration scenario. On the one hand, one wants to verify scientific assumptions, i.e. find something particular that is expected by the scientists. On the other hand, it is very important that anything that is *not* expected gets investigated, as it may represent very valuable information. Autonomous information systems have a hard time here, since they are commonly programmed to cope with foreseen situations. The capability to be truly flexible is one of the prime examples where humans will be far superior to robots for a long time to come.

In order to drive the development of autonomous systems in the context of space missions, it is necessary to specify the level of autonomy which is required for the task and feasible to implement with the given resources. Although classification of autonomy levels are given for certain domains (e.g. ECSS Standard for European Space Autonomy), no definition has been provided for the domain of robotic exploration systems, yet.

In this paper we seek to provide a starting point for such a taxonomy of autonomy levels and give information on how to potentially benchmark and test them. Further, we use our scheme to classify robotic exploration systems for the space domain, and compare it to the existing classification schemes. We show that the existing schemes are not sufficient to represent the various aspects of exploration robot autonomy and that the proposed scheme is more suited for this. Finally, the possibility to test and/or benchmark such a scheme is also investigated.

1.1 Defining Autonomy

Numerous papers tried to define what autonomy *means*. What this section will do is first define a few terms, that are critical to explain the rationale underlying the definition we picked in this work.

Task, Environment and Scenario

Task an abstract action which represents what the system should do. This definition will most often be general, i.e. can be reused across a wide range of systems and situations (e.g. Find evidence of water, map geological features, or explore cave system).

Environment a description of the *context* in which the robot will operate and achieve the assigned task. In contrast to tasks, environment definitions are very specific to a mission (e.g., crater rim at lunar south pole for the NEXT-LL mission).

Scenario a scenario is an association between one or multiple tasks and an environment.

What underlies the work presented in this paper is the belief that autonomy ratings should always be put in context, i.e. should always be considered with respect to a scenario. The rationale behind that is that a system that is well adapted to a particular environment (for instance lighting conditions on the Moon's south pole) might have poor autonomy on earth and high autonomy during its mission. In an identical manner, a system that is well suited for some tasks might not be usable at all for different ones. Take for instance exploration and infrastructure building tasks. The former requires the system to be able to handle the *unknown* (i.e. gather information and form a "big picture" of what is available), while the latter will probably provide an already good picture of the environment, probably in the form of a map. Some systems will probably be specialized in the first task, while others will be in the second task. They will both need specific sets of skills, some of them being common but some of them different.

Moreover, the definition of autonomy (and autonomy levels) should be general enough not to request a particular system architecture and/or solution. In other words, how autonomy and autonomy levels are defined should not constrain the space of solutions. It should only provide a standardized way to describe these solutions.

We therefore reduce the definition of autonomy to a minimum:

> Autonomy is the ability for a system to achieve certain tasks in a specific environment by itself

It should be noted, that our definition of autonomy does not include the performance of the system. A completely autonomous system is not guaranteed to perform well under any given performance metric, and might perform much worse than a system with a lower level of autonomy.

1.2 Why do we need autonomy levels ?

From a practical point of view, a classification of autonomy has two main uses. First, the ability to describe autonomous capabilities to "upper management", i.e. in the context of development programs, in order to plan technological developments. Second, the ability to compare systems with each other, and to specify requirements in the frame of specific missions.

From a general point of view, it allows to classify systems through their capabilities.

2 Review

2.1 Existing schemes

In all existing schemes that we know of, it is acknowledged that autonomy cannot be rated in one global scale. This is because there are different aspects to autonomy, each aspect being more or less independent of each other. It is therefore possible to improve an aspect independently of the other ones. To represent this, the schemes we reviewed split the autonomy classification they propose along various axis that are then split into fine-grained levels.

In this section, we will focus on three different schemes that we believe are representative of autonomy classification schemes. First, the ECSS standard ECSS-E-ST-70-11C for space segment operability, which is the European standard for autonomy levels, provides a very simple qualitative assessment of autonomy. Second, the ALFUS framework, which provides a generic *quantitative* scheme for measuring autonomy. Finally, a scheme that has been developed by the US Air Force, targeted at UAV autonomy.

The European space standard body, ECSS, splits the autonomy levels into two aspects: “Mission execution” and “Fault Detection, Isolation and Recovery” (FDIR). The mission execution aspect is very classical, from direct ground control in E1 to “goal-oriented mission operations on-board” in E4 [ECS08]. The FDIR aspect has only two levels: the F1 level in which the system is able to put itself in a safe condition upon failures, and the F2 level in which it can reestablish normal operations by himself. We unfortunately did not find the NASA equivalent of the ECSS standard for space operations.

The ALFUS framework [HMA07], in contrast, aims at providing metrics that can provide a quantitative assessment of autonomy, mainly in the domain of ground systems. They do so along three axis. First, the *Mission Complexity* aspect measures how complex the mission is from the point of view of the system’s decision-making component. It includes measures such as the number of interdependent choices that the system must do to achieve its goal. Second, the *Human Independence* aspect quantifies how much the system has to interact with its operators to achieve its mission. Finally, the *Environmental Complexity* aspect measures how difficult the environment is for a particular system.

Finally, the scheme in [Clo02] splits into four axis: Perception/Situation Awareness, Analysis/Coordination, Decision Making and Communication. One very important point that the authors make is that, while the general axis can be reused, the definition of each levels in the axis will – and should – be interpreted differently in different scenarios/domains. This is illustrated by the fact that general names are used for each of the levels (e.g. “Fault/Event Adaptive Vehicle” for level 5), while the detailed description is very specific to the author’s domain of interest (UAVs).

In our opinion, none of these schemes apply to the domain of space exploration systems, the domain we focus on.

First and foremost, we believe that a more detailed scheme than what ECSS proposes must be available for future space missions. Indeed, autonomy is a critical technology for deep space exploration, and a detailed scheme will provide decision-makers with the basic understanding of what an autonomous exploration system is made of in the context of exploration missions.

Second, quantitative metrics should be limited to very specific solutions. One trait of the ALFUS scheme is that the metrics are applicable only if one assumes that the evaluated solutions are based on some specific architecture. For instance, most of the decision-making metrics assume that planning and task decomposition schemes are at the root of the autonomous system. We, in contrast, believe that an autonomy description scheme should not constrain the solutions it can be applied on. Moreover, the ALFUS scheme advocates that the various metrics can be combined into a weighted sum to provide an overall autonomy value. We do not believe that such a quantitative combination is meaningful in the evaluation of autonomy.

Finally, we actually agree with most of the analysis included in the UAV classification scheme: that, to be useful in operational context, an autonomy classification needs to focus – at least partially – on the actual domain. As we will see later, we did need to take into account some aspects that are relevant for space exploration. Namely, the fact that most of the system’s environment is unknown (contrary to having fine-grained and reliable maps and localization systems as it is for instance the case on urban environments on Earth).

2.2 Existing Systems

The evaluation of existing systems is limited to lunar or planetary exploration missions that include a mobile robotic element. So far only two celestial bodies in the solar system (excluding of course earth) have been visited by mobile robots: the moon and mars.

Lunokhod The Russian Lunokhod I (1970) and II (1973) rovers [Bri07] where the first mobile robotic systems to operate on a celestial body. Both rovers where fully teleoperated. Video data was relayed back to the operators which directly controlled the movements of the vehicle. Lunokhod II managed to travel a distance of 37 km over in a time period about 4 months.

Sojourner was the first mobile exploration robot to successfully operate on another planet. In 1997 the rover operated for 83 days on the Martian surface and examined rock samples in a radius of 10 m around the lander. Because of the large time delay, the operators where only able to communicate with the rover once per sol (24.6 h). Because of this, sojourner already included autonomous traits, like simple waypoint navigation, including a hazard avoidance behaviour [Mat98]. Hazard detection was performed using an array of laser stripes for proximity sensing, and an accelerometer for detection of hazardous slopes and a bumper sensor as the last resort.

MER The Mars Exploration Rovers (MER) Spirit and Opportunity are continuing to explore the Martian surface since 2004. The rovers include software autonomous capabilities in the area of navigation, science instrument placement, resource management and science data gathering [Mai06]. Like the Sojourner, the MER operators only had a single communication window per sol. Autonomous navigation capabilities and hazard avoidance allowed the operators to give movement commands that go beyond the range at which obstacles can safely be identified and thus would have to rely on the on-board autonomy to perform this feature. Currently Opportunity is still operational after 6 years of mission time, and managed to cover a distance of more than 20 km in that time. The autonomy levels of the MER were increased through software update in the course of the mission time, which added a global path planner, target tracking and positioning as well as science event detection capabilities.

ExoMars and MSL Exomars and Mars Science Laboratory are two missions currently in the planning by ESA and NASA. Although not finally fixed, the autonomous capabilities of the two systems are expected to be higher compared to the MER rovers. Both navigation as well as science target selection abilities should be improved compared to the MER rovers. Work has also been performed on improved adaptive capabilities of the systems, like for example the ability to predict wheel slip based on past experience [BMH08].

MSR The next step after MSL and ExoMars is considered to be a Mars Sample Return Mission (MSR) which aims at collecting rock samples to be delivered to an ascent vehicle. Due to limitations in temperature cycling of the fuel, the ascent vehicle would have to lift off again within a years time, and thus limits the overall mission time. The mobile robot has to collect as many samples from different locations as possible within the given the time frame. This can only be achieved with a high level of autonomy, so that the rover does not have to rely on commands from the operators for travelling [BMH08].

NEXT-LL NEXT Lunar Lander is an activity currently pursued at ESA which aims at a precursor mission to human presence on the moon, and might contain a mobile robotic element for exploration. The requirements for autonomy on this mission are limited, since the communication latency is within range for direct teleoperation. Nevertheless, for technology demonstration purposes, and to improve reliability, the system could potentially have some autonomous abilities available, especially in the area of diagnosis and fault recovery.

3 Method

3.1 Designing and Interpreting Autonomy Levels

One very important aspect, when assessing the autonomy of a system, is that measuring the capabilities of single subsystems does in no way allow to infer the autonomy of the whole system. This is so for two reasons:

- autonomy can be created by having the components interact with each other (i.e. by having a system). This is seen in cognitive architectures, where learning algorithms can end up having a deep situation awareness while components – in themselves – have little understanding of the overall goal and mission
- conversely, bad interactions between components can *degrade* the overall system autonomy.

The proposed scheme gives a qualitative assessment of a system autonomy in a given scenario. In no way do we aim at providing a quantitative measure of autonomy.

The proposed scheme does not have any assumption on the underlying architecture, which is a problem we identified in the ALFUS scheme.

Finally, one issue have arisen when we designed this scheme. We first tried to represent these levels on linear scales, conveying that to get a level 5 on one aspect, one would have first to first obtain levels 1 to 4. However, it ended up being difficult to obtain for the decision-making aspect.

To solve this problem, instead of adding new aspects – which would have added complexity to the description – we kept the representation of each aspect linear. The levels therefore might not represent dependencies between more or less advanced capabilities, but instead a level of importance: we felt that, for instance, the ability to reason about time and resources (level 3 in decision-making) is more important than the ability to adapt to changing performance (level 6), and would therefore need to be implemented before. However, it should be stressed that it is our own opinion and that in some scenarios adaptation might be more important than complex reasoning capabilities.

3.2 Proposed Aspects

We propose to separate the description of autonomy into four different aspects (or axis):

Information Interpretation this is obviously critical in exploration scenarios. It describes the reasoning capabilities the system has with respect to the information that it has, and the information that is present in its environment.
The proposed levels are on Table 1

Decision-Making the mechanisms that allows the system to choose its course of action in order to achieve its goal. *Decision* in an autonomous system is a difficult term to define. What we propose is to limit it to the ability of the system to choose. I.e., given a scenario, if there are multiple ways to achieve the task, how is the system able to pick one way which is viable.
The proposed levels are on Table 2

Diagnostics assessment of the status of its different subsystems (software and hardware), and of how these subsystems interact together.
The proposed levels are on Table 3

Fault Recovery “Fault” is a bit misused here. What this aspect covers are all the mechanisms that allow the system to cope with the changes to its parts (including malfunctions).
The proposed levels are on Table 4

Level	Capability
1	Simple sensor processing without interpretation. Example: video camera with preprocessing, which simply stores the images and sends them to Earth.
2	Information assessment: can evaluate how important its sensor data is. An example is the event-based data gathering on the MERs, where data is stored only if the sensor information matches specific triggers
3	Is able to classify expected from unexpected. Most classifiers can tell how well certain samples fit into the classes they know. This capability extends on that by having the classifiers be able to announce that a data sample probably does not fit into the classifier's models
4	Model parameter correction: the classifiers can modify the number of classes and/or the importance of each class
5	Model structure correction: adapt the way classification itself is done, for instance by changing the type of classification itself.

Table 1: *Proposed levels for the* ***Information Interpretation*** *aspect*

Level	Capability
1	No decision-making capability. Executes prepared sequences of actions.
2	The system is able to decide its course of actions, but only reasons on what needs to be done (i.e. no reasoning on resource and/or time constraints)
3	Takes resources and/or time constraints into consideration
4	Can reason on how information affects the system. The system can represent the impact its decisions has on the performance of its subsystems (localization, map-building, . . .), and use that representation to make its choices.
5	Can evaluate the exactness of its own decisions. Given a certain information (and the assessment on this information), it is able to evaluate its decisions and provide a metric – qualitative or quantitative – that represents how sure it is that its decision is "the right one"
6	Adapts the reward / cost representation used for its decisions based on prior experience
7	Adapts the models used for reasoning to either fix it (corrections) or improve it. Example: optimizing dependencies between tasks in planning

Table 2: *Proposed levels for the* ***Decision-Making*** *aspect*

Level	Capability
1	No diagnostics
2	Single component failure detection
3	Subsystem failure detection
4	Detection of performance degradation
5	Detection of undesired behaviour. Example: detects temporal inconsistency in the actions: detects that the robot's plan execution system is stuck in a loop.
6	Adaptation to changing performance. During the system's lifetime, some components/hardware will probably evolve. The diagnostics routines that evaluate these components should therefore also be modified to take into account this evolution.

Table 3: *Proposed levels for the* ***Diagnostics*** *aspect*

Level	Capability
1	Fail-safe capability
2	Single-component redundancy
3	System reconfiguration in case a fault is detected. Example: the system is able to take into account that some sensor or subsystem is not available anymore, and still be functional in a degraded way.
4	Adaptation to changing performance. During the system's lifetime, some components/hardware will probably evolve. This obviously has an effect on the ability of the system to act. This capability will take these changes into account to both act as well as possible given the changed performance, and so that decision-making aspect can still operate.

Table 4: *Proposed levels for the* ***Fault Recovery*** *aspect*

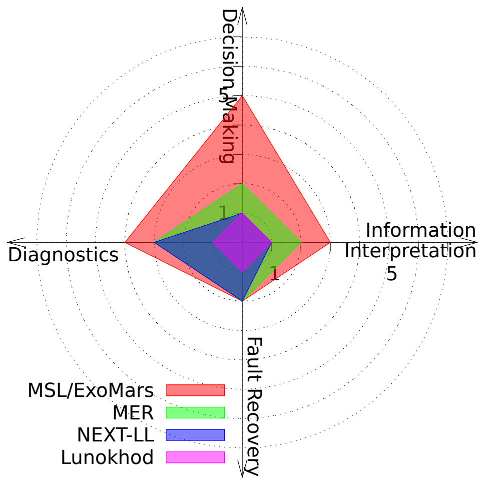


Abbildung 1: *Comparing the autonomy levels of different exploration robots based on the proposed classification scheme*

4 Classification of Existing Systems

In order to verify the proposed scheme, the previously mentioned systems have been analysed towards their autonomy aspects, and classified into the categories given. Figure 1 shows the result of this classification and compares the individual results. It does not come at a surprise, that the lunar systems rate much lower in the autonomy categories compared to the mars rovers. It is also notable that there is a low variance on the fault recovery axis.

5 Testing Autonomy

Benchmarking is the process of comparing one's business processes and performance metrics to industry bests and/or best practices (Wikipedia)

What is then commonly accepted as benchmarks are standardized quantitative metrics that allow to quantify how well a particular solution behaves with respect to existing best practices. As we argued already, we believe that this kind of methodology does not apply to autonomy capabilities, as *autonomy capabilities* are dependent from the particular scenarios the system will be confronted with.

These last years, challenges have started to appear (DARPA's Grand Challenge and Urban Grand Challenge, micro-UAV competition, ELROB [ELR], ESA's LRC [LRC], Sauc-E [Sau]). Instead of trying to benchmark systems against a standardized metric, they evaluate systems by making them compete against each other in the same tests.

One *very* important aspect of these challenges is that, while the competing teams get task and environment descriptions beforehand, the specifics of the scenarios (details of the environment, trajectories, . . .) are not available. It is central to compare each system's robustness, i.e. how the autonomy part can robustly perform regardless of the actual instance of a scenario that it has to confront.

However, while they provide a forum in which systems can be compared to each other, these challenges are far from being *tests*. I.e. they do not provide a way to verify that a given system *does conform* to the autonomy levels it is supposed to.

We will now investigate what kind of tests could be used for each of the autonomy aspects we identified, and outline some ideas of how these could be implemented.

Environment Representation and Information Gathering test environments should include environment that are both benign ("expected") and extreme ("unexpected"). Adding totally unexpected elements could also be used as a way to see how the information assessment system can handle unknowns-unknowns.

Decision making, Diagnostics and fault recovery Low autonomy levels on these aspects can be unit-tested extensively. Evaluating the higher levels, however, is more tricky, as it would probably involve tight cooperation between the three aspects.

In general, an interesting way of testing high levels of autonomy would be to see the system as playing a game (in the "game theory" sense of the term) against the rest of the world. From that point of view, a test would be interactive: the system to be tested would "play" against computers and/or humans that would decide what goes wrong in realtime. Their ability to impact the software of the system to be tested being, for instance:

- simulating hardware failures
- simulating sensing degradation, e.g. changing the sensor error models.
- injecting wrong information into the system's information database – thus simulating potential problems on the information assessment aspect.
- injecting wrong decisions superseding the actual decisions taken by the system – thus simulating potential problems on the decision-making aspect.

In difference from a unit-testing, all these changes would be implemented in real time in a real environment. *When* a change is performed is indeed critical: it will be the role of the "test player" to choose to implement the change at what seems to be the worst situation possible for the system that is being tested.

Importantly, such a testing "game" would need to stay within the boundaries which the system is supposed to handle. It would indeed make no sense to simulate critical sensing degradation on a system that claims a level of two on diagnostics and error recovery.

Testing the hardware capabilities for recovery, however, would be a lot harder. It would indeed require to modify the environment at will, so as to have a broad range of situations the system should be confronted with, which is not practical.

In general, to allow extensive testing, one would require to build:

- a standardized software interface that would be provide the interface necessary to build “game-based” tests, regardless of the actual system implementation. Such a scheme is already applied on a completely different scheme in the domain of automated planning [ICA].
- a network of testing facilities. Over the years, numerous testing facilities have been created by different institutions. Making these facilities available would be a great step towards the general goal. Nonetheless, it would still be important to conduct tests and/or challenges in outdoor environments, where the actual test setup is not known in advance to the tested system.

6 Conclusion

The aspect of autonomy for robotic space exploration is no doubt a very important factor in the mission and system design. Current methods of describing autonomy for these domains are either too limited or too generic. We have given an overview of existing and planned robotic systems for the exploration of celestial bodies, and assessed their autonomous capabilities. A scheme was generated that constitutes of four independent scales, which we believe to be fitting to describe the autonomous capabilities of the system for the given scenario. The scheme can give qualitative information about the level of autonomy. Benchmarking a system is an entirely different matter, and can not be performed so easily. We propose to extend the recent trend in robotics to hold competitions, as they are able to provide at least a comparative information on the autonomy of the systems. Further, we propose that a standardised software interface for for fault injection could lead to a better evaluation of the fault tolerance of a system by letting third parties create real unexpected events, rather than responding to expected faults. The work presented is meant to be taken as a starting point for discussions, and further investigation into the subject is required towards the usefulness of the proposed scheme, and the nature of the competitions and measures for evaluating autonomy.

References

[BMH08] Bajracharya, M. ; Maimone, M.W. ; Helmick, D.: Autonomy for Mars rovers: Past, present, and future. In: *Computer* 41 (2008), December, Nr. 12, S. 44–50. – ISSN 0018–9162

[Bri07] Brian, Harvey: *Soviet and Russian Lunar Exploration*. 2007 (Springer Praxis Books). – 239–286 S.

[Clo02] Clough, BT: Metrics, schmetrics! How the heck do you determine a UAV’s autonomy anyway? In: *NIST Special Publication* (2002), Nr. 990, S. 313–319

[ECS08] ECSS: ECSS-E-ST-70-11C Space engineering - Space segment operability / ECSS Requirements and Standards Division. 2008. – Forschungsbericht

[ELR] *ELROB - The European Robot Trial*. http://www.elrob.org/

[HMA07] Huang, Hui-Min ; Messina, Elena ; Albus, James: AUTONOMY LEVELS FOR UNMANNED SYSTEMS (ALFUS) Volume II: Framework Models / NIST. 2007. – Forschungsbericht

[ICA] *ICAPS Competitions*. http://idm-lab.org/wiki/icaps/index.php/Main/Competitions

[LRC] *ESA’s Lunar Robotic Challenge*. http://www.esa.int/esaCP/SEM4GKRTKMF_index_0.html

[Mai06] Maimone, M.; Biesiadecki, J.; Tunstel, E.; Cheng, Y.; Leger, C.: Surface navigation and mobility intelligence on the Mars Exploration Rovers. In: *Intelligence for Space Robotics*. San Antonio, TX, USA : TSI Press, 2006, Kapitel 3, S. 45—-69

[Mat98] Matijevic, J.: THE PATHFINDER MISSION TO MARS: Autonomous Navigation and the Sojourner Microrover. In: *Science* 280 (1998), April, Nr. 5362, S. 454–455. – ISSN 00368075

[Sau] *Student Autonomous Underwater Challenges (Sauc-E)*. http://www.dstl.gov.uk/news_events/competitions/sauce/index.php

Intelligent Mobility

Terrain Characterization

Version:	1.0
Issue date:	2009-09-15
Contacts:	Sylvain Joyeux sylvain.joyeux@dfki.de Jens Mey jens.mey@dfki.de

ADDRESS
DFKI Robotics Innovation Center
Robert-Hooke-Straße 5
D-28359 Bremen

PHONE
+49 (0) 421 218 64100
FAX
+49 (0) 421 218 64150

INTERNET
robotic.dfki-bremen.de
E-MAIL
robotic@dfki.de

Table of Contents

1 MOTIVATION 3

2 STATE OF THE ART 3

2.1 IDENTIFYING SOIL 3
2.2 DESCRIBING SURFACE MORPHOLOGY 4
2.3 DESCRIBING MECHANICAL SOIL PROPERTIES 5
2.3.1 Young's and shear modulus 5
2.3.2 Critical Angle of internal friction/ repose and cohesion 7
2.3.3 Measuring soil properties 9
2.4 ESTIMATING TERRAIN TRAFFICABILITY 13
2.5 TERRAIN CHARACTERIZATION AND IDENTIFICATION IN ROBOTICS 13

3 PROPOSED METHOD 14

3.1 REQUIREMENTS FOR THE CHARACTERIZATION 14
3.2 THE CHARACTERIZATION PARAMETERS 14
3.3 RESULTS 15

4 CONCLUSION AND OUTLOOK 16

5 REFERENCES 17

5.1 LITERATURE 17
5.2 OTHER REFERENCES 18

6 APPENDIX 18

6.1 LITERATURE OVERVIEW 18
6.2 USCS AND AASHTO SOIL CLASSIFICATION SYSTEMS 18

1 Motivation

The ability to characterize the terrain in front or underneath oneself is important for locomotion adaptation and navigation in unknown environments: Optimal locomotion speed and pattern can be chosen according to detected ground properties, identified obstacles can be avoided by integrating them in the motion planning process. All these operations are of particular importance for mobile robots in outdoor and planetary exploration missions as they have to behave entirely autonomously.
As a first step towards specialized locomotion patterns, the terrain of the experimental site needs to be described. The performance of the robot on different surface and ground conditions can be evaluated afterwards.

2 State of the Art

Terrain characterization includes three main tasks as identifying the soil, describing surface morphology and mechanical soil properties. It also can be used for trafficability estimations and as it is already used in the robotic field of research.

2.1 Identifying soil

There are (at least) two systems available for identifying soil which are generally accepted in engineering:
The Unified Soil Classification System (=USCS) (ASTM[1] D 2487 “Standard Practice for Classification of Soils for Engineering Purposes”)
AASHTO[2] Soil Classification System

Apart from minor differences, the classifications generally show great similarities.
Both classifications are generally based on

- particle size
- particle size distribution and
- plasticity of the soil.

[1] American Society for Testing and Materials
[2] American Association of State Highway and Transportation Officials

Both classifications divide soil in classes of

- gravels
- sands
- silts and
- clays.

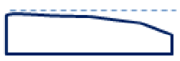
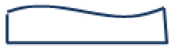
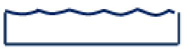
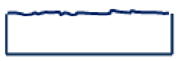
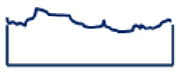
USCS also includes the class “Organic materials”. The major classes (see above) differ regarding their particle size (identified with sieves). Sub-classes of gravel and sand differ in size distribution, sub-classes of silts and clays are distinguished by plasticity due to water content.

Both systems do not directly relate their classes to mechanical properties, but the AASHTO system roughly describes the suitability of the classes as a subgrade for highway construction purposes.

2.2 Describing surface morphology

In an engineering description, surfaces are structured in several orders. If we consider a perfectly flat surface, any major derivation from this is a “first order” irregularity. If the resulting surface has additional waves, these are “second order” irregularities. Together, both form the macro morphology. The micro morphology contains roughness from grooves (3. order) and scratches or scales (4. order). The resulting surface morphology can be a superposition of 1.-4. order structures (Hoischen 2003).
This description is usually used for characterizing deviations due to manufacturing errors or imprecision but is also useful for general surface descriptions (Lemburg 2009).

Table 1: Engineering description of surface morphology

surface morphology	description
1. order 	variations in shape
2. order 	corrugation
3. order 	roughness: grooves
4. order 	roughness: scratches, scales, ...
	resulting morphology: superposition of 1.-4. order

2.3 Describing mechanical soil properties

2.3.1 Young's and shear modulus

In soil dynamics, engineers are interested in the reactions of the ground to an applied force. Hence, they want to determine the relationship between stress (force per area) and deformation (=strain). Deformation can be completely elastic (when the force is removed, the deformation decreases until it reaches to the originally shape) or elastic-plastic (parts of the deformation stay after the forces has been removed and the new shape varies from the original). The correlation between stress and deformation is given by the material properties. These are expressed by the *Elastic* or *Young's modulus E* (reaction to linear stress) and the *shear modulus G* (reaction to shearing stress) and often illustrated in stress-strain curves. A solid (including soils) is defined as a material that can support shear stress without moving. However, every solid has an upper limit to how much shear stress it can support.

$$E = \frac{linear\ stress\ \sigma}{linear\ strain\ \varepsilon} = \frac{F/A_0}{\Delta L/L_0}$$

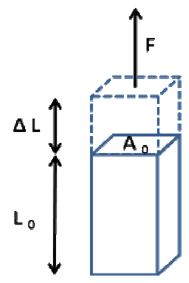


Figure 1: The force F (here: pulling, pushing also possible) induces a linear stress and causes a change in length ΔL and therefore a line strain.

$$G = \frac{shear\ stress\ \tau}{shear\ strain\ \gamma} = \frac{F\ /A}{\tan\theta}$$

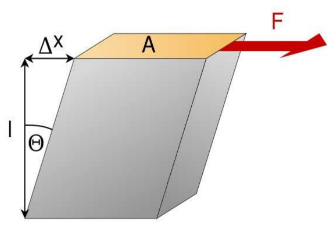


Figure 2: The force F deforms the object (max. displacement Δx) and induces both shear stress and strain. Image: Wikimedia commons

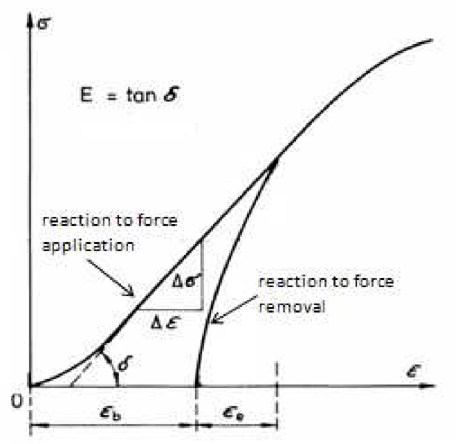


Figure 3: Stress-strain curve of a soil. With increasing stress σ, the strain (= deformation) ε increases non-linear. The deformation can be divided into an elastic part (here: ε_e) and an irreversible part (here: ε_b). Image changed after Kunze et al. (2002).

The general behavior of soils depends both on the properties of the soil itself as

- void ratio
- degree of saturation
- critical angle of internal friction/repose
- cohesion

and of the properties of the applied stress

- mean principle stress
- stress frequency
- strain amplitude
- number of stress cycles.

Additionally to that, the stress-strain-curve of soil is not linear. Hence, E and G (the slopes of these curves) of one particular soil are not constant values. They depend on the already applied stress and strain.

As these dependencies are complex, *predicting soil reactions is rarely impossible*. Young's modulus E and shear modulus G or equivalent values *have to be determined experimentally* for a given soil sample (Studer et al. 2007)(see 2.3.3).

Once determined, E and G are material properties which are independent of the properties of the body that applies the force. E and G cannot be used to directly calculate reaction forces between the soil and an object (foot, wheel). The calculation of reaction forces in an elastic collision (based on conservation of linear momentum) depends on additional factors that have to be determined empirically.

2.3.2 Critical Angle of internal friction/ repose and cohesion

Instead of characterizing soil with E and G modules, it is also common to describe the shear strength of a soil by its angle of internal friction/angle of repose φ.

$$\tan \varphi = \frac{f_f}{f_n}$$

While forces normal to the surface press the soil together and usually stabilize it, forces along the surface (shear forces) tend to displace material and to destabilize the system. A shear force has to work against (the internal) friction. If it exceeds the friction force, the system gets unstable. The vectors of normal and shear force result in an angle which is generally used to specify the critical ration of normal to shear forces.

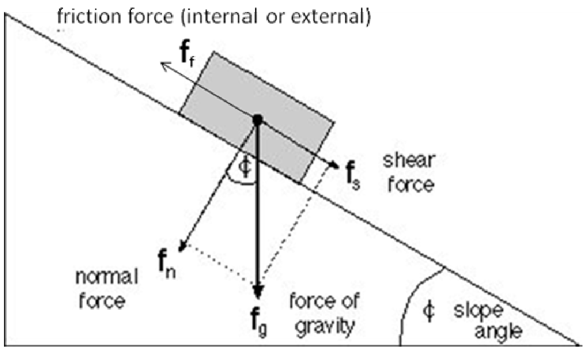


Figure 4: Forces on an objects lying on an inclined plane: The force of gravity splits up in two components. The shear force acts along the plane, the normal force perpendicular/normal to it. In equilibrium, the friction force has to equal the shear force. If the shear force exceeds the friction force, the object slides down.

For soil, this angle is called angle of repose or angle of internal friction and can also be seen in the conical pile of any granular material. The largest possible cone angle is found, when the gravitational force component along the surface equals the friction.
For vehicles, the angle of internal friction is one factor that describes how much torque can be applied to the ground before the soil/the wheel starts slipping. Additionally to the internal friction, the external friction between ground and wheel has to be taken into account.

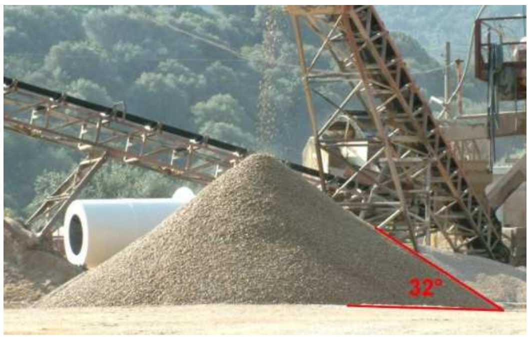


Figure 5: Angle of repose in a conical pile. Image: wikimedia commons

Table 2: Angles of repose/internal friction for typical soils. From Fowler (1919) and The British Steel Construction Manual (Online resource).

	angle of repose/ internal friction φ [°]	
material	**After Fowler (1919)**	**After BSCM**
clay, dry	29	30
clay, damp, well drained	45	45
clay, wet	16	15
earth, dry	29	-
earth, moist	45-49	50
earth, very wet	17	15
gravel, clean	48	40
gravel, with sand	26	30
sand, fine dry	31-37	30-35
sand, wet	26	35
sand, very wet	32	25
shingle, loose	39	40
peat	14-45	-

The cohesion *c* and the internal friction angle φ are the two key terrain parameters. These parameters can be used to compute/estimate the maximum terrain shear strength τ_{max}, and thus give an estimate of vehicle trafficability, from Coulomb's equation:

$$\tau_{max} = \left(c + \sigma_{max} * \tan\varphi\right)$$

where σ is the normal stress acting on the terrain region (Iagnemma 2002).

2.3.3 Measuring soil properties

Laboratory measurements:

The "Triaxial shear test" is the most precise and accurate test set-up for measuring soil properties (Studer et al. 2007).

For the test, the soil is filled in a flexible cylinder. The top or bottom closing plate of the cylinder can be moved to apply pressure on the soil. Additionally, the cylinder is placed in water that provides pressure along its sides. The water pressure and plate pressure add up to a three-dimensional stress that contains linear as well as shear stress components. The distance that the upper platen travels is measured as a function of the force required to move it.

Additionally, the pressure of the surrounding water is controlled. The stress on the platens is increased until the material in the cylinder fails. From the triaxial test data, it is possible to extract fundamental material parameters about the sample, including its angle of internal friction and cohesion (www.wikipedia.com: "triaxial shear test").

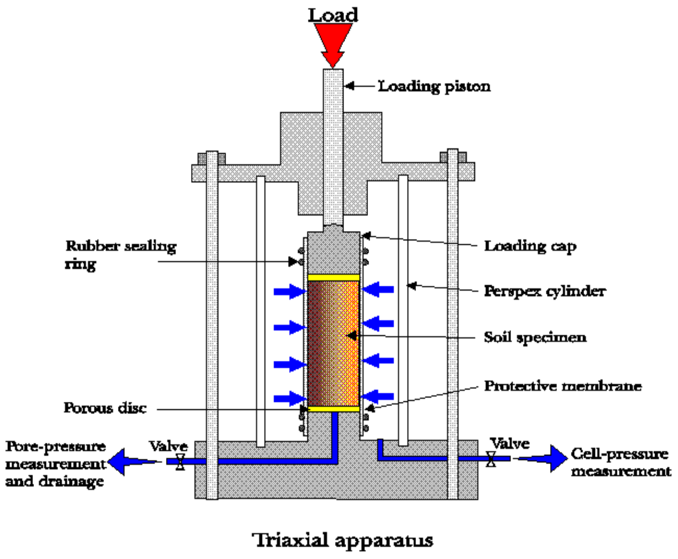
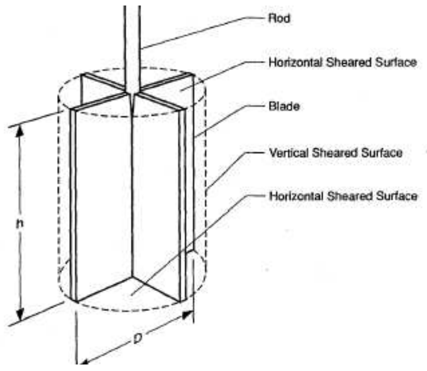


Figure 6: Triaxial shear test set-up (from: University of West England, http://environment.uwe.ac.uk/geocal/SLOPES/SLOPSITE.HTM).

Field measurements:

There are several test devices for in-situ shear tests. Among others, there are

- the "Shear vane":



•

Figure 7: Shear vane device (from Shoop 1993, after Kogure et. al 1988)

- the “Vane cone”:

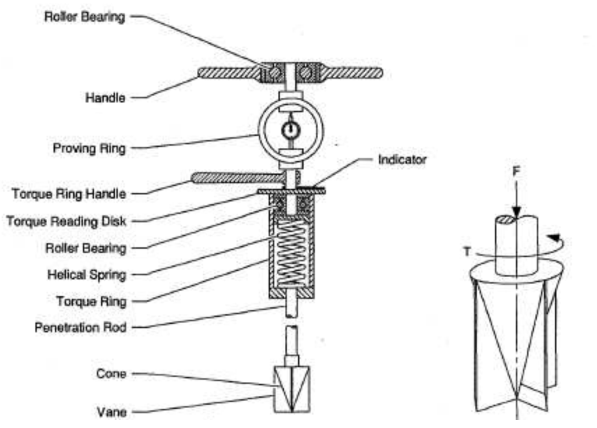


Figure 8: Vane cone (from Shoop 1993, after Young and Youssef 1978)

- the “Cohron Shearograph”

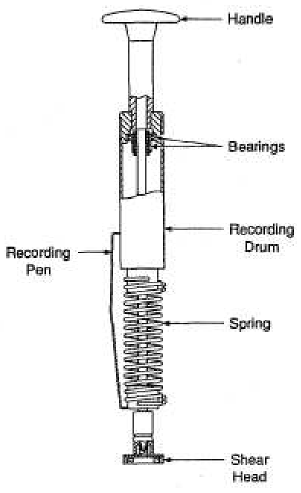
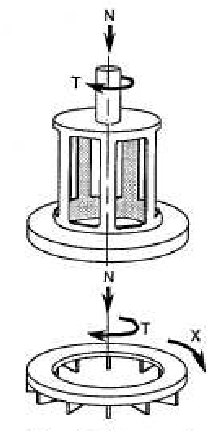


Figure 9: Cohron shearograph (from Shoop 1993, after Karafiath and Nowatzki 1978)

- and the “Shear annulus”



•

Figure 10: Shear annulus (from Shoop 1993)

All these devices are pressed into the soil and then rotated. The force needed for rotating is recorded. Parameters like cohesion and angle of internal friction can be calculated. To obtain comparable results with one method, some standards are available for carrying out the experiments. Results from different testing methods do not equal each other (see Tab. 3). Most devices are not adequate for every type of soil.

Table 3: Comparison of different methods for cohesion and friction measurements (from Shoop 1993, after Stafford and Tanner 1982)

	Torsional shear	**Triaxial test (Fig. 6)**		**Shear vane (Fig. 7)**
Soil	**Annulus (Fig. 10)**	**undrained**	**drained**	
a) Cohesion (in kPa)				
Sandy clay loam	39,3	6,5	17,7	43,3
Clay	41,9	11,1	14,4	54,1
Clay with stone	88,6	33,5	41,9	92,8
Peat	36,1	11,3	15,5	50,3
Remolded clay	62,3	10,3	21,1	38,5
Remolded sand	4,2	4,9	3,1	5,5
b) Friction angle (in degree)				
Sandy clay loam	33,2	25,4	29,7	-
Clay	13,7	17,3	26,4	-
Clay with stone	22,9	11,3	11,3	-
Peat	30,8	16,9	21,9	-
Remolded clay	21,3	6,0	2,8	-
Remolded sand	8,8	31,8	33,3	-

As the results generally differ depending on the testing device, it was suggested to design the device according to the objectives of the experiments. Hence, if soil characteristics are needed for estimating its qualities as a road subgrade, wheel motion should be simulated. The terrain-wheel shear parameters can be calculated and can be used directly for trafficability estimations. For more detailed information, see Shoop (1993).
Additional to these simple mechanical test devices, cone penetration devices are available that obtain the shear modulus by measuring the travel speed of seismic waves. The cone penetration test (CPT) can also collect various additional data via induces fluorescence, conductivity sensors, cameras etc. CPT testing equipment forces the cone into the soil using hydraulic rams mounted on either a heavily ballasted vehicle or using screwed-in anchors as a counter-force.

2.4 Estimating terrain trafficability

Terrain trafficability is only estimated for a well-defined application. One example is terrain characterization for operational planning of forest work (Forestry commission 1995). The forest terrain is described by ground conditions, ground roughness and slope. Each of these categories consists of several classes that are ranked regarding their trafficability. The classes of slope (“Level“, „Gentle”, “Moderate”, “Steep”, “Very Steep”) or roughness (“Very even”, “Slightly even”, “Uneven”, “Rough”, “Very Rough”) can only be defined when they refer to dimensions and abilities of typical forest machines like tractors or harvesters.

2.5 Terrain characterization and identification in robotics

In robotics, mechanical soil properties (e.g. Iagnemma 2002, Ojeda 2005, Karlsen 2007) and surface morphology (e.g. Howard 2001, Brooks 2005, Karlsen 2007, Ascari et al. 2006) are used for terrain characterization. Mechanical properties are derived via wheel-terrain-interaction analysis; morphology is described by cameras, vibration sensors or flexible pressure sensor arrays.
Additionally, characterization is based on parameters like color or texture from camera data (e.g. Manduchi 2005, Halatci 2008).
The general approach is to obtain data sets on several terrains in a training procedure and to compare them to the data from an unknown terrain. Data sets are compared by calculations for obtaining terrain identification. Iagnemma et al. (2002) implemented a terrain identification algorithm that requires some “low speed tests to enable high speed behavior” afterwards.
For an overview, see Appendix 1.

3 Proposed method

3.1 Requirements for the characterization

The following requirements are given:

1. Only criteria that are related to trafficability are included. (Parameters like color might help to identify the terrain but not to characterize it.)
2. It should be possible for humans to measure or estimate the criteria.
3. The terrain description should be platform independent. The characterization should only include properties of the terrain itself but no results from an interaction of the robot with the terrain.

3.2 The characterization parameters

The proposed terrain parameters for characterization are shown in Tab. 4. They were derived from characterization approaches that were presented earlier. Additionally to the parameters themselves, it includes the appropriate measuring devices and corresponding units. As the actual parameter values strongly depend on the measurement device (compare Tab. 3) and are hard to obtain from literature, some approximate values are given for orientation. For rough estimations, some complex measurement set-ups can be replaced by easier measurement devices or simplifications and assumptions.

Table 4: Overview on terrain parameters

terrain parameter	measured by	given in [unit]	order of magnitude	estimated by
slope	inclinometer	Degree [°] or meters of elevation over 100 meter distance [%]		-
obstacle height	measuring tape, image analysis	height [m]		-
obstacle frequency	counting, image analysis	number per reference area or length [1/m² or 1/m]		-
obstacle flexibility	bending test	bending modulus [N/m²=Pa]	wood: ~ 5MPa	defining rough classes based e.g. on the deformation caused by a human step
linear strength of the soil	triaxial tests, cone penetration tests incl. seismic wave (measured under compression)	Young's modulus [N/m²=Pa]	concrete, cured: ~30 GPa	terrain-„object“-interaction
shear strength of the soil	triaxial tests, cone penetration tests incl. seismic waves	shear modulus [N/m²=Pa]	sand, dry: ~ 100-200 MPa	terrain-„object“-interaction
cohesion of the soil	triaxial tests, in-field shear tests, cone penetration tests incl seismic waves	stress [N/m²=Pa]	sand, dry: neglectible clay, soft: 10 kPa clay, stiff: 20 kPa	terrain-„object“-interaction
angle of repose/internal friction of the soil	triaxial tests, in-field shear tests	ratio of friction forces to normal forces [°]	sand, dry:~35° clay, damp:~45° concrete, cured: - compare Tab.2	literature values, natural angle of incline of a conical pile, terrain- “object”-interaction

3.3 Results

- Soils can be easily identified, which is a first step towards terrain characterization. Unfortunately neither the USCS nor the AASHTO connects the resulting classes to any property that characterizes the terrain regarding its trafficability. Therefore the schemes are not useful for the planned application (compare to requirement 2).

- The engineering way of describing surfaces as a superposition of several structural orders/layers can be adapted to describe terrain morphology. As terrain obstacles show larger variations than fabrication derivations, additional descriptions of the obstacles are needed. A first order description can be given by the slope of the terrain. Furthermore, the density/frequency of the obstacles and their height and flexibility are needed.
- Soil properties like Young's and shear modulus are important properties of the terrain, but hard to obtain (conflicts with requirement 3). Furthermore, they cannot be used directly to calculate reaction forces (which could lead to a trafficability estimation) even when the properties of the robots feet/wheels are known (conflicts with requirement 2).
- Material characteristics like reactions to linear and shear stress, that do not equal the "official" E and G values but describe the same physical properties, are used in several robotic applications for terrain characterization. For example, Ojeda et al. (2005) use the wheels of their skid-steered robot to perform real-time measurements on the shear stress-strain behavior of the terrain.
- Cohesion and angle of internal friction also describe shear strength and can be measured in the field. By using test devices that simulate the real stress situation, the test results can be used directly for trafficability estimations.
- Trafficability is always an estimation that refers to one defined vehicle, as its dimensions, weight, style of locomotion and the purpose of the locomotion are needed to evaluate the terrain's parameters. Additionally, parameters like terrain material characteristics are more useful and much easier to obtain when they are related to the vehicle. These findings lead to the conclusion that the characterization should be platform-dependent (conflicts with requirement 4).

4 Conclusion and Outlook

The suggested characterization parameters provide a comprehensive description of the terrain. Qualitative parameter values for different terrains are scarcely available and results from different measurement devices are not comparable. Hence, for characterizing and comparing different terrains, own measurements or estimations are needed.

Although it does not meet the original requirements, it is suggested to characterize terrains platform-dependent. Many useful parameters are related to the system that is moving on the terrain: The size of the moving vehicle or robot and its way of locomotion is crucial for the critical size of obstacles. The material properties of the wheels or feet and the weight of the robot are important to estimate the reaction forces on the ground and the robot and their behavior.

If terrain is characterized platform-dependent, it is also possible to estimate linear and shear strength, cohesion and friction angle directly by terrain-wheel-interaction. Thus, complex test set-ups like triaxial tests can be replaced. The results from terrain-wheel interaction test would not only characterize the terrain, its results could also be used directly for trafficability estimations. The latter could not be achieved by results from other tests (e.g. cohesion, friction angle from in-field shear tests, Young's and shear modulus from laboratory triaxial tests).
The characterization can also be tested on Moon and Mars –like soils. They are simulated with the appropriate internal friction angles, cohesion and particle size distribution are available for research purposes (McKay et al. 1994, Allen et al. 1997).

5 References

5.1 Literature

Ascari, L.; Ziegenmeyer, M.; Corradi, P.; Gaßmann, B.; Zöllner, M.; Dillmann, R.; Dario, P. (2006): Can statistics help walking robots in assessing terrain roughness? Platform description and preliminary results. Proceedings of the 9th ESA workshop on Advances Space Technologies for Robotics and Automation, ASTRA 2006 ESTEC, Noordwijk, The Netherlands. November 28-30, 2006.

Allen, C. C.; Morris, R. V.; Lindstrom, D. J.; Lindstrom, M. M.; Lockwood, J. P. (1997): JSC Mars-1 - Martian regolith simulant. Lunar and Planetary Science XXVIII.

Brooks, C.A.; Iagnemma, K. (2005): Vibration-Based Terrain Classification for Planetary Exploration Rovers. IEEE Transactions on Robotics, Vol. 21, No. 6.

Forestry Commission. Technical Development Branch (1995): Terrain classification. Technical Note 16/95.

Fowler, C.E. (1919): A Practical Treatise On Engineering And Building Foundations Including Sub-Aqueous Foundations. Wiley.

Halatci, I.; Brooks, C.A.; Iagnemma, K. (2008): A study of visual and tactile terrain classification and classifier fusion for planetary exploration rovers. Robotica, volume 26, pp. 767–779.

Hoischen, H. ; Hesser, W. (2003): Technisches Zeichnen. 29. Auflage. Cornelsen, Berlin

Howard, A.; Seraji, H. (2001): An Intelligent Terrain-Based Navigation System for Planetary Rovers. IEEE Robotics & Automation Magazine.

Iagnemma, K.; Dubowsky, S. (2002): Terrain estimation for high-speed rough-terrain autonomous vehicle navigation. Proc. of the SPIE Defense and Security Conference, Unmanned ground vehicle technology IV, Orlando, FL, USA.

Karlsen, R.E.; Witus, G. (2007): Terrain Understanding for Robot Navigation. Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego, CA, USA.

Kunze, G.; Göhring, H.; Jacob, K., Scheffler, M. (editor) (2002): Baumaschinen : Erdbau- und Tagebaumaschinen. Vieweg, Braunschweig.

Lemburg, J.P. (2009): Methodik der schrittweisen Gestaltsynthese. Dissertation. Lehrstuhl und Institut für Allgemeine Konstruktionstechnik des Maschinenbaus. RWTH Aachen. Zugleich erschienen in: Schriftenreihe Produktentwicklung und Konstruktionsmethodik 6. Shaker, Aachen.

Manduchi, R. (2005): Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. Autonomous Robots 18, 81–102.

McKay, D.S.; Carter, J.L.; Boles, W.W.; Allen, C.C.; Allton, J.H. (1994): JSC-1: A New Lunar Soil Simulant. Engineering, Construction, and Operations in Space IV. American Society of Civil Engineers, pp. 857-866

Ojeda, L.; Borenstein, J.; Witus, G. (2005): Terrain Trafficability Characterization with a Mobile Robot. Proc. of the SPIE Defense and Security Conference, Unmanned Ground Vehicle Technology VII, Orlando, FL, USA.

Shoop, S. (1993): Terrain Characterization for trafficability. CRRL report 93-6; Cold regions research and engineering laboratory. US Army Corps of Engineers.

Studer, J.A.; Laue, J.; Koller, M.G. (2007): Bodendynamik. 3. Auflage. Springer, Berlin Heidelberg.

5.2 Other References

AASHTO system:
http://training.ce.washington.edu/WSDOT/Modules/04_design_parameters/aashto_terms.htm

British Steel Construction Manual (last update 2001):
www.civl.port.ac.uk/britishsteel/media/BSCM%20HTML%20Docs/Angle%20of%20repose.html

USCS:
www.civil.queensu.ca/people/faculty/raymond/Notes/341-2UndergradCourseNotes/05-CLASS.PDF

6 Appendix

6.1 Literature overview

6.2 USCS and AASHTO soil classification systems

The iMoby toolchain

Sylvain Joyeux

DFKI Bremen - Forschungruppe Robotik
& Universität Bremen
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

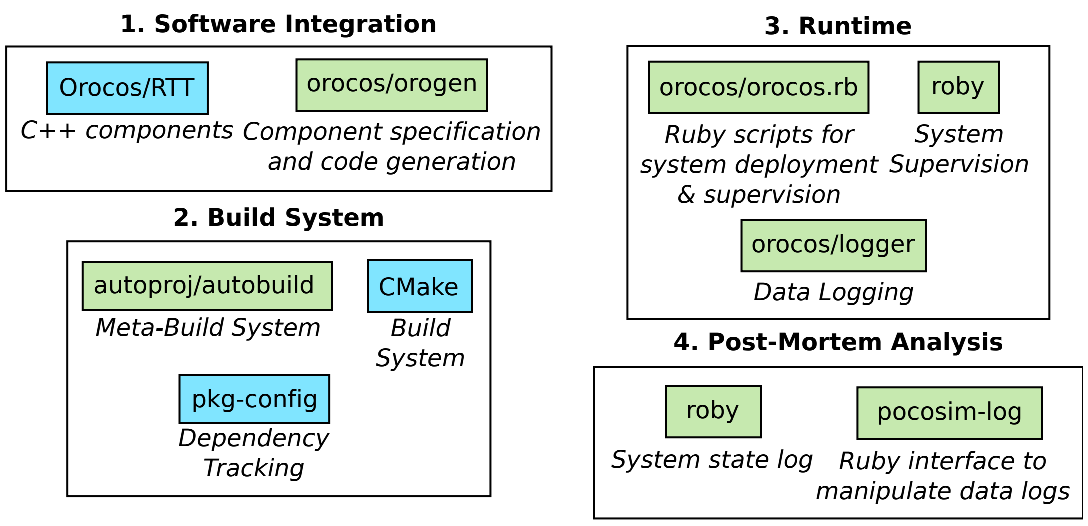
Outline

1 Software Integration

2 Build

3 Runtime

Tools Overview



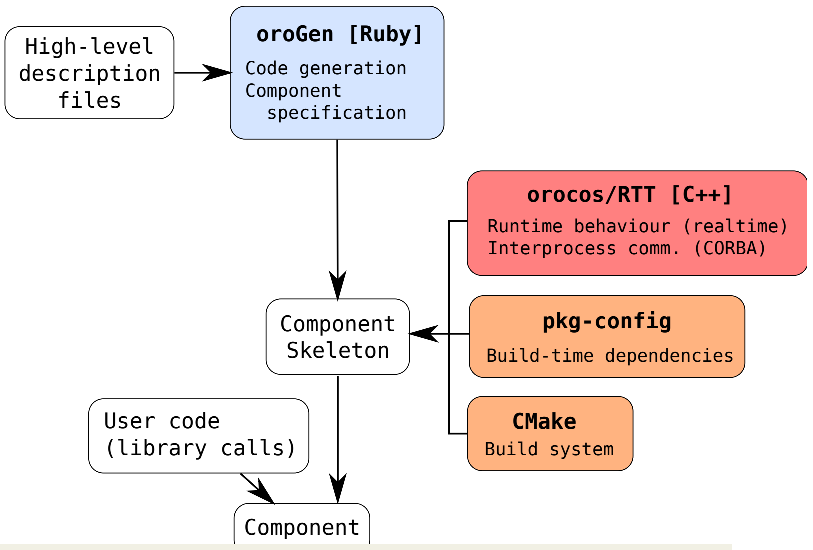
1. Software Integration

Process

1. integrate the functionality in neatly packaged libraries
   - cmake for build system
   - pkg-config for build-time dependencies
   - reuse as much as possible
2. interface that functionality in an oroGen module

The oroGen integration

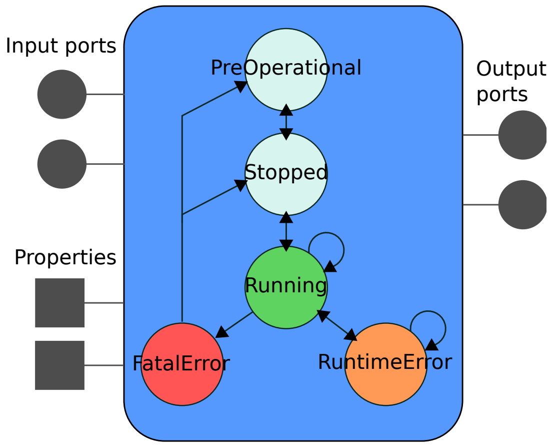
Users guide: `http://doudou.github.com/orogen`



What is a module (a.k.a. *task context* in RTT)

Representing computation

- code organized in a state machine
- dataflow made of
  - ports: dynamic data flow
  - properties: configuration
- control flow
  - methods: synchronous calls
  - commands: asynchronous calls



Writing a module: 3 steps

declare the dependencies on what libraries and task libraries your will your module depend ?

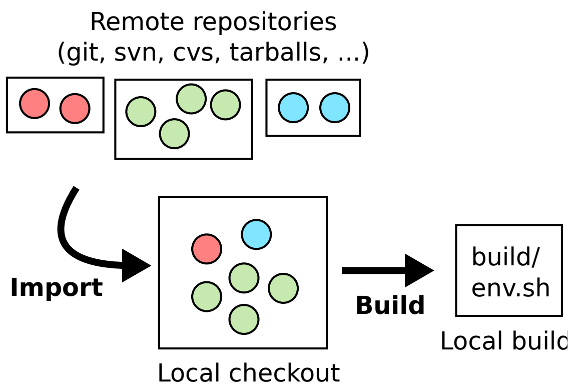
define new types what types will you need in the interface of your module, that are not already in other modules. Define them as C++ structures, with some constraints.

define the interfaces what data do you need ? What data do you produce ? What are the static configuration values ?

2. Build

Autoproj process overview

Users guide: `http://doudou.github.com/autoproj`



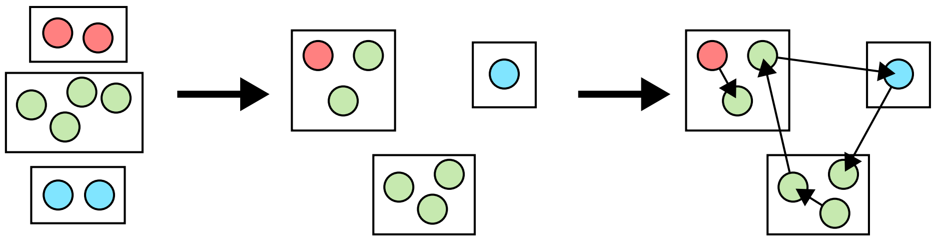
3. Runtime

The Process

Task libraries (oroGen)

Static Deployments (orogen)
associate task contexts with threads, and puts them into processes

Runtime Deployment (orocos/orocos.rb)
manages the UNIX processes, the state machines, the data flow, ...



Method 1: Ruby and orocos/orocos.rb

Users guide: `http://doudou.github.com/autoproj`

```
Orocos.spawn 'lowlevel', 'imu', 'gps' do
  imu = TaskContext.get 'imu'
  # Set property (configuration variable)
  imu.port = '/dev/ttyS1'
  # Asks the module to check the
  # availability of the IMU
  imu.configure
  # Start acquiring data
  imu.start
  # The IMU is configured and runs. Display.
  imu_readings = imu.orientation_readings.reader
  loop do
      sleep 1
      pp imu_readings.read
  end
end
```

Method 2: roby

- model-based deployment
  - a reusable part declarations about the modules
  - a robot specific part the actual robot and deployment definition
- dynamic reconfiguration

Runtime

Method 2: example

```
Robot.devices do
  device(Mb500).
    period(1.0).
    device_id("/dev/dgps_mb500")

  device(XsensImu).
    period(0.010).
    device_id("/dev/ttyS1")

  device(DfkiImu).
    period(0.018).
    device_id("/dev/imu_dfki")

  com_bus('can', :as => 'can0').
    device_id '/dev/can1'

  through 'can0' do
    device(Joystick).
      period(0.1).
      device_id(0x100, 0x7FF)
    device(Sliderbox).
      period(0.1).
      device_id(0x1D0, 0x7FF)
    device(Motors).
      period(0.001).
      sample_size(4).
      device_id(0, 0x700)
    device(ExperimentMarkers).
      period(0.1).
      device_id(0x1C0, 0x7FF)
    device(SystemStatus).
      period(0.1).
      device_id(0x101, 0x7FF)
  end
end
```

Runtime

Method 2: contd. example

```
add(PoseEstimation).
    use  "Orientation" => 'xsens_imu',
         "IMUSensors" => 'xsens_imu'
add(Sysmon::Task)
add("piv_driving", ManualDriving).
  use 'Control.Controller' => Control::PIVController
```

- configures the can bus properly
- configures each driver properly
- adds any needed connection
- computes the required connection policies

4. The End

What do you need to think about ?

- where to put your types
- have your types generic enough
- the RTT does not do type slicing !!!
  - ⇒ think about what you are going to use these types for

  example: the base::RigidBodyState type
- do not over-engineer your modules
  - ⇒ if the functionality is neatly packaged, changing the modules is almost painless

Comparison of Amber and Zigbit wireless modules

Inhaltsverzeichnis

1 Testapplication **1**

2 Amber Configuration **2**

3 Zigbit Configuration **2**

4 Experiments **3**

5 Matlab results **4**

5.1 1m distance 4

5.1.1 Zigbit 4

5.1.2 Amber 5

5.2 10m distance 6

5.2.1 Zigbit 6

5.2.2 Amber 7

5.3 Outdoor 8

5.3.1 Zigbit 8

5.3.2 Amber 9

6 Average values **10**

6.1 1m distance 10

6.2 10m distance 10

6.3 Outdoor 10

1 Testapplication

For the comparison of the charactaristics of amber and zigbit modules was a simple ping-pong application used. This application based on the `arc_packet` and `arc_driver` structure.

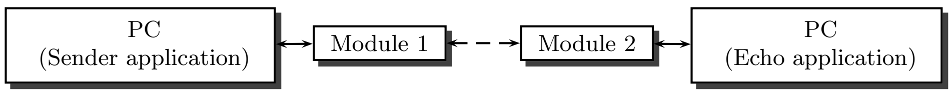


Abbildung 1: Test application

The procedure is as follows:

1. The sender application creates a random data packet and sends the packet via serial interface to module 1
2. Module 1 transmits the packet to module 2
3. Module 2 forwards the frame to the echo application, which simply send the frame back.
4. After the frame is received at module 1, the sender application compares length and data with the origin frame and recently send a new frame.

Due the `arc_packet` structure the maximum data length for amber is at 16 Bytes (+ 4 Bytes protocol overhead), for zigbit at 20 Bytes.

The timeout parameters are set as follows:

Packet send timeout	200 ms
First byte timeout	100 ms
Resend attemps	4

The sender application provides every second an output with the following values:

- Number of correct frames
- Number of error frames (lenght-, compare or timeout-error)
- Bandwidth in bytes/s
- Latency (send-receive time difference by using `gettimeofday()`)

All values are visualised by using matlab R2008b.

2 Amber Configuration

The amber modules are configured in the same way as currently (with firmware version 2.0.0) used on the asguard:

```
AT&V

Local address       : 1
Target address      : 1
Baud rate           : 115200,8,N,1
Flow control        : Disabled         (0)
Operating mode      : Transparent      (2)
RF frequency        : 869412500 Hz
RF channel          : 5
RF data rate        : 9.6kbps (3)

AT&Z

Frequency correction        : -3
PA power                    : 200
Max. retrys                 : 2
Escape char                 : +
TX char limit               : 20
TX time limit               : 1
End delimiter 1st byte      : 13
End delimiter 2nd byte      : 10
```

3 Zigbit Configuration

Frequency	868 Mhz
Modulation	QPSK
RF data rate	100 kbps
Channel	0
Tx power	12,6mW (+11dBm)
Baud rate	115200,8,N,2
Antenna	W1063(900 MHz, 3 dBi)
Firmware	1.8.0

The following equation was used to convert the given dBm value to the transmit power:

$$p(mW) = 10^{\frac{l(dBm)}{10}} \cdot 1mW \tag{1}$$

4 Experiments

There are several experiments:

1. 1m static distance between sender and echo (indoor)
2. 10m static distance between sender and echo (indoor-outdoor)
3. Clockwise walk (picture 2) with the echo module around the dfki building, starting at the main door. The second module stays in the asguard room (ground floor, with windows to the parking lot side).

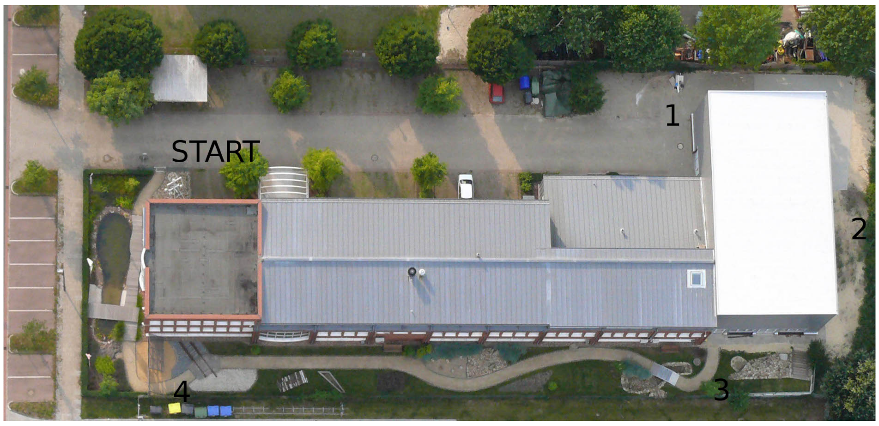


Abbildung 2: DFKI (topview)

Some additional infos:

- The antenna of the modules in all experiments were in a horizontal position.
- A test to use zigbit and amber at the same time failed. The zigbits do not work anymore, probably caused by the 200mW against 12,6mW.
- All experiments took place at “good” weather.

5 Matlab results

5.1 1m distance

5.1.1 Zigbit

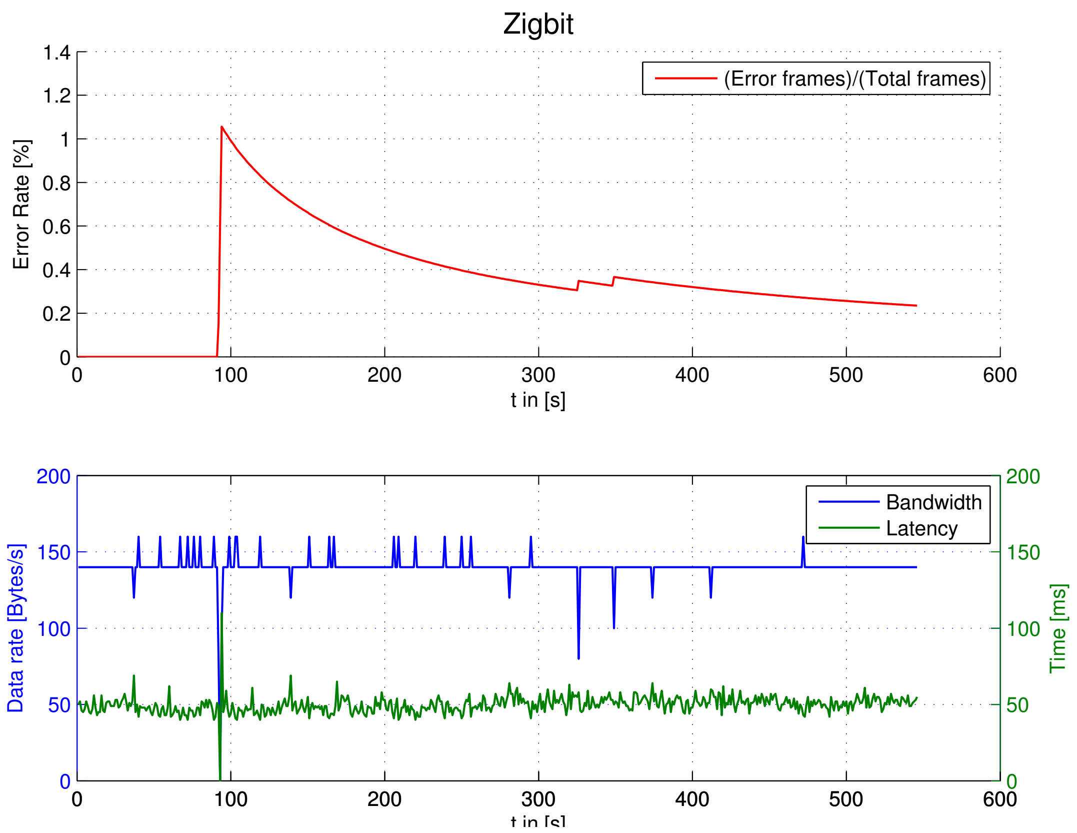
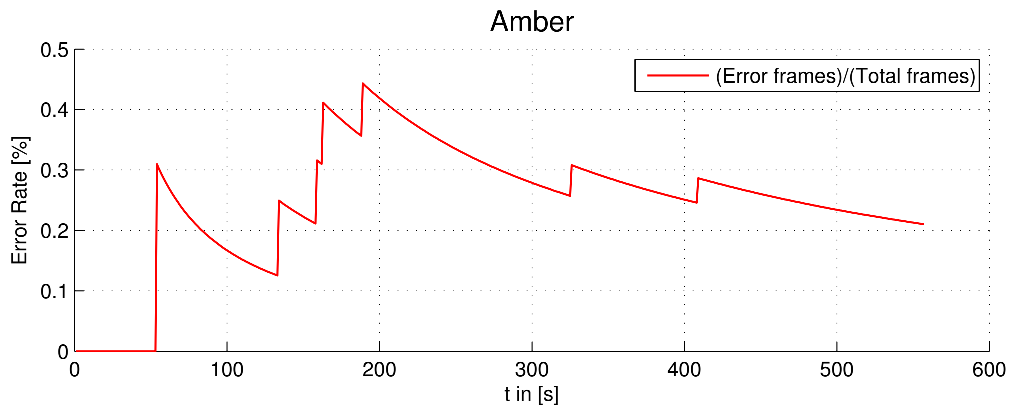


Abbildung 3: Zigbit (1m distance)

5.1.2 Amber



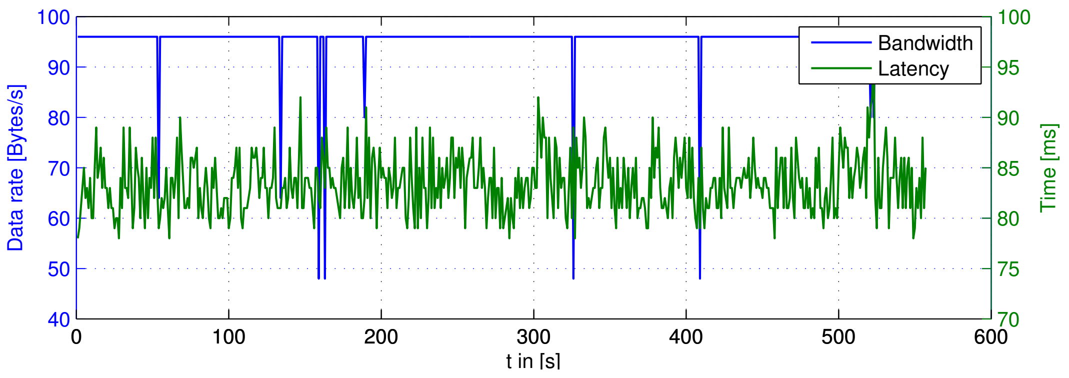


Abbildung 4: Amber (1m distance)

5.2 10m distance

5.2.1 Zigbit

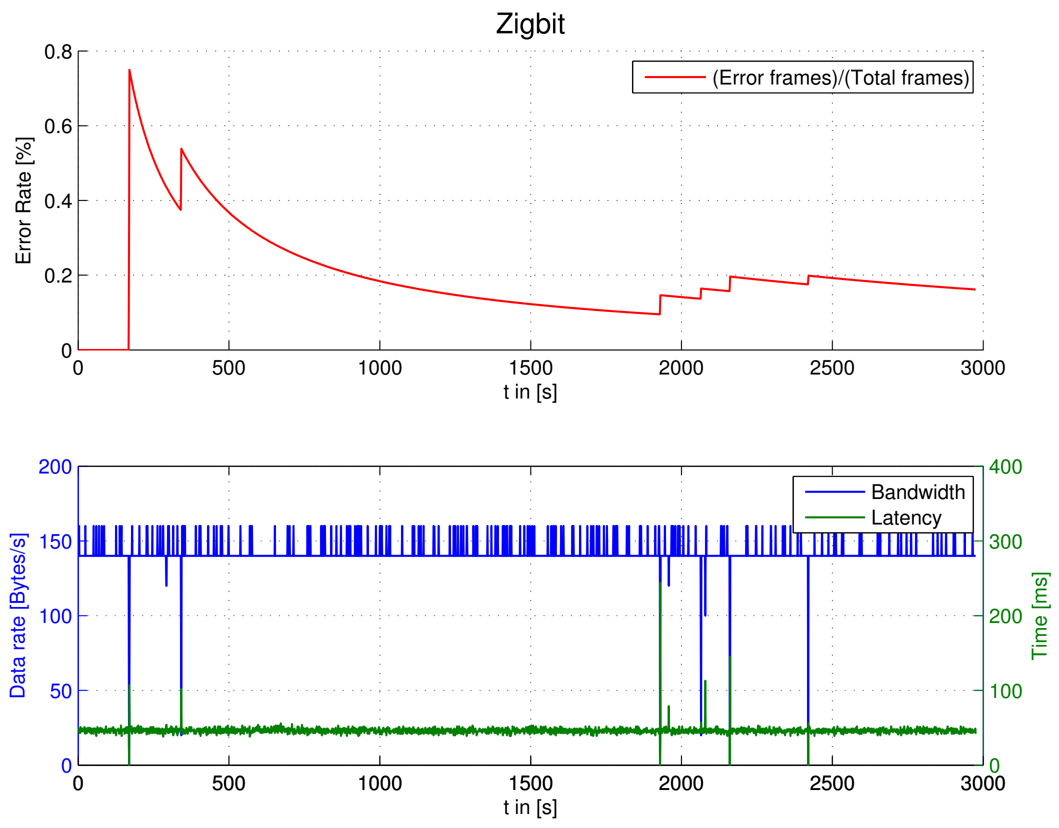


Abbildung 5: Zigbit (10m distance)

5.2.2 Amber

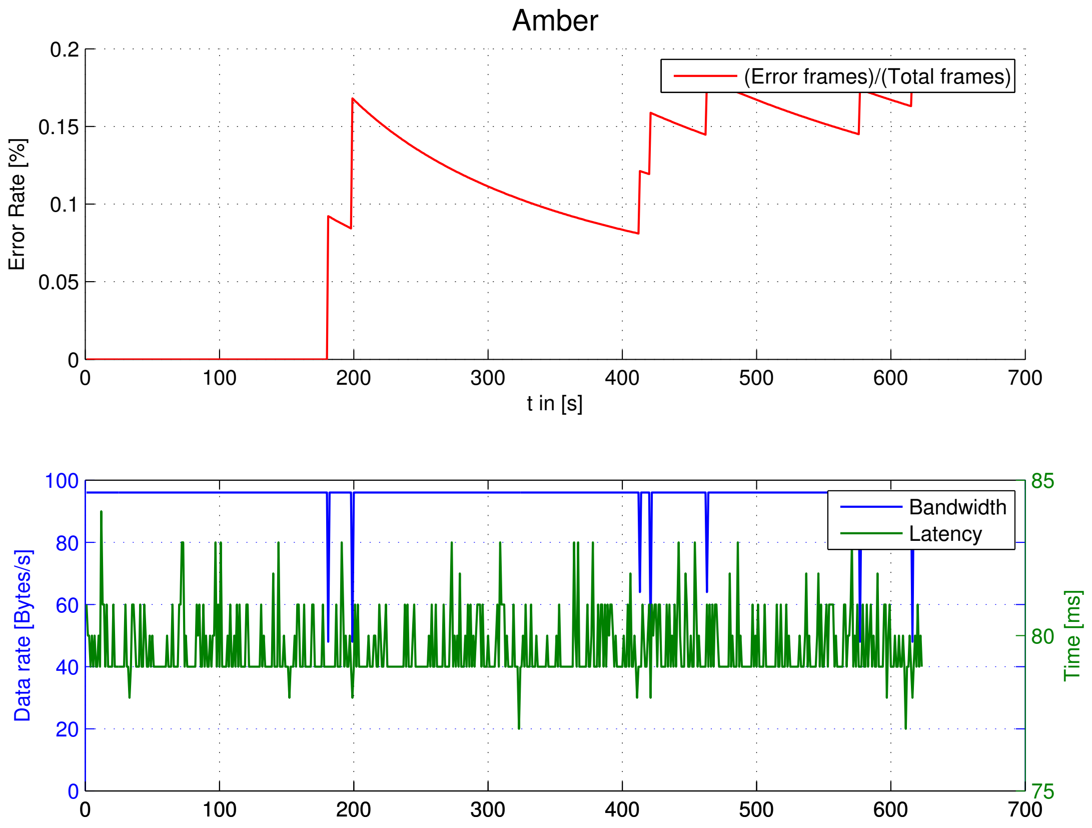


Abbildung 6: Amber (10m distance)

5.3 Outdoor

5.3.1 Zigbit

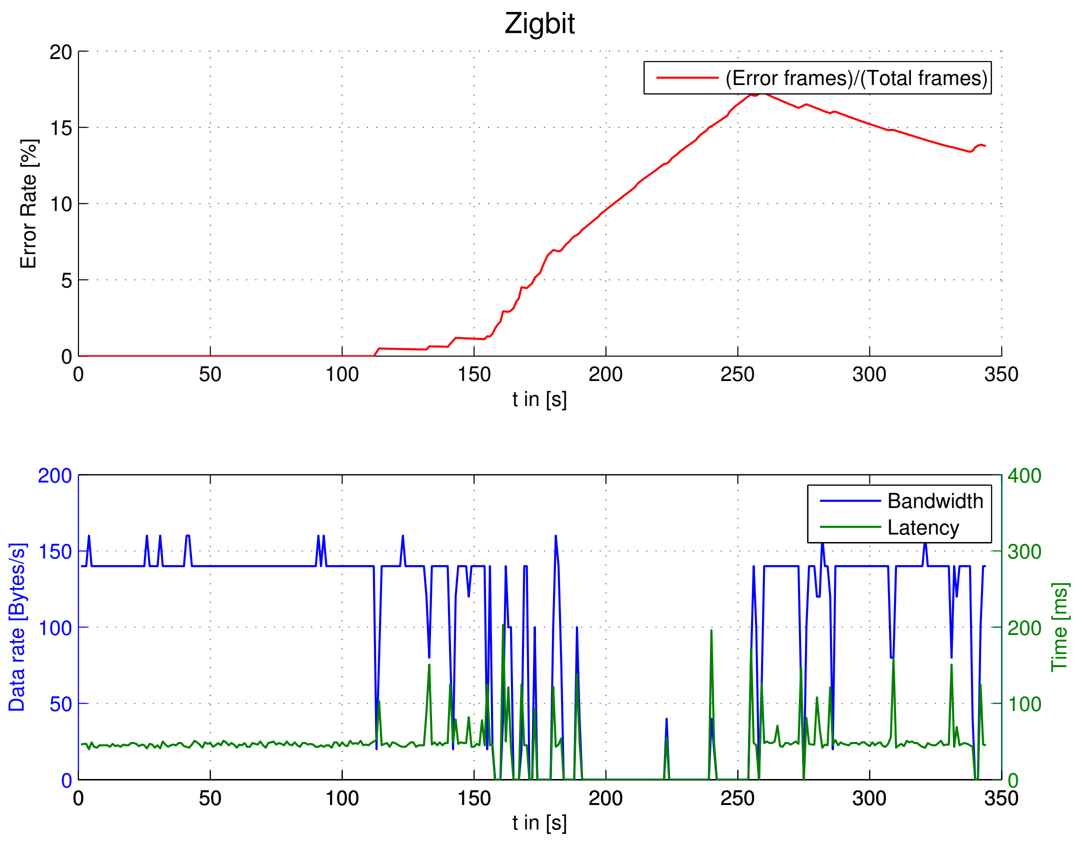


Abbildung 7: Zigbit (Outdoor)

- Behind the exploration hall (150s until 200s) is no connection available.
- The connection on the testtrack (after 250) is worse than on the parking lot.

5.3.2 Amber

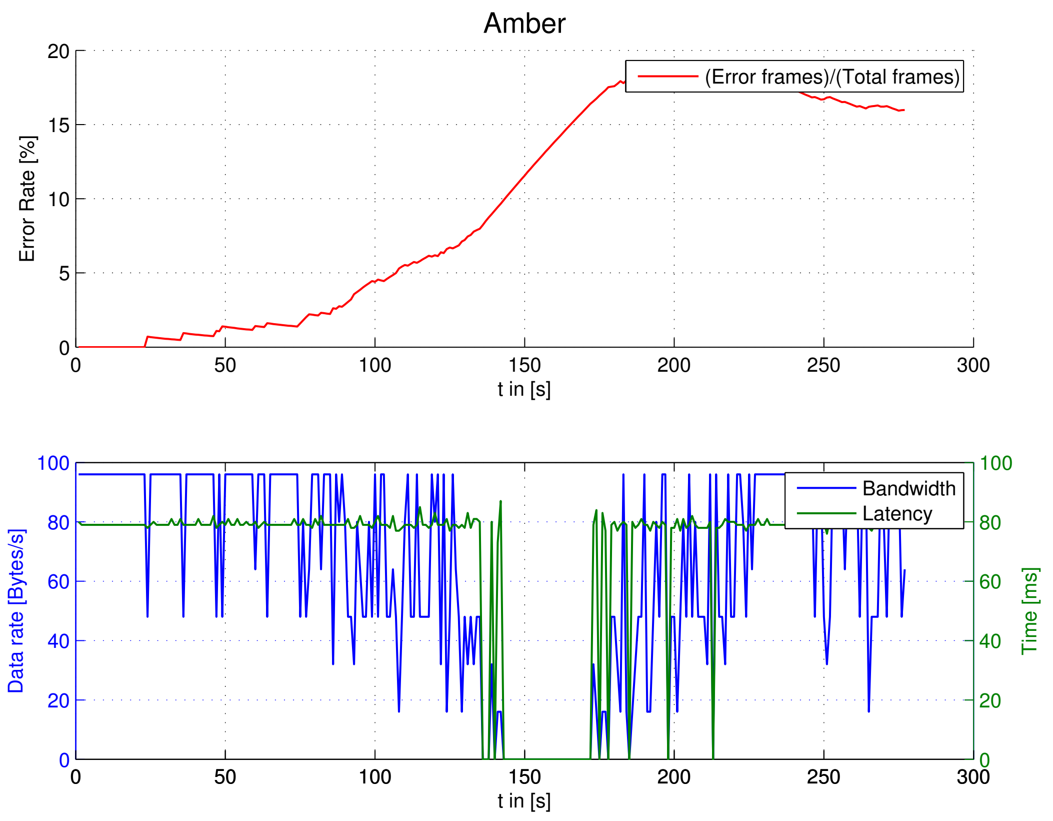


Abbildung 8: Amber (Outdoor)

- Nearly the same behavior as the zigbit modules → behind the exploration hall is no connection available
- The connection stability decreased after some meters on the parking lot

6 Average values

6.1 1m distance

	Zigbit	Amber
Average frame error [%]	0.35	0.25
Average bandwidth [bytes/s]	140	95.4
Average latency [ms]	49.93	83.5

6.2 10m distance

	Zigbit	Amber
Average frame error [%]	0.2	0.1
Average bandwidth [bytes/s]	140.95	95.5
Average latency [ms]	46.25	79.6

6.3 Outdoor

	Zigbit	Amber
Average frame error [%]	6.95	9.61
Average bandwidth [bytes/s]	99.36	63.8
Average latency [ms]	53.92	72.25

Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard

Ajish Babu*, Sylvain Joyeux*, Jakob Schwendner*, Felix Grimminger*

*DFKI Robotics Innovation Center, Bremen, Germany
e-mail: {firstname.lastname}@dfki.de

Abstract

Hybrid Leg-Wheel Robots have gained increasing popularity over the last years, as they can combine the terrain negotiation ability of legged systems with the efficiency and simplicity of wheeled systems. In this paper, the effects of locomotion patterns on the system's efficiency are studied for the legged-wheel robot Asguard. The locomotion pattern is controlled by setting the motion offsets between the wheels, and maintaining the synchronization during the motion using a cascaded position-velocity controller. The front-back and left-right wheel offsets are changed to generate different patterns. The efficiency of locomotion for these offsets, defined by specific resistance, is experimentally determined at different speed and terrain configurations. Experimental data was gathered with the help of a motion tracker system, to also analyze the vibration of the robot. The locomotion offsets showed a significant impact on the efficiency, especially at low speeds.

1 Introduction

The Asguard system (Figure 1) was developed at German Research Center for Artificial Intelligence (DFKI) with the purpose of being able to traverse through unstructured and uneven terrains. Currently, the robot is part of the project Intelligent Mobility, which aims at giving the robot ability to autonomously explore planetary or lunar surfaces. The robot's innovation lies in the hybrid leg-wheel (Figure 2) design, which is formed of five spike-like structures. The novelty of this design generates the need for an extensive study of the body configuration in order to develop a good understanding of the abilities and control requirements.

Leg-wheel hybrid designs have certain advantages over pure leg or wheel designs. First, its design is simple and sturdy, and is inherently stable compared to walking robots. Indeed, the robot, even without power, can stay standing as is the case with wheeled robots. Second, it can handle more difficult terrain than wheeled systems, as its leg-wheel has a better ground contact in most situations.

However, the mechanical linkages between the four

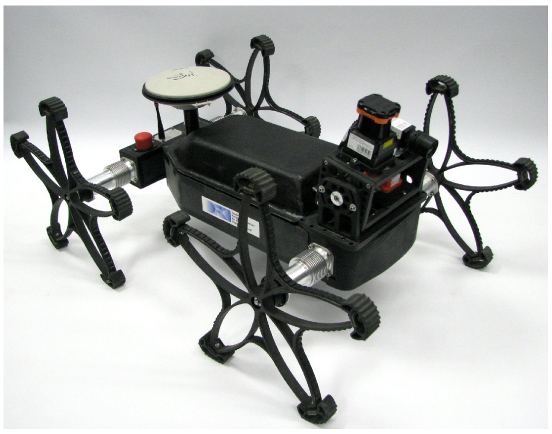


Figure 1. Asguard robot

legged-wheels removes the flexibility in foot placement that legged systems have. It also has some impact on the way the wheels should be controlled, as will be explained later.

Researchers have always tried different models of locomotion, mostly taking inspiration from the biological world. Study on the motion gait pattern of Pika [8] shows how gait pattern is used in locomotion. There is a noticeable change in locomotion pattern with change in speed in animals. This gait pattern and the angle of attack [2][8] transforms to the synchronization between the wheels in Asguard. The angle of attack is the angle at which the leg hits the ground. It becomes more important at higher speeds, at which the amount of bounce in the robot determines the optimum angle of attack. It might be possible to control the bounding behavior and the angle of attack through the synchronization. A bad synchronization could result in a higher angle of attack and thereby reducing the forward momentum. The scope of this paper is limited to lower speed ranges (0.2 to 0.5 m/s). At these speeds, the bounding of the robot and the effect of angle of attack is low.

Several works in the direction of locomotion study of robots have been previously undertaken. A preliminary controller for six-legged RHex is designed in [13]. An au-

Table 1. Specifications of Asguard

Length	95 cm
Width	51 cm
Height	44 cm
Leg length	19 cm
Mass	12.9 kg
Motors	4× 24 V DC motor
Gears	46:1 planetary gears
Body joint limit	-40° to +40°
Maximum speed	2 m/s

tomated gait adaptation of RHex has been studied in [16]. [12] depicts the bounding locomotion of Scout II. These locomotions and controller are not directly applicable to Asguard, due to the design differences. All of these publications use the dimensionless term *specific resistance* to assess the locomotion performance. The Gabriellivon Karman diagram [7] compares the specific resistance of a wide range of land vehicles.

Asguard has been the basis for numerous publications and theses. [3], [4] discusses the motivation and design of the robot. [5] proposes a compliance control architecture based on proprioceptive data, giving the robot ability to adapt automatically to different slopes. [11] studies the bounding behavior of the robot. [9] analyzes and optimizes the legged-wheel design.

This paper studies the locomotion of the robot with the objective of developing effective control strategies. The wheel design requires adapted control strategies, as simple velocity control designed for a wheel system works, but is not optimal. The challenge lies in trying to get a good performance by understanding the inherent physical abilities of the robot. Analytical analysis possibilities are limited due to several system complexities and nonlinearities. Experiments are performed here to gain an insight into the system properties.

The remainder of the paper is arranged as follows. Section 2 gives details of the mechanical, hardware and software design of the robot. Section 3 describes the locomotion principle and possible opportunities for improvement. Section 4 explains the experimental setup. Finally, the experimental results are presented in section 5.

2 Design of the robot

The robot body is 50 cm long with leg length of 19 cm. It can attain a maximum speed of 2 m/s. The robot is designed such that the weight distribution on the front axle (60%) is higher than in the rear axle (40%). The specifications of the robot are given in Table 1.

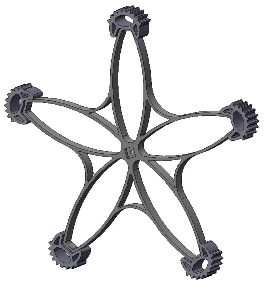


Figure 2. Legged-wheel design

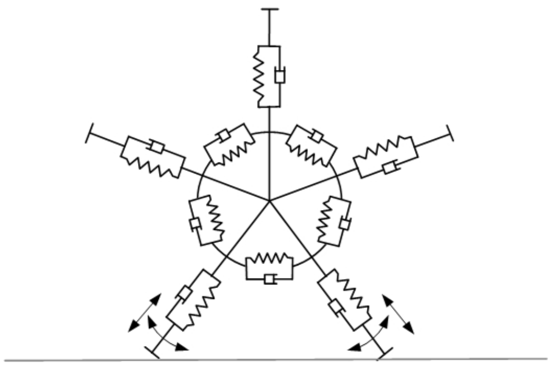


Figure 3. Wheel flexibilities

2.1 Mechanical and Hardware Design

The key components of the system are the legged-wheels, motor drives and the robot body. Two main sources of flexibility in the system are the legged-wheel and the flexible motor coupling. The legs have flexibility, both in linear and radial directions (Figure 3). In certain scenarios this results in bumpy behavior at higher speeds. The linear and radial spring constants and damping parameters are optimized in [9]. The flexible coupling acts as a shock absorber, reducing the jerks affecting the robot body.

The robot is also equipped with a passive joint, connecting the front part and the rear part of the body. This additional degree of freedom allows the robot to maintain contact with the ground with all four wheels even on uneven surfaces, which improves overall traction. The leg tips are connected to a soft foot, which is designed to increase longitudinal friction and reduce transverse friction, to improve skid-steering abilities.

The robot is fitted with PC104 stack and various sensors (IMU, cameras, laser scanners and time-of-flight cameras). Moreover, a differential GPS provides a ground truth from which it is possible to validate localization algorithms. The robot can operate autonomously or can be controlled manually with a custom designed control-pad

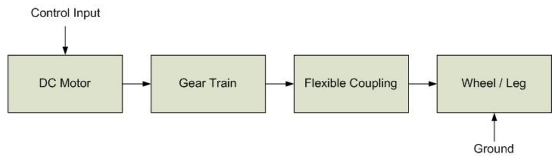


Figure 4. Wheel drive model

or a joystick.

The wheel motors are 24 V brushed DC motors with 46:1 planetary gear systems. Custom designed H-Bridge motor boards are used to drive the motors. The drives are connected with incremental encoders for feedback control, which give a tick resolution of around $70\mu rad$ on the wheel side. The motor is connected to the wheel through a planetary gear system and a flexible coupling (Figure 4), which makes the estimation of the actual wheel position very difficult. The coupling also introduces additional motion effects between the motor and the wheel.

2.2 Software

The control loop is implemented on the embedded PC104 system, implemented in the Orocos realtime framework, on top of a Linux with RT-PREEMPT patches. The communication between the PC104 and the H-bridge hardware (which controls the motors) is using a CAN-bus. Latency measurement showed that the control loop latency (between the sensors and the actuator) is below 3 ms for a control frequency of 1 kHz.

The software also includes a complete realtime data logging part, which allows to save all the evolution of the control loop at its running frequency (1 kHz)

2.3 Motion Controller

Proportional Integral Derivative (PID) based velocity controllers tend to unintentionally change the wheel synchronization due to load disturbances. It tries to maintain the speed of the wheel, but fails to account for the effect of disturbances on the position. So, the asguard motor is controlled by using cascaded position and velocity controllers known as the PIV controller [10]. The position controller helps in maintaining the synchronization between the wheels. The reference position and velocity profiles, generated by the trajectory generator, acts as the input to this controller. Additionally, it is provided with a first order velocity smoothing. Provisions for velocity and acceleration feed-forward are also given. This controller allows us to study the effect of inter-wheel synchronization on the overall locomotion performance, which is presented in the paper.

PIV controller consists of three cascaded control loops. The outermost position loop, inner velocity loop and the innermost current control loop (inherent in the motor). The position loop works with a proportional gain and the velocity loop with a proportional-integral gain. The controller outputs the Pulse Width Modulation (PWM) command to the H-Bridge board, which drives the motor. The feedback obtained is the position information from the incremental motor encoders. The advantages of PIV controller are as follows.

1. Non-linearities in the inner loop does not affect the outer loop
2. Good disturbance rejection capabilities
3. Easy to tune

The synchronization is performed by starting the robot with all wheels in the same position to calibrate the wheel position. The position offsets are then applied to the individual trajectories. The tuned PIV gains are given in Table 2.

Table 2. PIV controller gains

Position proportional	3.80
Velocity proportional	0.07
Velocity integral	0.65
Integral windup	0.06
Velocity smoothing	0.6
Velocity feed-forward	1.0
Acceleration feed-forward	0.0

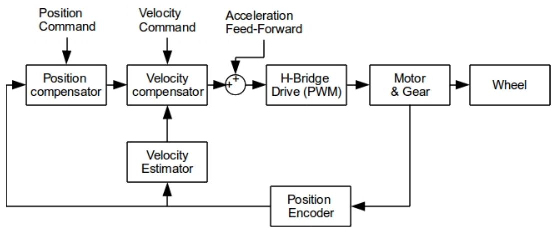


Figure 5. Overview of PIV controller

2.4 Effect of Wheel Synchronization

If from the controller point of view, the robot is considering a wheeled system, neglecting its legged properties, the result is a vibrating system with inefficient locomotion. The wheels can be synchronized to have a particular relative orientation at an instance in time, thereby emulating gait patterns of legs. This influences the energy transfer between different parts of the robot, effectively smoothening the locomotion and improving its efficiency.

3 Locomotion of the robot

In legged robots, the leg motion alternates between *swing* phase and *stance* phase. Simple velocity controllers for Asguard occasionally resulted in stance phase and swing phase blocking [4] each other and at times flipping the robot. Asguard can have these two phases simultaneously. Primarily, the legged-wheels have two alternating positions

1. Vertical stance
2. Double-contact stance.

Vertical stance is the position where only one leg is in contact and is perpendicular to the ground. At double stance two legs are in contact with the ground at the same instance. The blocking of the swing phase happens at the double-contact stance. The impact of this blocking can be reduced using effective synchronizations.

3.1 Improving locomotion

The locomotion can be improved by maintaining a smooth flow of energy within the system. We consider two scenarios to explain the principle behind the locomotion improvement.

In the first scenario, assume the leg-wheels are moving in perfect synchronization (no offsets). At the vertical stance, there is an additional potential energy in the robot owing to its height. This potential energy is converted to kinetic energy as the legged wheel rotates further. But this kinetic energy is abruptly removed when it reaches the double contact stance, creating unnecessary vibrations. Additionally, to move from the double stance to the vertical stance, the motor should provide additional energy and lift the robot up.

In the second scenario, assume the Front-Left wheel is offset to Back-Left by $\frac{\pi}{5}$. Similarly, Front-Right wheel is offset to Back-Right wheel. When the front wheels are in vertical stance, the back wheels are in double stance. As the wheels rotate the potential energy from the rear part gets transformed to kinetic energy. This kinetic energy assists in overcoming the double stance block and also to gain the potential energy for the vertical stance of the front part. The transfer of energy between the states should have an impact on locomotion efficiency, which is also likely to change depending on system velocity and surface material properties. The same principle might be applied for left and right wheels.

We define three sets of offsets (Figure 6).

1. Front and Back wheels ϕ_{FB} (FB-offset)
2. Left and Right wheels ϕ_{LR} (LR-offset)
3. Left and Right cross wheels ϕ_{LR} (LR-cross-offset)

The FB-offset is the difference in angle between the front wheel and the rear wheel (Figure 6: top-left). LR-offset is the offset between left and right wheels (Figure 6: top-right). LR-cross-offset is same as the LR-offset, though in different directions (Figure 6: bottom).

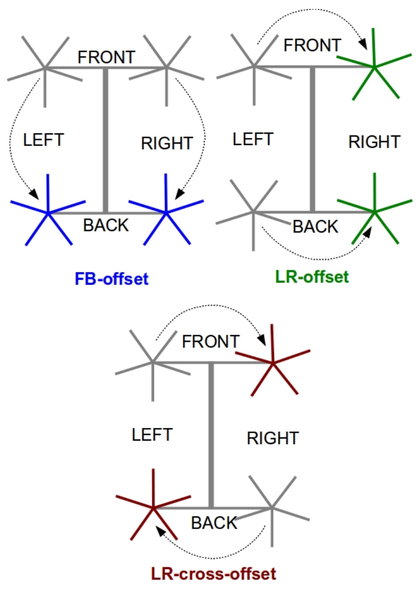


Figure 6. Wheel offsets

It is not possible to maintain a left-right synchronization while turning using skid steering. So, more importance is given to front-back offsets. The aim is to find the most efficient front-back offset and study the change in efficiency for different left-right offsets.

3.2 Locomotion Efficiency

Locomotion can be evaluated using specific resistance or Froude number. The term specific resistance for locomotion was introduced in [6]. It is defined as the energy consumption per unit distance per unit weight[14] [7].

$$\epsilon = \frac{E}{Mgd} \tag{1}$$

where E is the energy consumed, M is the mass of the vehicle, g is the acceleration due to gravity and d is the distance traveled. It has since been used in numerous literatures to compare wide range of locomotions.

Froude number [14] is usually used to characterize animal locomotion. It is calculated as

$$F_r = \frac{V^2}{gh} \tag{2}$$

where V is the velocity of walking or running, g is the acceleration due to gravity and h is the height of the hip joint from ground.

Specific resistance is more suited for analyzing robot motions and hence will be used in this work. To put the specific resistance (ε) values in perspective [14], cars have a specific resistance in the range 0.07 to 0.4. Specific resistance for a human walking is between 0.02 to 0.1. RHex robot using automated tuning methods obtained a specific resistance of 0.6 [16]. Quadruped Robot running with Bounding Gait [15] had a specific resistance of 0.32. McGeer's gravity walker [14] shows the lowest specific resistance among walking robot at 0.01. Among powered legged robots, ARl Monopod [7] with ε=0.7 shows the lowest specific resistance.

The energy consumed in our case is calculating the integral of power consumed by the motor. So the total energy is given by

$$E = V_{app} \int I(t)PWM(t)dt \tag{3}$$

where V_{app} is the battery voltage, $I(t)$ is the current measured at the motor and $PWM(t)$ is the PWM input given to the motor. When the current is measured at the source, the equation becomes,

$$E = V_{app} \int I(t)dt \tag{4}$$

3.3 Error propagation in Specific Resistance

The equation for specific resistance (1) can be used to estimate the error propagation due to measurement inaccuracies. Error propagation[1] for multiplication is given by

$$(x \pm \delta_x)(y \pm \delta_y) = (xy) \pm (\sqrt{y^2\delta_x^2 + x^2\delta_y^2} \tag{5}$$

and error propagation for division is given by

$$\frac{(x \pm \delta_x)}{(y \pm \delta_y)} = \frac{x}{y} \pm \left(\sqrt{\left(\frac{x}{y^2}\right)^2 {\delta_y}^2 + \left(\frac{1}{y}\right)^2 {\delta_x}^2} \right) \tag{6}$$

Average value and expected errors for $energy = 300.0 \pm 30.0$ Nm, $distance = 8.0 \pm 0.25$ m and $mass = 13.0 \pm 0.1$ kg. Using (5), (6) and the average values given above, the error propagation was estimated to be approximately 10.0%.

3.4 Vibrations

Vibrations on the robot body can affect the proper functioning of the robot sensors and can damage the components in the long run. Unnecessary vibrations reduce the efficiency of locomotion and is therefore an interesting parameter to investigate as it gives indications as to why some configurations are better than others. The amplitude and frequency of the vibrations should be as low as possible. In this work, the vibration was analyzed only for limited configurations and only important results are presented. This is due to the limited field of view of the motion capture system.

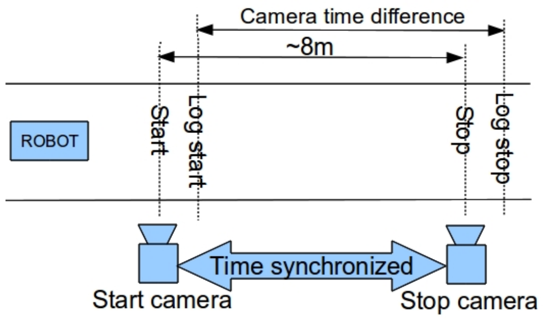


Figure 7. Experimental Setup

4 Experiments

Figure 7 illustrates the experimental setup. Two time-synchronized cameras are used to measure the time difference between the start and finish. This time difference is used to extract the corresponding data from the log. The actual data extracted is the log from log start to log stop (Figure 7). The tests were performed on both hard ground and sand. Hard ground has low damping compared to the legs and sand has high damping effect compared to the legs. Robot forward velocities are taken in the the ranges low (~0.2 m/s), medium (~0.4 m/s) and high (~0.5 m/s). At velocities higher than 0.5 m/s, the robot tends to deviate from the designated track.

The offsets can range from $\frac{-\pi}{5}$ to $\frac{\pi}{5}$, which covers the angle between two adjacent legs. Due to the symmetry of the robot, the range can be limited between 0 and $\frac{\pi}{5}$. For FB-offset and LR-cross-offset, the values 0, $\frac{0.5\pi}{5}$ and $\frac{\pi}{5}$ were used. For LR-offset, the values 0, $\frac{0.25\pi}{5}$, $\frac{0.5\pi}{5}$, $\frac{0.75\pi}{5}$, and $\frac{\pi}{5}$ were used. The offsets will be represented only as multiples of $\frac{\pi}{5}$ in the rest of the discussion and plots.

The robot logs the battery voltage, PWM input and current at 1 kHz frequency. The data logged during the time difference measured by the camera is used to calculate the specific resistance. The length of the track which is measured for each test setup, is approximately 8 m from start to finish.

Separate experiments were conducted using a 3D motion capture system to capture the vertical vibrations of the robot. The motion capture system from Qualysis captures the X-Y-Z motion of the robot at 100 Hz using a single marker. The three IR cameras are placed such that the cameras have an intersecting field of view along the track.

5 Results

In general, increasing the FB-offset and LR-cross-offsets resulted in improved efficiency. Figures 8 and 9 show the effect of increasing the FB-offset on sand and

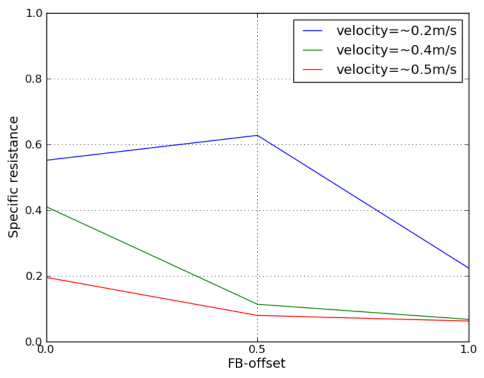


Figure 8. ε **vs FB-offset on hard ground**

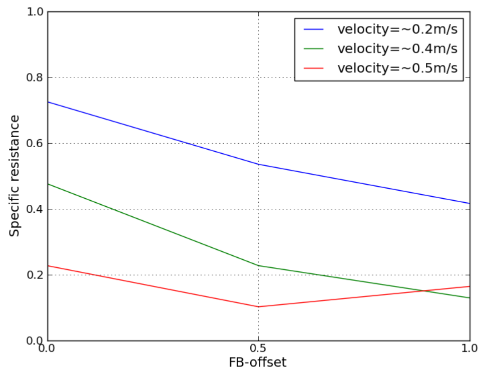


Figure 9. ε **vs FB-offset on sand**

hard ground respectively. At 0.2 m/s speed, increasing FB-offset, reduces specific resistance from 0.72 to 0.22 on sand. In similar situations on hard ground the value changes from 0.55 to 0.27. An exception to this trend was found on hard ground at 0.5 FB-offset and 0.2 ms speed (Figure 8). The specific resistance here is higher than the lower offset. This value seems to deviate from the trend and further investigation is required to determine its cause. Figures 10 and 11 show the effect of increasing LR-cross-offset. The trend here is similar to that of FB-offset.

The effect of change in specific resistance is dependent on speed. Lower speeds show higher response to change in offsets. But higher speeds are more efficient, with or without offsets.

Figures 12 and 13 show the effect of FB-offset=1.0 and varying the left-right offset. The results when FB-offset is 1.0 and varying left right offsets shows no appreciable change in efficiency. This shows that a FB-offset of 1.0 alone could be used to improve the locomotions efficiency.

The type of ground does not seem to significantly impact the specific resistance trends. Although hard ground slightly reduces the specific resistance compared to sand, due to the damping effect of the sand. The variance of the specific resistance was not determined and occasionally, some outliers were also found.

An example of the vibration differences is shown in Figure 14. At a LR-cross-offset of 1.0, the amplitude of vibrations is reduced to a great extend (25 mm lesser) when compared to 0.0 LR-cross-offset. But additional rotational motion (roll) is induced on the robot.

6 Conclusion

In this paper, we investigated a way to improve the overall locomotion efficiency at low speeds of a legged-

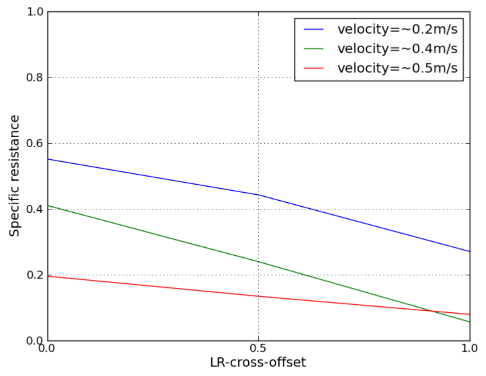


Figure 10. ε **vs LR-cross-offset on hard ground**

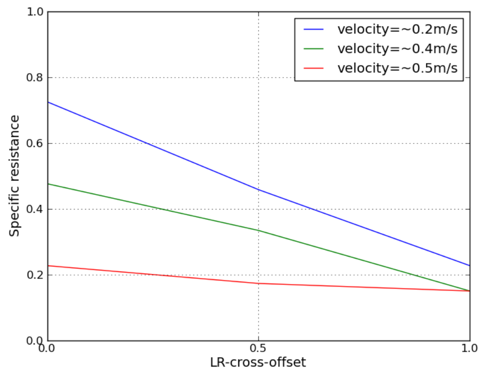


Figure 11. ε **vs LR-cross-offset on sand**

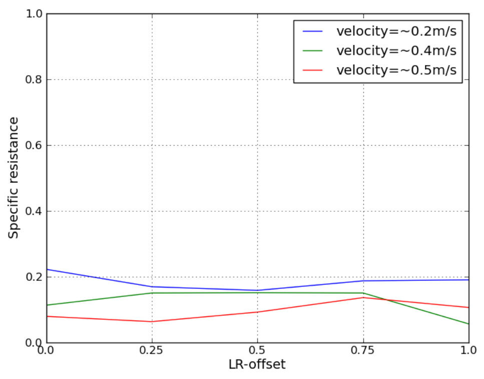


Figure 12. ε vs LR-offset on hard ground with FB-offset = 1.0

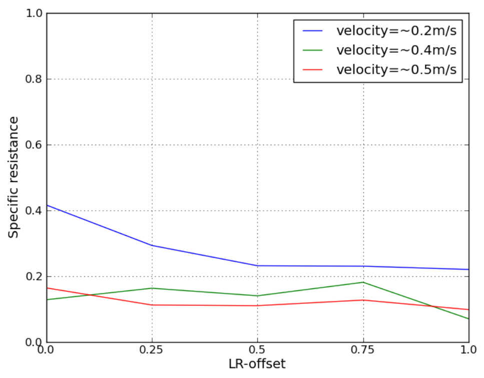


Figure 13. ε vs LR-offset on sand with FB-offset = 1.0

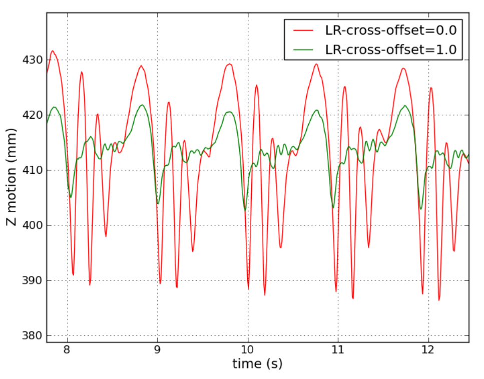


Figure 14. Vertical motion at 0.2m/s

wheel system like Asguard. Our approach tried to reduce the energy loss during locomotion by improving the potential and kinetic energy exchange between parts of the robot. We achieved this by controlling the wheel synchronizations. The lowest specific resistance was achieved with a complete left-right-cross offset at 0.4 m/s speed on hard ground. The experimental results showed that it was possible to efficiently control Asguard in a way compatible with its skid-steering nature.

In future, automated method for determining efficiency could be developed to extend the study to higher speeds and different patterns.

7 Acknowledgment

The work presented here is part of the Project "Intelligent Mobility", which is funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907.

This work would not have been possible without contributions from Christopher Gaudig and Kai von Szadkowski.

References

[1] Appendix v: Uncertainties and error propagation. Technical report, Case Western Reserve University, September 2004.

[2] Reinhard Blickhan, Andre Seyfarth, Heiko Wagner, Arnd Friedrichs, Michael Günther, and Klaus D. Maier. Robust behaviour of the human leg. In *Adaptive Motion of Animals and Machines*, pages 5–16. Springer-Verlag Tokyo, 2006.

[3] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner. Asguard: A hybrid legged wheel security and sar–robot using bio–inspired locomotion for rough terrain. In *IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance*, January 20088.

[4] M. Eich, F. Grimminger, and F. Kirchner. Adaptive stair–climbing behaviour with a hybrid legged–wheeled robot. In *11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, September 2008.

[5] Markus Eich, Felix Grimminger, and Frank Kirchner. Adaptive compliance control of a multi-legged stair-climbing robot based on proprioceptive data. In *In Industrial Robot: An International Journal*, number volume 36 Issue 4, pages pages 331–339. Emerald Group Publishing Limited, 2009.

[6] G. Gabrielli and T. H. von Karman. What price speed? In *Mechanical Engineering*, volume 72, pages 775–781, 1950.

[7] P. Gregorio, M. Ahmadi, and M. Buehler. Design, control, and energetics of an electrically actuated legged robot. *IEEE Trans. Systems, Man, and Cybernetics*, 27:626–634, 1997.

[8] Remi Hackert, Hartmut Witte, and Martin S. Fischer. Interactions between motions of the trunk and the angle of attack of the forelimbs in synchronous gaits of the pika (ochotona rufescens). In *Adaptive Motion of Animals and Machines*, pages 69–77. Springer-Verlag Tokyo, 2006.

[9] Philippe Högemann. Biologically inspired optimization of the hybrid quadruped asguard robots legged-wheel design, December 2008.

[10] David Kaiser. Fundamentals of servo motion control. Technical report, Parker Compumotor, Rohnert Park, California, October 2001.

[11] Janosch Machowinski. Dynamisches laufen mit dem system asguard, August 2009.

[12] Ioannis Poulakakis, James Andrew Smith, and Martin Buehler. On the dynamics of bounding and extensions: Towards the half-bound and gallop gaits. In *Adaptive Motion of Animals and Machines*, pages 79–88. Springer-Verlag Tokyo, 2006.

[13] Uluc Saranli, Martin Buehler, and Daniel E. Koditschek. Design, modeling and preliminary control of a compliant hexapod robot. In *IEEE Int. Conf. Robotics and Automation*, pages 2589–2596, 2000.

[14] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008.

[15] S. Talebi, I. Poulakakis, E. Papadopoulos, and M. Buehler. Quadruped robot running with a bounding gate. In *Proc. 7 th Int. Symp. on Experimental Robotic (ISER00*, pages 281–289. Springer-Verlag, 2000.

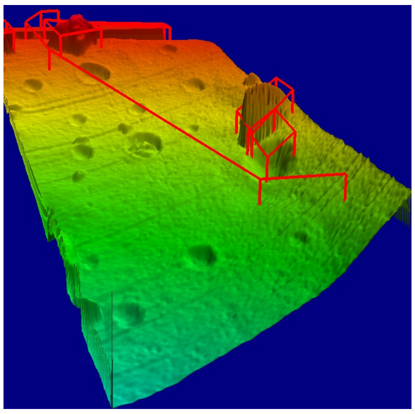
[16] Joel D. Weingarten, Gabriel A. D. Lopes, Martin Buehler, Richard E. Groff, and Daniel E. Koditschek. Automated gait adaptation for legged robots. In *IEEE Int. Conf. Robotics and Automation (ICRA), New Orleans, LA*, April 2004.

AG Navigation and Planning

- Main focus until now: Navigation
  - Environment representation
  - Localization
  - Mapping
  - Simultaneous Localisation and Mapping (SLAM)
  - Path/Motion planning

Environment Representation

- Deliberative systems require Environment Representation (ER) for path planning/motion planning
- ER is required for localisation and mapping (and of course SLAM)
- A lot of algorithms exist to operate on ER, but there are a limited number of ways to represent an environment
- Developing libraries that can be used by different projects can lead to:
  - Easier transfer of Algorithms
  - A common visualisation



Requirements

- Requirements have been collected from the different projects
- This list is translated into a description of
  - Types of Maps to represent
  - Relationships between the Maps
  - Common Services and Attributes

The Environment Representation is a model of the world defined by several different maps and their relations to each other

Maps as functions

- Maps are nothing but functions in a mathematical sense
- The input of the function is the spatial component (e.g. 2D, 3D)
  - E.g. $z = f(x, y)$
- The output depends on the type and dimension of data

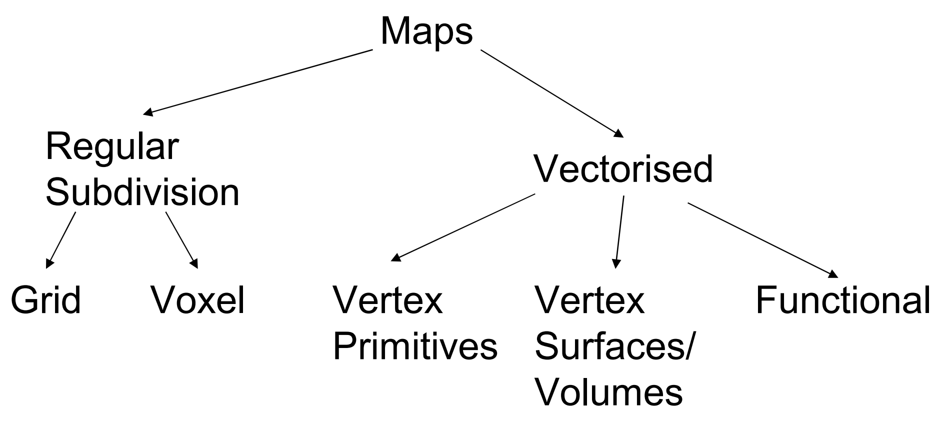
- Maps provide either
  - Attributes for a certain region in space
  - A region of space that belongs to a certain group/attribute (inverse function)

Output Space

- All maps can have output spaces
  - Discrete sets (classes)
    Example: Grass, Sand ...
    Example: traversability classes
  - Real values
    Example: elevation maps
  - Real value tuples
    Example: probabilistic elevation
  - Arbitrary Objects e.g.
    - maps of feature descriptors (SIFT)
    - multi-level surface maps (Sets of intervals)

Universität Bremen

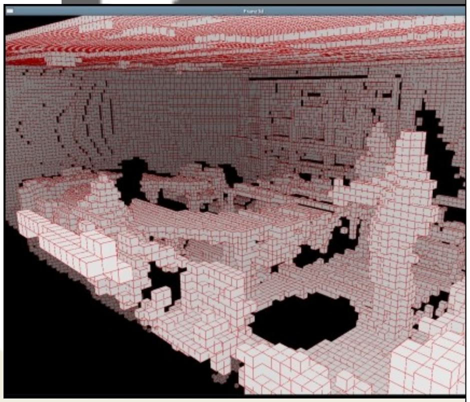
Maps: Taxonomy



Universität Bremen

Regular subdivision of space

- Grid Map (2D)
  - Input space is discrete, regular and usually bounded
  - Example: Occupancy Grid
- Voxel Map (3D)
  - Same as Grid map, but in 3D
  - Example: Occupancy map

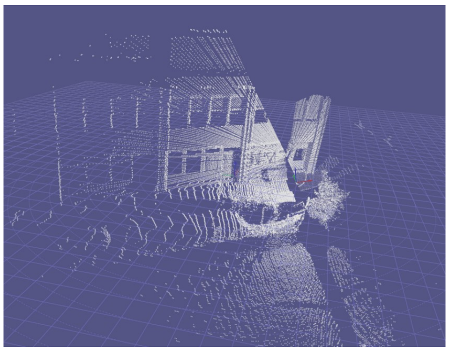


Universität Bremen

10

Vectorised: Vertex Primitives

- Set of Vertices (2D)
  - Input continous 2D space
  - Example: 2D – Laserscan
- Set of Vertices / Pointcloud (3D)
  - Input continous 3D space
  - Example: 3d – Laserscan / Stereovision

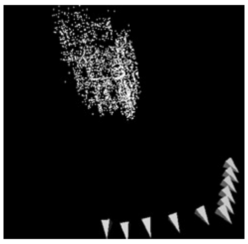
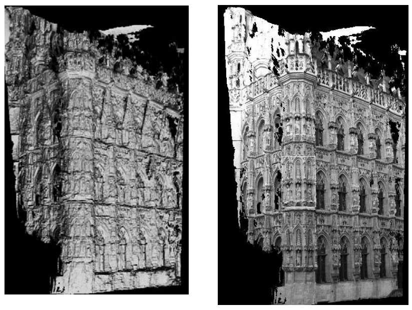


Universität Bremen

11

Vectorised: Surfaces/Volumes

- Use Vertex Primitives
- Triangular Meshes (3D)
  - Connecting three vertices to form triangle
  - Closed – Represents volume of space
  - Open – Representing Surfaces
- Polygons (2D)
  - Connecting two vertices
  - Closed – Representing area
  - Open – Representing lines

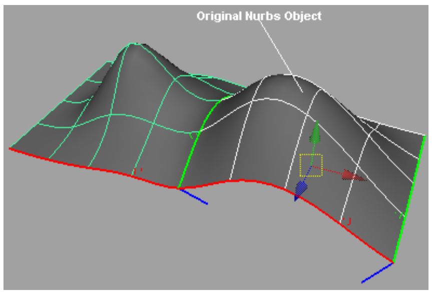



Universität Bremen

12

Vectorised: functional

- Use functions to represent surfaces/volumes
- Functions can either describe surfaces or volumes
- Surfaces
  - Example: NURBS objects
- Volumes
  - Example: Height map
    z = f(x, y)
  - Example: Density maps
    d = f(x, y, z)
- Pro: compact
- Cons: complex handling

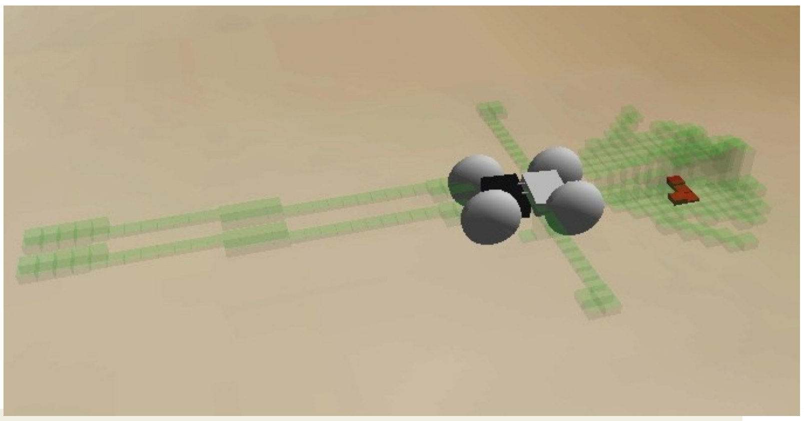


Universität Bremen

13

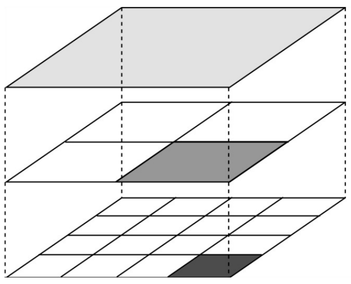
Maps: Additional notes

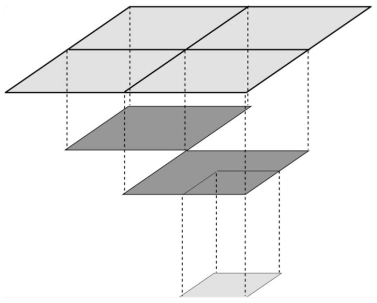
- Maps should be able to represent
  - absence of knowledge
  - knowledge about free space (e.g. laser's out of range readings)



Hierarchical maps

- structures maps of identical type (same input and output spaces)
- represented in a tree structure, with a child being more detailed than the parent
- standard "level of details" setup is contained there, with the added constraint that all maps have the same boundaries.





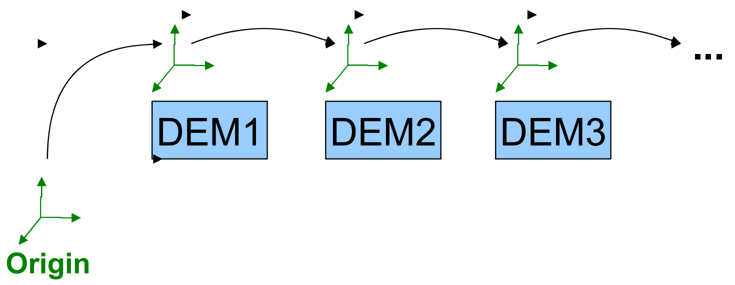
Map relations

- We want to represent how maps relate to each other
- Identified three separate concerns:
  - Maps are expressed in a given frame of reference
    - Represent it: **Frame tree**
  - Some maps are generated from a set of input maps
    - Represent the generation process: **Operator graph**
    - examples: projection, merging, distortion, ...
  - Some areas can be known with more details than others
    - Multiple level of details: **Hierarchy tree**

Implementation: FrameTree

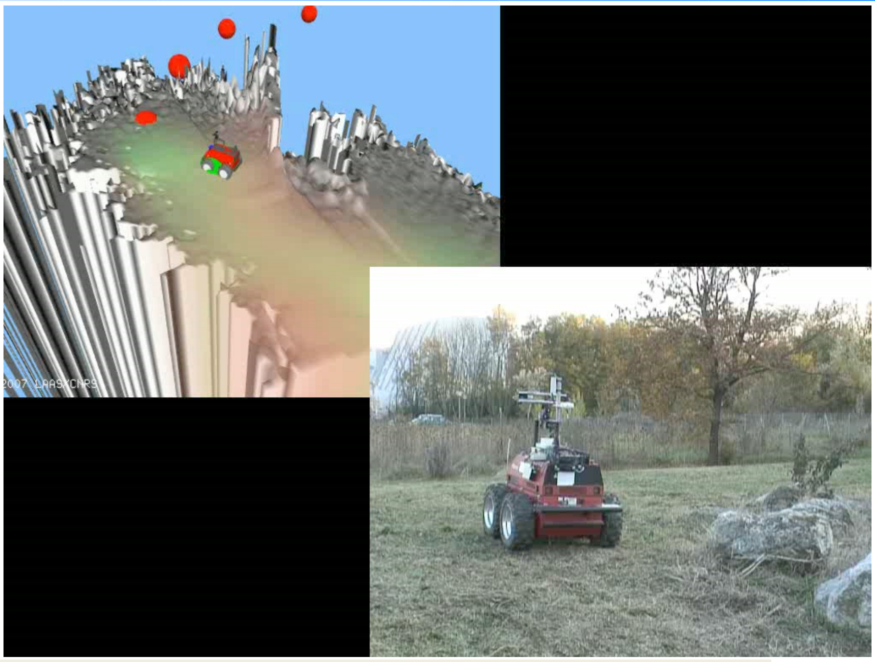
- Tree for transformation between maps
  FrameTree: manages FrameNode Objects
  - FrameNodes store Isometry Transform relative to parent
  - Always possible to compute transformation between any two FrameNodes in the same FrameTree

Example: acquisition of digital elevation maps



→ **FrameTree**: change in pose of the robot between the acquisition of two DEMs

Example: DEM fusing



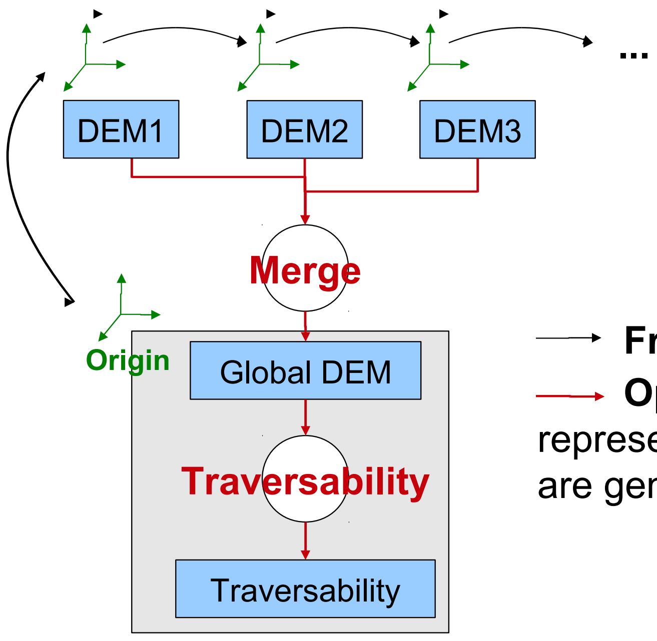
Implementation: Operators

- Basic operator interface [input_maps, parameters, output_maps]
- Examples
  - Projection: takes a 3D layer and produces a 2D layer
  - Transformation3D: takes a non-grid 3D layer as input and applies an arbitrary transformation to its contents.
  - Merging: takes a set of input maps of identical types and produces a merged version of them.
- Operators can be managed in a graph
  - Different update strategies (manual, automatic, lazy)

Universität Bremen

20

Example: Traversability map



→ **FrameTree**

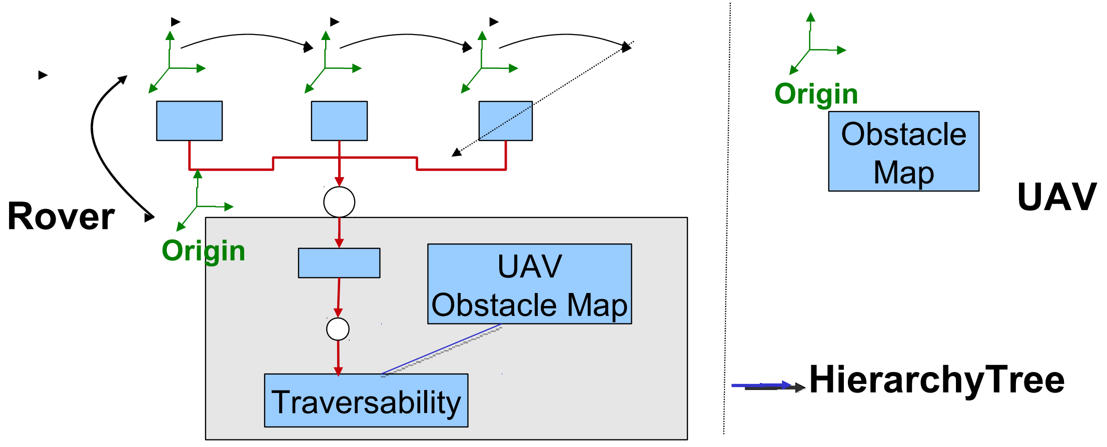
→ **OperatorGraph**: represents how some maps are generated from other maps

Universität Bremen

21

Implementation: Hierarchy Tree

- Tree for spatial hierarchy of maps (HierarchyTree)
  - Represent hierarchical maps (no spatial relation)
  - Each map has at most one parent



Universität Bremen

Visualisation: Requirements

- Data to display
  - Visualisation of all map types from ER
  - Relations between maps (e.g. Listview)
- How to display
  - nice aesthetics (e.g. options for subsampling/lighting/transparency, skinning)
  - Different types of data in one view (e.g. pointcloud + mesh + features)
  - project 2d-map textures onto 3d-objects
  - visualisation aids (grids, axis, horizon, ...)
  - ability to change visalisation attributes of object

Universität Bremen

Visualisation: Requirements (2)

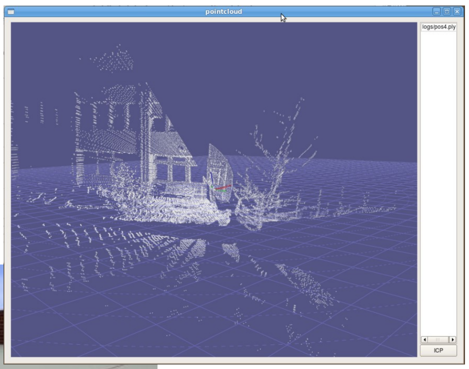
- Camera
  - Control of the camera should be powerfull/intuitive
  - Camera needs to be able to follow objects
- Manipulation/Selection
  - Simple manipulation (rotation/translation/scaling)
  - Manipulate lighting
  - Subregions of the data need to be selectable
- Show information
  - display information about objects (e.g. number of primitives, meta-data, comments)
  - option to display data next to the object

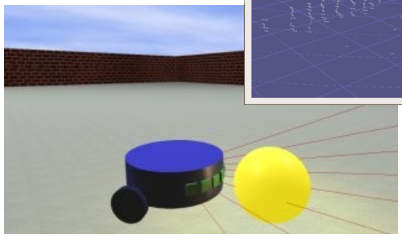
Visualisation: Requirements (3)

- Import/Export
  - export of subregions/subselections
  - export of images/videos
  - save state of views (camera, lighting, display of metadata a.s.o)
- Performance
  - efficient dynamic update of live data

Visualisation: Implementation

- Needs to be separated from Environment Representation!
- Could be based on existing applications
  - MARS
  - OSG
  - GameEngine (Ogre, ...)
- Or new development (OpenGL)
- Criteria
  - Lightweigt
  - Modular
  - Easy to adapt





Universität Bremen

The way to collaborate

- There are practical issues to inter-project code sharing
- Open practical issues:
  - Build system (cmake, automake)
  - Revision control system (svn, git, mercurial)
  - Source structure (package-like, monolithic)
- Discussion moved from AG-NavPlan to AG-Framework

- Decision for the environment representation
  => in the next 2 to 3 weeks **(not more !)**

Upcoming tutorial: git, cmake (next week or so)

Universität Bremen

Conclusion

- Requirements for Data Structure and Visualisation are there
- Starting points for the code exist
- Process needs to be put in place on how to collaborate

-> Great opportunity to have interproject co-operation on a code level

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovations Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Self Localisation using Embodied Data for a Hybrid Leg-Wheel Robot

Jakob Schwendner and Sylvain Joyeux

Abstract— **Robotic systems that are able to navigate autonomously in unstructured outdoor terrain have a large potential in a number of applications like planetary exploration or search and rescue scenarios. Localisation is usually performed through dead-reckoning with the help of visual means. The approach described in this paper uses only sensory information internal to the system to localize in a partially known environment. A model of the robot and sensory information on its configuration and orientation are used to match candidate contact points with an environment model. A particle filter implementation is developed, which uses this information together with the odometry to track the pose of the robot. The experiments conducted on a hybrid leg-wheel robot show that the approach is able to track the position of the robot within an average error of 0.5 m for test runs of up to 140 m distance travelled. One potential of this approach is to reduce the requirements on the visual parts when integrated into SLAM frameworks.**

I. INTRODUCTION

Mobile outdoor robots have a large potential to aid in the challenges we will be facing in the next decades. Space exploration, search and rescue, agriculture, environmental monitoring and security are but a few application areas where the contributions from automated mobile systems are likely to become more important in the future. Automated systems have long been confined to strictly controlled environments, like for example in manufacturing or warehouse settings. Partly structured or completely unstructured environments, as is usually the case in outdoor applications, are much more difficult to handle from a locomotion, manipulation and cognition point of view [1]. One aspect which is of key importance to mobile systems is navigation. The task of navigation in general is the ability to go from A to B. In order to do that, the system has to localise itself and perform actions (e.g. moving) which change its current state. For both cases maps are helpful, but not strictly necessary. However, while in principle one can get from A to B without any localisation at all, reaching B is then a matter of pure chance.

Localisation for mobile robots has been extensively researched for the 2D case, which can be applied to structured indoor environments [2]. For outdoor environments, the Global Positioning System (GPS) is often a good way of providing means of localisation. However, in some situations, like urban canyons or tree canopies, GPS does not necessarily provide the accuracy required. It may not even be available at all, as is the case in a planetary exploration setting. The main alternatives to GPS for localisation of mobile robots are usually vision based or use some form of laser range finding [1]. What we argue in this paper is that the interaction between the robot body and the environment is neglected in these approaches, interaction that can provide useful information for localisation [3]. Such data is often described as proprioceptive data in the literature, which is a term borrowed from biology. However, when applied to technical systems, this categorisation is sometimes ambiguous. To avoid any confusion, the term *embodied data* is used here instead. Embodied Data is defined as sensory information originated within or on the border of the system in question. Two classes of embodied data are defined in [3]. Direct embodied data is the immediate form, which includes information like environment contact data or joint angle readings. Indirect embodied data on the other hand includes locomotion properties like robot speed or energy efficiency.

German Research Center for Artificial Intelligence (DFKI), Robotics Innovation Center (RIC), Bremen, Germany
{firstname.lastname}@dfki.de

The work described in this paper is concerned with the problem of robot localisation in outdoor environments using only direct embodied data. In principle, this is comparable to e.g. finding your way in a known, hilly terrain at night. A map of the environment is available, and the task is to use information about your orientation with respect to gravity and environment contact information to create a best guess of your current position.

In the following section, some related works are presented and put into context. The methods sections describes the experimental setup and the model of the robot and the environment, as well as the odometry and measurement model. Further, a description of a particle filter is given, which is used to track the position of the robot. The experiments that were conducted on the Asguard v3 [4] system are presented in the results section.

II. PREVIOUS WORKS

The approach presented in this paper uses an a-priori map and measurements relating the robot to that map in order to estimate its current position. Works like [5] and [6] have addressed this problem in the context of autonomous flight system for military applications. These Terrain Based Navigation approaches usually employ some kind of optimal filter to estimate the flight path given inertial navigation, a digital elevation map and distance to ground using a radar sensor. More recent works use non-linear bayesian filtering to improve on the performance of the earlier systems. [7] uses a point mass filter, which satisfies the Cramer-Rao lower bound. Similar performance with lower complexity and additional flexibility has been achieved by employing particle filters [8] for the non-linear filtering. [9] applied a particle filter in an underwater scenario, which in terms of sensor information is comparable to the aerial application. All of these works use exteroceptive information in order to perform the localisation in the known environment.

Ground based vehicles have a different sensor profile. They naturally track the height profile of the environment, since they are in constant contact with it. Localisation using exteroceptive sensors has been extensively researched (see e.g. [10]). The use of proprioceptive or embodied data for aiding localisation and mapping tasks has not received so much attention.

Hoffman and Krotkov [11] performed some work on the walking robot Ambler to generate a map of the environment using a laser range finder. They use a feedback loop for controlling the map error directly underneath the feet of the robot, and by doing so reduce the overall map error. They use a forward kinematic model to calculate the foot positions and compare these values to the DEM values. The resulting error is used to correct the map height. By doing so the authors were able to eliminate a drift in map error. The work shows some early examples of embodied data use for localisation, limited to the relative height component of the map.

Chitta et al. [12] have investigated the use of proprioceptive and orientation data for the four legged robot LittleDog (dimensions are 30 cm x 18 cm x 15 cm). The approach uses a particle filter to track the 2D pose (x, y, θ) of the robot. Experiments have been performed in a controlled environment on a 120 cm by 60 cm terrain board, where the approach showed a significant improvement over using odometry data alone. Chitta et al. are using an environment model based on Digital Elevation Maps (DEM), which can only encode simple surfaces and are not able to handle any form of multiple levels or uncertainty. Further, the absence of data in the regions the robot is passing through is also not covered.

III. METHOD

A. Experimental Setup

The experiments to verify the described approach have been performed on the Asguard v3 platform (see Fig. 1), which is being developed at the DFKI Robotics Innovation Center in the course of the on-going "Intelligent Mobility" project [4]. Asguard [13] is a hybrid leg-wheel robot [14], [15] with four drive actuators, which has been designed to be able to traverse difficult outdoor terrain. Each of the four wheels has five feet attached to them, and the main body is connected to the rear by a one degree of freedom passive joint (body twist joint). The role of this passive joint is to ensure that all wheels have contact with the ground at all times. Three sensors on the system are relevant for the experiments. Encoders provide the absolute angular positions of the wheels with respect to the body and the position of the passive joint. A MEMS (Micro Electro Mechanical System) based inertial IMU is attached to the front body part for providing orientation readings. Finally, a RTK (Real-Time Kinematic) GPS is used to track the position with an

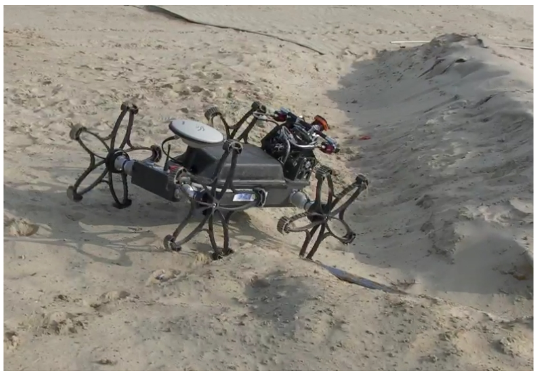


Fig. 1. The Asguard v3 [4] platform has a similar mechanical design compared to the Asguard v2 [13], but includes additional sensors and on-board processing. The main sensors used for this work are the wheel and joint encoders and an IMU. An RTK GPS was used for providing reference measurements.

accuracy of up to 1 cm (under certain conditions). The GPS is not used in the localisation approach, but only for ground truth measurement. Note that touch sensors have not yet been used for this work, and that our method therefore has to estimate the contact points. This is possible if the assumption is made that the robot is in contact with the environment at all times. This assumption will be discussed later in this paper.

For verification of the approach a number of trial runs have been recorded on a 30 m x 50 m sand field, which has variations in height of around 1 m and possesses small local features like trenches and hills (see Fig. 3). Six individual scans of the area were taken using a commercial laser scanning system, and later merged using an ICP algorithm. The point cloud was reduced to an average spacing of 0.02 m and outliers removed.

B. Overview of the Algorithm

The root idea of the algorithm is to compare the geometrical configuration of the points of contact between the system and its environment with the one that could be expected from the map. At the core of the algorithm is a particle filter that samples the (x, y) position of the robot body and its heading (θ). Existing particles are propagated using an odometry model. The likelihood of each sampled (x, y, θ) triplet is then evaluated by a measurement model. This model compares the *measured* contact points with the contact points that can be expected according to the map.

The remainder of this section will properly define each part of the algorithm.

C. System Model

For further investigation, we differentiate between the world frame W and the body fixed frame B. The world frame is a local Cartesian frame using UTM coordinates. The axis of the frame are aligned such that x, y and z are pointing east, north and up (ENU). The axis of B are aligned with respect to the robot such that x, y and z are right, forward and up. Further, we define another frame Y, which has the

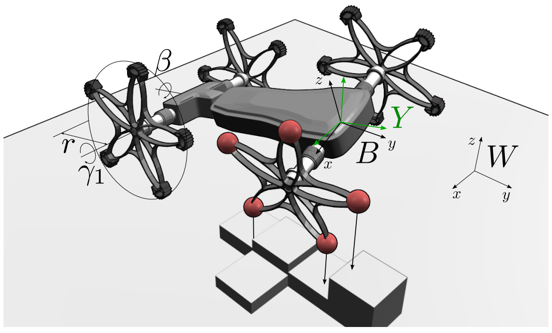


Fig. 2. The mechanical configuration of Asguard consists of four leg-wheels, with five feet each. The body center is located in the center of the front axle. The back can passively rotate around the body axis. The distances of the feet positions (red) to the ground are shown for one wheel.

same origin as the body fixed frame B, but is oriented in such a way that the xy plane of Y is always parallel to that of W (see Fig. 2). The yaw (orientation around z axis) part in the B to W (body to world) transformation T_B^W and in the Y to W transformation T_Y^W are always the same. For a decomposition of the rotation around the individual axis $R_B^W = R_z R_y R_x$, the rotation from body to Y frame can be given as $R_B^Y = R_y R_x$.

Let C be the configuration space of the robot, with $c \in C = (\gamma_1, \ldots, \gamma_4, \beta)$ consisting of the four wheel angles as well as the angular position of the body twist joint. For the wheel and foot indices $i \in I_w$ and $j \in I_f$, with $I_w = 1 \ldots 4$ and $I_f = 1 \ldots 5$, the foot positions in body frame B are provided by a simple forward kinematic function

$$p : I_w \times I_f \times C \to \mathbb{R}^3, \tag{1}$$

which is not detailed here.

D. A Priori Map

The a-priori map was generated from the point cloud of the experimental area described above.

There are various data structure that can be used to represent such environments. Three of them have been evaluated: DEM, point cloud and multi-level surface maps. The simple DEM model from [12] was used first, but showed problems with structures of multiple levels and can not handle uncertainty information. Operating on the point cloud directly yielded more accurate results, but also required the most computing power. A suitable middle ground has been found using Multi Level Surface Maps [16] (MLS). MLS have a 2D regular grid structure, were each cell in the xy-plane can hold a number of surface patches which hold the z-value and an associated uncertainty measure.

The model is such that for a given point p in world frame, and an interval $l \in \mathbb{R}$, the map function

$$m(p,l) = \begin{cases} (z,\sigma) & \text{surface with } z \in [p_z - l/2, p_z + l/2] \\ \emptyset & \text{no surface in interval} \end{cases} \tag{2}$$

provides a height and uncertainty value. If there are multiple levels in the map, the function will return the surface value which is closest to the z component of p.

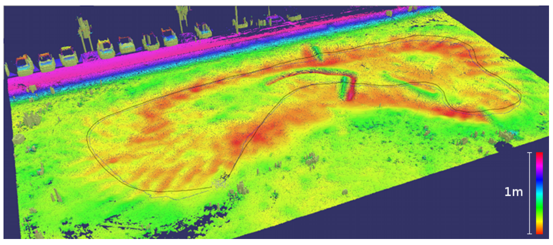


Fig. 3. The experiments were performed on a 30 m x 50 m sand field, which has local variations in height of up to 1 m. The colored areas are known horizontal surface patches, yellow represents vertical patches. The black trajectory is the GPS measurement, while blue shows the filter position for a single lap run.

E. Odometry Model

Odometry information is classically used as a rough estimation for the pose change of the robot based on the movement of the actuators [17]. Given body configurations $c_k, c_{k-1} \in C$ at time steps k and $k-1$, an estimate of the movement d in the forward direction is given by

$$d = \frac{1}{4}\sum_{i=1}^{4}(\gamma_i^k - \gamma_i^{k-1}) r_{\text{eff}}. \tag{3}$$

Since our platform has irregular wheels, an effective wheel radius r_{eff}, smaller than the actual wheel radius, is used. Skid steering vehicles usually have a large error in the estimation of rotational change. Therefore the rotational part is measured using the IMU. At each time step k the IMU provides the rotational part $q_k \in SO(3)$ of the body to world transformation. The rotation of the body frame between $k-1$ and k is trivially

$$R_{B_k}^{B_{k-1}} = q_{k-1}^{-1} q_k. \tag{4}$$

For smoothness, the robot is assumed to travel with the body y axis tangential to the body trajectory in the world frame. We define $\bar{n}$ as

$$\bar{n} = e_y \times R_{B_k}^{B_{k-1}} e_y, \tag{5}$$

where e_y is the unit vector along the y-axis (forward axis).

Assuming angles smaller than $\pi/2$, which is valid as an odometry must run at high sampling rates, the absolute value of the rotation of the robot between the two time steps is $\Delta\theta = \sin^{-1}|\bar{n}|$. This leads to the absolute values of the body translations along resp. the x and y axis of B_{k-1}.

$$\Delta x = \begin{cases} \frac{d}{\Delta\theta}(1-\cos(\Delta\theta)) & \text{if } \Delta\theta \neq 0 \\ 0 & \text{if } \Delta\theta = 0 \end{cases} \tag{6}$$

$$\Delta y = \begin{cases} \frac{d}{\Delta\theta} sin(\Delta\theta) & \text{if } \Delta\theta \neq 0 \\ d & \text{if } \Delta\theta = 0 \end{cases}. \tag{7}$$

The values for $\Delta\theta = 0$ are obtained by continuity properties. The robot body center of rotation is also not clearly defined for a skid steering system. Expressed in the frame B_{k-1}, using the translation t_{cr} from the origin to the center of the robot, the translation part of the odometry is

$$t_{B_k}^{B_{k-1}} = R_y(\cos^{-1}(\frac{e_z \cdot \bar{n}}{|\bar{n}|}))((\Delta x, \Delta y, 0)^T + R_z(\Delta\theta)t_{cr} - t_{cr}). \tag{8}$$

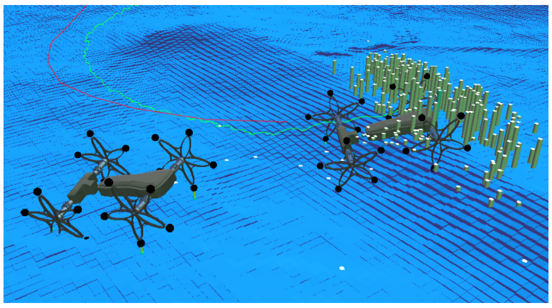


Fig. 4. Each of the grey bars is a position sample of the particle filter. The height of the bar is proportional to the particles weight. For the same orientation and body configuration reading, the particle on the left has received a lower weight compared to right, because it does not fit the terrain as well.

Where $R_y(\alpha)$ (resp. $R_z(\alpha)$) are the rotation of angle α around the y (resp. z) axis.

Finally, equations (4) and (8) together provide the odometry transformation $T_{B_k}^{B_{k-1}}$. The associated error model is a mixture model with a normal distributed part covering model uncertainties and measurement errors, and a uniform part which handles slip. The covariances of the normal part are linear functions of the travelled distance d, the rotation angle $\Delta\theta$ and the tilt angle of the robot.

F. Measurement Model

The measurement model provides the likelihood of a particle, based on a particular body configuration $c \in C$ and $\hat{q}$, which rotates the yaw fixed frame Y to the world frame W. The configuration c are essentially the encoder angles, while $\hat{q}$ is the IMU reading q with the yaw component removed.

In order to test the likelihood for a combination of c and $\hat{q}$ given a pose sample $x = (x, y, z, \theta, \sigma_z)$ and a map m, the relative foot positions are compared to the map. The method assumes that all four wheels have contact, which is a valid assumption on Asguard since the twist joint of the system is passive and the system in used in speeds where jumps are virtually nonexistent.

The forward kinematic function (1), the measurement $\hat{q}$, the configuration c and the Y to W frame transformation $T_Y^W(x)$ defined by pose particle x are used to get the position of the feet in world frame W. For each wheel $i \in I_w$ and foot $j \in I_f$

$$b_{ij} = T_Y^W(x)\hat{q}p(i,j,c). \tag{9}$$

Based on (2) the distance d_{ij} of the foot position to the map can be calculated as

$$\begin{aligned} (m_{ij}, \sigma_{ij}^{[m]}) &= m(b_{ij}, l) & (10)\\ d_{ij} &= b_{ij} - m_{ij}. & (11) \end{aligned}$$

The interval l is a parameter that was set to be three times the σ_z standard deviation of the particle. Distance and uncertainty of the distance form a pair (d_{ij}, σ_{ij}), which in effect combines the map $(\sigma_{ij}^{[m]})$ and particle uncertainties (σ_z). Note, that the map equation (2) may not be defined for all positions.

Since the robot does not have direct contact point measurement, it is needed to estimate which of the feet is in contact with the ground. If all pairs are defined for a single wheel i, the pair with the lowest d_{ij} value is selected as the contact point for that wheel. No contact point is selected otherwise. This utilises the constraint that the robot body can not be inside the ground.

The virtual sensor reading z_k and its probability estimate, given the map and a state x can now be defined as

$$z_k = \{(d_i, \sigma_i) | \text{wheel } i \text{ has contact estimate}\} \tag{12}$$

$$\hat{p}(z_k|m,x) = \prod_{(d,\sigma)\in z_k} \phi(\frac{d+\xi}{\sigma}). \tag{13}$$

Here, ϕ is the normal distribution, and ξ is a z-offset value. By deriving the quadratic terms in the exponent of ϕ, the ξ value for which $\hat{p}(z_k|m,x)$ is at a maximum can be derived as

$$\xi = \left(\sum_{(d,\sigma)\in z_k} \frac{d}{\sigma}\right)\left(\sum_{(d,\sigma)\in z_k} \frac{1}{\sigma}\right)^{-1}. \tag{14}$$

Note that $\hat{p}(z_k|m,x)$ is not yet normalised. This is because the number of valid contact points $|z_k|$ might be less than four, due to insufficient information in the map. The normalisation will be discussed in the filter section.

G. Particle Filter

The particles of the filter are encoded as 2D pose particles (x, y, θ). Additional to the x and y position, the yaw angle θ is also tracked to compensate for deviations of the IMU heading due to the presence of ferromagnetic materials in the environment, which bias the heading reading from the IMU. The estimated pitch and roll angle provided by the IMU measurement have an error which is bounded through the accelerometers. Their absolute errors are low enough (around $2°$) to be considered part of the noise.

In addition, each particle carries information about the z position of the robot and its standard deviation σ_z. These values are not sampled to reduce the size of the sampling space, but are carried along to allow handling of multiple levels. The state x of the filter at time k is therefore

$$x_k^{[n]} = (x, y, \theta, z, \sigma_z), \tag{15}$$

where N is the set of particle indexes and $n \in N$ the index of a single particle.

The measurement model presented in the previous section estimates the probability $p_k^{[n]} = p(c_k, q_k|x_k^{[n]}, m)$ of the sensor measurements (c_k, q_k) given a particle and a map.

The probability of particles that have a full set of contact points (so $|z_k| = 4$) can directly be compared to each other. However, for particles with partial or no contact information the probability given by (13) still needs to be normalized. This is handled by factoring the average influence of each contact point per z_k and taking a discounted mean as the normalization factor $\bar{p}$ as

$$A = \{\hat{p}(z_k|x_k^{[n]})^{(1/|z_k|)}|n \in N, |z_k| > 0\} \quad (16)$$

$$\bar{p} = \eta \sum_{a \in A} \frac{a}{|A|} \quad (17)$$

and as such, the weighting of the probability function is now

$$p(z_k|x_k^{[n]}, m) = \hat{p}(z_k|x_k^{[n]}, m)\bar{p}^{4-|z_k|}. \quad (18)$$

With a discounting factor η as unity, particles with no contact information would be assigned an average weighting. Usually it is helpful to assign a factor $\eta \leq 1$ in order to penalize particles on unknown terrain.

The Particle Filter used is a Sequential Importance Resampling filter (SIR) [18], [8], where an initial set of particles is generated using a random distribution over $X \sim \mathcal{N}(\mu_0, \sigma_0)$. For each step k, the odometry is used to project the particles. If the odometry has moved more than m_{trans} or rotated more than m_{rot}, the weights of the individual particles are then updated using the measurement model. This improves the performance of the filter and also decreases the dependence between consecutive measurements. Without this step, a robot standing still would converge to the most likely particle and eliminate all other possibilities. After normalisation of the weights, the effective number of particles is compared to a threshold value. If the particles are below the threshold value, a resampling step is performed, which draws a new set of samples from the given distribution. This procedure will ensure that more particles in regions were high-weight particles are concentrated, i.e. the regions where the actual robot pose is more likely to be.

IV. RESULTS

The described method was evaluated in both simulation and real world scenarios. The results in this section are all referring to real data logs obtained from the system, which were then processed off-line. The runtime of the algorithm for 250 particles was 30 s for a 180 s log file on a 2.8 GHz processor. Five different runs were recorded, with travelled distances ranging from 90 m to 140 m. Three of the runs are laps around the sand field track, with little variation in height and only sparse landscape features. Another run loops six times around a spot which has a height variation of 0.8 m, and a single small hill as a landscape feature. On the last run, the robot moves randomly across an area of the field rich in features like trenches and hills.

A. *Position Tracking*

Fig. 5 and Fig. 6 show xy position plots for one of the laps and the side loop run. The GPS reference reported an estimated error of within 5 cm during all the runs. Added that the experimental location is far from buildings (no multipath) and is under an open sky (very few occlusions), the GPS measurement can be considered very close to ground truth. The position reading for the particle filter is a weighted average (centroid) of all the positions of the particles.

We observe on these experimental runs that the centroid of the particles is able to track the position of the GPS

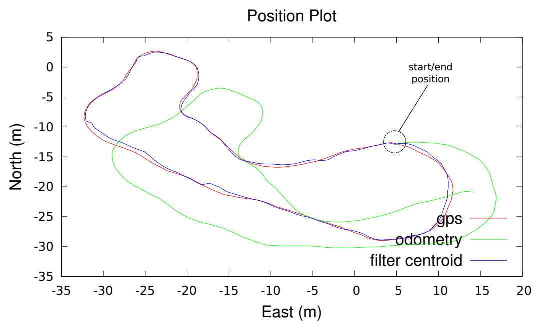


Fig. 5. Comparing xy position for the particle filter centroid, the odometry and the GPS for a run of 125 m around the sand field. Maximum deviation of the centroid compared to the GPS is 0.83 m for this run.

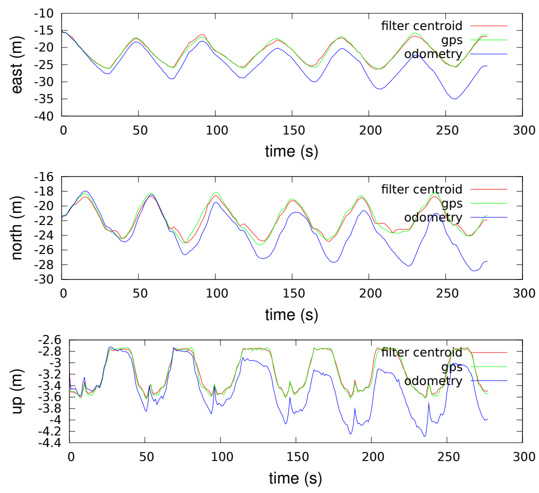


Fig. 6. This run shows multiple loops over the same area. Over time, the odometry accumulates drift due to slip and model errors, while the filter is able to track the position within fixed error bounds.

within a fixed error bound, while the odometry error grows unbounded. It can also be noted that the trajectory for the centroid is less smooth compared to that of the odometry.

B. *Metric*

Table I shows the metrics of the individual runs. The mean position error is the mean of all differences between either odometry or filter centroid and the reference position (GPS). Samples are taken each second. Final error gives the distance to the reference position at the end of the run. The filter reliably keeps the mean error within 0.5 m of the reference trajectory. Further, the total distances travelled for the runs are given. It is notable that the odometry is consistently larger than the other two, which is most likely due to slip. The numbers show that the proposed method is able to compensate for this distance error.

C. *Stability*

Once the filter was adapted to the system configuration, all the runs were evaluated using the same set of parameters.

	Mean Position Error [m]		Max Error [m]	
Run	Centroid	Odometry	Centroid	Odometry
Lap1	0.35	8.74	0.83	12.60
Lap2	0.37	9.34	1.06	12.92
Lap3	0.36	10.33	1.02	16.79
Side Loop	0.49	4.29	1.46	11.09
Cross	0.40	3.23	0.97	5.78

	Distance Travelled [m]		
Run	Centroid	Odometry	GPS
Lap1	125.83	141.97	125.19
Lap2	128.28	140.96	127.51
Lap3	124.81	135.85	123.85
Side Loop	136.84	161.63	143.89
Cross	89.67	100.31	88.46

TABLE I
COMPARING FILTER CENTROID AND ODOMETRY VS GPS DATA

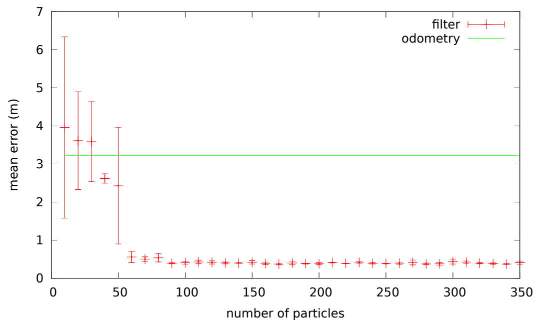


Fig. 7. Parameter variation for the number of particles used vs mean error compared to GPS. The cross run was evaluated three times for each parameter, with a different random seed.

Variations in the number of particles around this working point have been performed in order to evaluate the stability of the filter. The results on Fig. 7 show near constant performance when the number of particles is above 100.

V. CONCLUSIONS AND FUTURE WORKS

This paper outlines how embodied data and a previously known map can be used to improve the localisation of a robotic system over simple odometry. A probabilistic error model was provided for both the odometry and the measurement. A particle filter was used to track the pose of the system, and is able to handle maps with multiple levels and with uncertainty. A number of experiments were conducted on the Asguard v3 robot, to verify the approach. The results show that the filter is a significant improvement over odometry alone, and is able to keep a small, bounded, error compared to ground truth.

One limitation of the approach is the requirement of an a-priori map. This map also needs to offer a sufficient variation in map-height, as the approach will not work on a perfectly flat terrain. Future work on this subject will be looking into how to use this information in a SLAM environment, possibly together with data from optical sensors. Interesting prospects of extending the proposed method would also include measurement models for indirect embodied data. This could include average slip count or energy efficiency. Given a segmented map, this might provide the ability to distinguish between different substrates like the gravel on a pathway and the grass to the side.

VI. ACKNOWLEDGMENTS

The work presented here is part of the Project ”Intelligent Mobility”, which is funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907.

REFERENCES

[1] K. Berns, K.-D. Kuhnert, and C. Armbrust, “Off-road Robotics - An Overview,” *KI - Künstliche Intelligenz*, 2011.
[2] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, 2006.
[3] J. Schwendner and F. Kirchner, “eSLAM - Self Localisation and Mapping Using Embodied Data,” *KI - Künstliche Intelligenz*, 2010.
[4] S. Joyeux, J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig, “Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI,” *KI - Künstliche Intelligenz*, 2011.
[5] L. Hostetler, “Optimal terrain-aided navigation systems,” Sandia Labs., Albuquerque, NM (USA), Tech. Rep., 1978.
[6] W. E. Longenbaker, “Terrain-aided navigation of an unpowered tactical missile using autopilot-grade sensors,” in *Guidance and Control Conference*, 1982.
[7] N. Bergman, L. Ljung, and F. Gustafsson, “Terrain navigation using Bayesian statistics,” *IEEE Control Systems Magazine*, 1999.
[8] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, “Particle filters for positioning, navigation, and tracking,” in *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.
[9] K. Anonsen and O. Hallingstad, “Terrain Aided Underwater Navigation Using Point Mass and Particle Filters,” *IEEE/ION Position, Location, And Navigation Symposium*, 2006.
[10] R. Madhavan, “Terrain-aided localization of autonomous ground vehicles,” *Automation in Construction*, vol. 13, no. 1, 2004.
[11] R. Hoffman and E. Krotkov, “Terrain mapping for outdoor robots: robust perception for walking in the grass,” in *Proc. IEEE ICRA*, 1993.
[12] S. Chitta, P. Vernaza, R. Geykhman, and D. Lee, “Proprioceptive localization for a quadrupedal robot on known terrain,” in *Proc. IEEE ICRA*, 2007.
[13] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner, “Asguard: a hybrid legged wheel security and sar-robot using bio-inspired locomotion for rough terrain,” in *IARP/EURON Workshop on Robotics for Risky Interventions and Enviromental Surveillance*, 2008.
[14] A. S. Boxerbaum, J. Oro, G. Peterson, and R. D. Quinn, “The latest generation Whegs robot features a passive-compliant body joint,” in *Proc. IEEE IROS*, 2008.
[15] A. Martin-Alvarez, W. de Peuter, J. Hillebrand, P. Putz, A. Matthyssen, and J. de Weerd, “Walking Robots for Planetary Exploration Missions,” in *WAC’96, Second World Automation Congress*, Montpellier, France, 1996.
[16] R. Triebel, P. Pfaff, and W. Burgard, “Multi-level surface maps for outdoor terrain mapping and loop closing,” in *Proc. IEEE IROS*, 2006.
[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
[18] N. Gordon, D. Salmond, and A. Smith, “Novel approach to nonlinear/non-gaussian bayesian state estimation,” in *Radar and Signal Processing*, vol. 140, no. 2, 1993.

Simultaneous Localisation and Mapping using Terrain Classification

Jakob Schwendner, Patrick Paranhos and Christopher Gaudig

***Abstract*— Mobile robotic systems have the ability to change our everyday life. Robust localisation and mapping play a vital role in improving the autonomy of robotic systems especially in unstructured outdoor environments. While visual methods have proven very successful, they still have shortcomings in certain situations. The application of embodied data has the potential to augment and in some cases even replace exteroceptive perception. In this paper a method is proposed, which uses a particle filter for solving the SLAM problem using correspondences between visual and embodied data. The association is performed using classification based on textons for the visual part, and a novel method based on torque profiles for the non-visual part. The method is evaluated on the Asguard v3 system in a partially structured outdoor environment. The method shows promising results in providing alternative measurement models for localisation and mapping frameworks.**

I. INTRODUCTION

Mobile robotics systems have gained importance over the last years. Especially in the application of exploration, robots can provide a safe and cost effective way to fulfill mission objectives, with direct applications both in space exploration and on Earth in search-and-rescue (SAR) missions. Localization and mapping is a challenge as the robot has to build a model of the environment and know his relation to it, in contexts where **(i)** GPS is either absent (space) or very degraded (SAR).

Simultaneous Localisation and Mapping (SLAM) is a formulation of the problem of having to both build a map and localize withing the (imperfect) map that the system is building. The classical way of doing this in field robotics is to use camera systems, which correlate visual percepts from one position with those from other positions (see Fig. 1). This has brought up impressive results in a lot of environments. There are however conditions where these methods fail or do not provide the required accuracy. This includes e.g. situations like low or dynamic lighting, lack of features, repetitive patterns and so on. Moreover, vision-based systems are highly CPU intensive. A possible way to augment this has been proposed in [1], by using so called embodied information: sensor data which originates from within the body of the robot. [2], [3] have shown that information on the structure of the environment can already be used to localise within a known environment using a particle filter. Similar approaches have been described in terrain based localisation for airborne systems (e.g. [4]).

In this paper, we investigate how terrains with little physical structure but different terrain types can also be used to provide additional information for localisation and mapping, by finding correspondences in the data for terrain types that both look different and "feel" different. Unlike classical vision based methods, which correspond visual with visual information, this method corresponds visual data with proprioceptive data. This has been used by [5] in order to learn the classification of terrain properties and in [6] in order to learn the classification of laser scan based terrain properties based on vibration data of the robot. Hoffman and Krotkov [7] have additionally shown some early examples of using these principles for SLAM, using it for compensating the height error of a map on their Ambler robot.

German Research Center for Artificial Intelligence (DFKI), Robotics Innovation Center (RIC), Bremen, Germany {firstname.lastname}@dfki.de

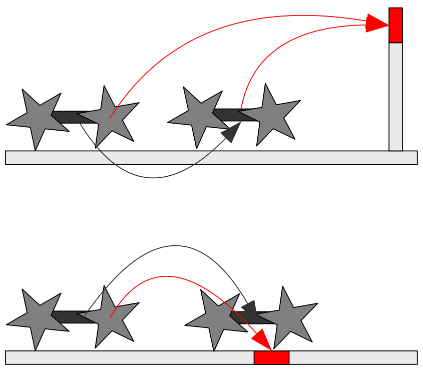


Fig. 1. Classical SLAM approaches use visual to visual correspondences (top) to estimate pose changes between two time steps. In this paper we employ visual to embodied correspondence (bottom), which can provide additional information which is very relevant to the system's navigation loop.

This work contributes to the state of the art in two aspects. (1) The application of visual and non-visual correspondences in a full SLAM scenario. (2) Classification of terrain based on wheel slip, for a leg-wheel hybrid system.

The paper is organized as follows. First, an overview of the developed method is provided, and in particular of the SLAM process. Then, both exteroceptive and proprioceptive methods of terrain classification are described. Results are given for the classification methods used and the overall process is evaluated on the Asguard v3 system in an outdoor scenario. Finally, we conclude on this work and propose some future directions to extend this SLAM method.

II. METHOD

A. eSLAM

For the problem of simultaneous localisation and mapping, we employ a method based on the FastSLAM idea.

FastSLAM [8] is a probabilistic filtering method, which uses weighted particles to represent the probability distribution of an association between a map and the current localisation in that particular map. The particle filter is said to be Rao-Blackwellized, as errors in pose are assumed to be the main contributors to map errors, and thus maps are considered conditionally independent. Each particle therefore carries its own map in addition to the pose information.

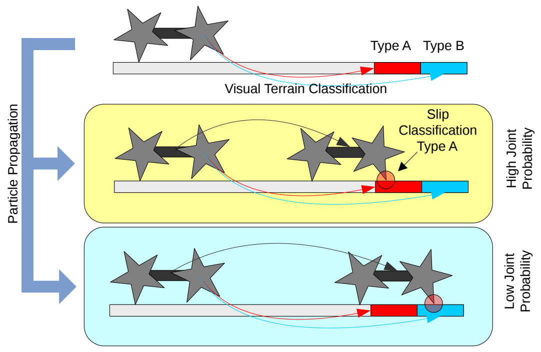


Fig. 2. The general idea is that at any particular time step the terrain is classified using visual methods (top). Any wheel slips are used to classify the terrain directly underneath the robot. The joint probability of visual and slip based classification can lead to high (center) and low (bottom) particle weights in the filter.

The problem domain that we investigate inherently keeps the system of interest in constant contact with the environment. This provides the possibility of reducing the need to represent all six dimensions of the probability distribution of the 3D rigid body state (position and rotation) of the system as particles. Also, using an inertial measurement unit (IMU), the gravity vector can be estimated well enough. The state of our system is thus sampled in position on the plane orthogonal to the gravity vector and rotation around the gravity vector (x, y, θ). Each particle then carries its weight as well as an absolute notion of mean height over the plane and a normal probability distribution (z, σ_z). Following the FastSLAM idea, each particle also carries its own map m. The map representation used is the Multi-Level Surface maps (MLS) [9], where each cell can have multiple height entries and also differentiates between horizontal and vertical cell entries. Each of these cell entries also has an encoded terrain information as a tuple $t = (t_n)$ where t_n represents the probability that this cell entry is of terrain type n. Updates of the cells are performed according to the original paper from Triebel et. al. [9]. The additional terrain type information that is associated with each cell is updated using the mean between the existing cell's value and the update from the measurement.

Fig. 2 gives a high level overview of the general idea that is employed in this work. Current particles are projected using a simple 3D odometry based on the IMU and the wheel encoder readings [2]. The stereo camera system generates a distance map, which is converted to a point cloud. This point cloud is augmented with information from the visual terrain classification (see next section). For each of the particles, this point cloud is used to update the map of the respective particle (map update step). Finally, the particle weights are updated in the *measurement step*, using two different measurement methods that are compared and combined in this paper.

The first measurement model is the method used in [2], which uses an environment contact model in order to estimate the likelihood of a measurement given the current particle state. Basically, this is how well the body pose provided by the particle in addition to the IMU fits the shape of the terrain.

The second measurement model, which we propose in this paper, uses information from embodied terrain classification based on wheel slip detection (next section). This information, like the visual information, also provides a tuple where each dimension corresponds to the probability of belonging to the specific terrain class z_k. The weight of the particle is updated according to the probability function

$$p(z_k|t_m) = \begin{cases} p_{\text{same}} & \max z_k = \max t_m \\ p_{\text{different}} & \max z_k \neq \max t_m \end{cases}$$

where t_m is the tuple from the map, and the $\max$ function provides the index of the element of the tuple with the highest value. The two probabilities for same classification and different classification are set as parameters. The two measurement steps described are considered independent, and can be used individually, jointly or even together with a measurement step in a visual-visual correspondence.

Projection and measurement steps are not called in a fixed time period, but are triggered by distance travelled by the robot according to the odometry. This is to improve the independence between noise of two different measurements. As described in [2], a minimum effective particle threshold is used to trigger importance resampling of the particles according to their weight.

B. Terrain Classification

The methods for terrain recognition can be divided into two classes: exteroceptive and proprioceptive. Exteroceptive are those whose measurements depend on stimuli originating from outside the body, such as cameras and laser scanners based methods. Proprioceptive are those whose measurements depend on measuring the internal state of the robot, such as vibration and traction based methods. The information used by these two classes are complementary, since exteroceptive methods extract the information based on how a terrain “looks”, while proprioceptive methods on how a terrain “feels”. [5] combined both classes in a self-supervised terrain classification framework, where what was “seen” by the robot is combined with what was “felt”, allowing the system to learn to predict the terrain properties. In this work, we use a texton based approach to perform visual classification of terrain types, and a novel torque profile based method for terrain classification at wheel slip events.

1) Exteroceptive Classifier: There are several approaches to vision-based terrain classification. Both [10] and [11] present methods where several low-level classifiers including an RGB-based color classifier and one based on textons (texture patches) are used and later fused to obtain a result better than those produced by the individual classifiers.

An approach that does not rely on color images but works on gray-scale data is presented in [12]. Here the *SURF* and *Daisy* descriptors are used for classification.

In [13], local binary patterns (LBP) are used for texture classification.

In order to provide an exteroceptive look-ahead terrain classification, a visual approach using the stereo camera system on the Asguard robot has been implemented. Terrain class regions are identified in one camera image and are later mapped onto the depth data obtained by a dense-stereo algorithm.

The classifier is based on histogram back-projection and uses a set of reference images for each of the predetermined terrain classes. For the experiments presented here, four images per terrain class were selected. The implementation uses the *OpenCV* computer vision library. All reference images are converted from RGB to the HSV color space. This representation has the advantage of being somewhat more robust against changes in illumination. Then, for each image, the HSV histogram is computed and stored. During the classification run, the live camera image (640x480 pixels) is scaled down to 320x240 pixels to allow for faster processing and then converted to HSV.

For each pixel a histogram back-projection is performed: the corresponding bin in the HSV histogram is determined, but instead of incrementing the pixel count for this bin, a lookup in all of the reference histograms is performed. The resulting pixel counts can be viewed as non-normalized probabilities with respect to the empirical probability distributions represented by the reference histograms. See [14] for the basic technique of histogram-based image indexing.

For each of the previously stored histograms, an output image containing the pixel counts is created. To this intermediate result a simple threshold and the morphological operation of *opening* (erode, then dilate) are applied. The outcome is an image containing *blobs* representing the terrain class of the corresponding reference histogram. The size of each region is computed and small blobs are discarded. As a last step the resulting images for each terrain class (corresponding to the number of reference histograms) are merged using a logical *OR* operator. Figure 3 shows an example of the input data and the resulting classifier output.

2) Proprioceptive Classifier: A robot internal state is affected by the robot interaction with the environment. Thus, through proprioceptive sensing, it is possible to perceive and differentiate between environments. In the work of [15], for example, a legged robot was able to correctly classify 94% of the terrain shapes by measuring motor currents and ground contact forces while doing a pre-defined motion on terrain samples. [16] estimates terrain parameters by relating the measured physical quantities with the equations that describe smooth rigid wheel traveling through deformable terrain. Finally, [17] presents a comparison between vibration based techniques for terrain classification, where the support vector machine method proposed obtained the best result.

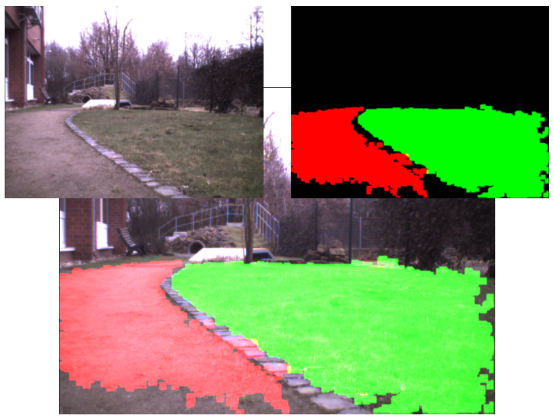


Fig. 3. The camera-based terrain classifier can estimate what terrain the robot will encounter next. Upper left: Input image. Upper right: Classifier output, terrain classes *grass* (green) and *sand path* (red). Bottom: Overlay visualization of terrain classes.

The deflection in the motor coupling has a direct relation to the torque applied at the wheel-leg. So by measuring the deflection in the coupling, the wheel-leg-ground contact points and the body attitude the traction force can be estimated through the robot kinematic equations.

The true translation, $\Delta^t(N)$, of a 4 wheel differential drive vehicle is given by the set of eqs. (1). The encoders measure the apparent translation in each wheel, $\Delta^n(N)$, and an IMU the heading changes, $\Delta\gamma$. To solve the system 4 hypotheses are considered, in each hypothesis only one wheel is assumed as not having slipped. This generates 4 solutions for the system, where in each solution the amount of slip on the other 3 wheels is estimated. If this value is above a threshold, then the wheel is considered as having slipped under that hypothesis. So, if 3 hypotheses reach the same conclusion of which wheels slipped and which did not, then the slip is detected.

$$\begin{aligned} \Delta^t(N) = &\quad \Delta^m(N) + slip(N), N = 1..4 \\ \Delta^t_{left} = &\quad \Delta^t(1) = \Delta^t(2) \\ \Delta^t_{right} = &\quad \Delta^t(3) = \Delta^t(4) \\ \Delta\gamma = &\quad (\Delta^t_{left} - \Delta^t_{right})/axis \end{aligned} \tag{1}$$

The basic idea of the proprioceptive classifier in this paper is to recognize if different terrains have sufficiency distinguishable friction coefficients, the traction force patterns during slip (i.e. when the traction force crosses this threshold) will also be distinguishable. In order to use this idea in practice, a support vector machine (SVM) is trained, using the libsvm library, to recognize the traction force patterns that characterize different terrain. Each wheel-leg has a SVM classifier to recognize the specific traction patterns during slip of that wheel-leg. The SVM is trained off-line with hand-labeled data sets. The feature space is composed of the concatenation of 3 normalized histograms: The histogram of the traction forces, of the body linear velocities and of the body angular velocities. Only the data measured during the N last wheel-leg "steps", where a slip is detected, are considered valid for training / classification 4. N is the size of the sliding window and a "step" is defined as a 72 degree (due to the Asguard wheel geometry).

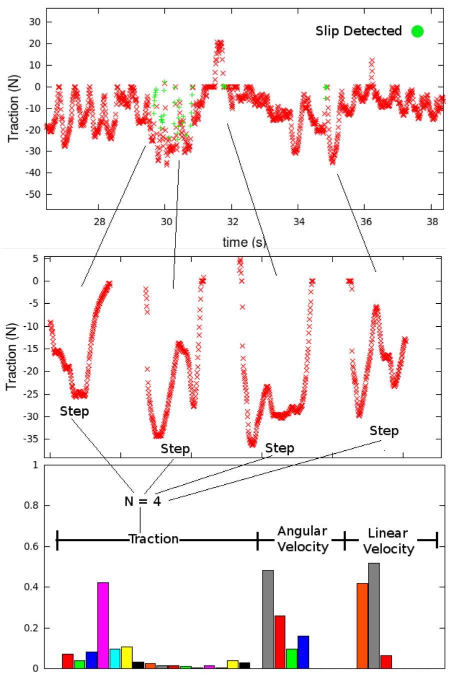


Fig. 4. Feature Space formed of 4 steps (N=4) where slip was detected.

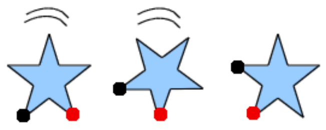


Fig. 5. Wheel-Leg Step

III. EXPERIMENTAL RESULTS

A. *Experimental Setup*

The experiments for validating the proposed method have been performed on the Asguard v3 robot [18] (see Fig. 6). This is a leg/wheel hybrid [19], [20] system with very good terrain negotiation capabilities. The Asguard v3 uses an inertial measurement unit, a laser scanner with 30 m range, a stereo camera system and wheel and body joint encoders. The sensor data is recorded and properly timestamped using the Rock [1] framework. Dense stereo processing is done using libelas. The logs have been recorded on the DFKI Test-track (see Fig. 7) and are post processed for the results shown in the results section. The Asguard systems also uses an RTK GPS, which is in principle able to get positioning data with up to 2 cm accuracy. This is used as a reference for comparing the results.

[1]www.rock-robotics.org

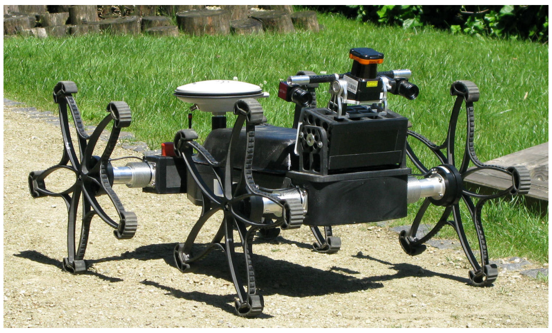


Fig. 6. The asguard v3 system was used to perform the experiments shown. This leg/wheel hybrid as a weight of around 13 kg. The experiments were conducted at a speed of around 0.5 m/s.

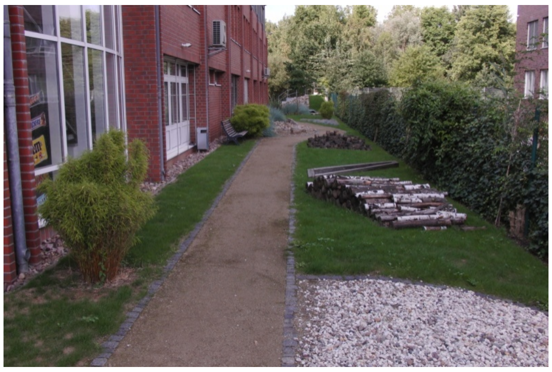


Fig. 7. The DFKI Test Track has a size of around 8 m by 70 m and features multiple structured obstacles. It also has a variety of ground substrates like gravel path, grass or pebbles.

B. *Proprioceptive terrain classifier*

In order to train the proprioceptive classifier, the Asguard system drove autonomously along a pre-defined trajectory of 24 m on a bare ground path (center path on Fig. 7) and the same trajectory on grass. The trajectory was repeated 6 times for the earth path and 10 times for the grass path. The slip detection algorithm detected per wheel in average a "step" with slip every 1.23 m for the gravel and 1.74 m for the grass. The data gathered were hand-labeled and separated into a training set and a validation set. The metric for qualifying the SVM results is the balance accuracy (BA), defined as the arithmetic mean of true positive rate (TPR) and true negative rate (TNR)

Three different SVM configurations are trained and validated on the data gathered, the results are represented in I, the different configurations are depicted as follow:
A - the SVM in the configuration as detailed in the section above.

B - a SVM with the feature space composed only of the normalized histogram of the traction forces.
C - the feature space is the same as in the configuration A, but all "steps" are used, instead of only the "steps" with slip.

TABLE I
THE BALANCE ACCURACY OF THE 4 WHEEL-LEGS

	A					B	C
	N=1	N=2	N=3	N=4	N=5	N=3	N=3
Back Left	77%	79%	80%	85%	87%	70%	68%
Back Right	73%	78%	87%	87%	85%	80%	68%
Front Right	67%	76%	83%	85%	92%	82%	70%
Front Left	68%	71%	77%	85%	88%	72%	65%

The trained SVMs where used to generate terrain classifications data on a single run on the test track. This slip based classification was then provided to the filter, which matches the classification data with the results from the visual classification as described in the previous section.

C. SLAM

Fig. 8 shows a graphical representation of the internal state of the SLAM filter.

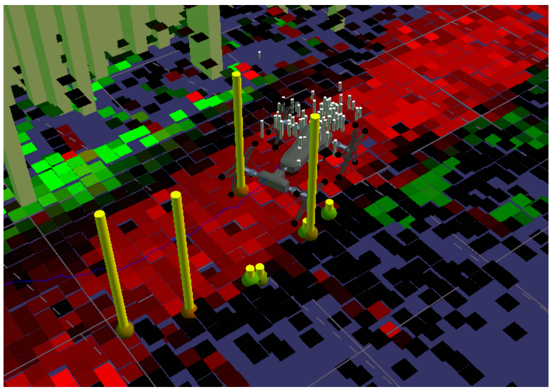


Fig. 8. Visualisation of internal filter state for the test track run. The particles are shown as grey bars with their height proportional to the particle's weight. The yellow bars represent points where slip based classification events occurred, and how well (bar height) they matched with the classification from the visual data.

The gathered log has a length of 125 s and spans a distance of around 70 m. Fig. 9 shows the resulting map, which matches reasonably well with the original. The visual classifier was able to classify the map on relevant parts along the trajectory (see Fig. 10) of the robot.

Looking at the errors in Fig. 11, we can see that the method described was able to improve over odometry for the test track run in most cases. In the beginning of the log, odometry is still better, but accumulates error over time. This is also the case for the filtered positions. However, the error seems to be reduced by the error models employed. The strongest improvement was visible when using a combination of the shape based and the slip based update methods.

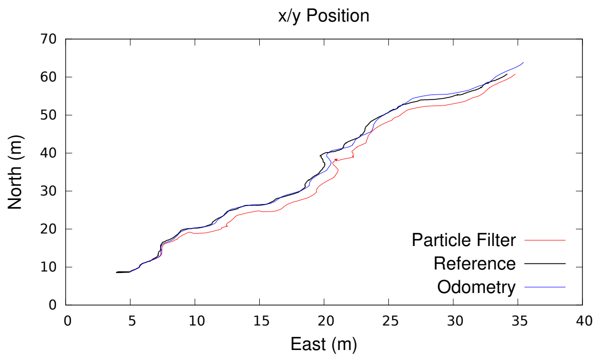


Fig. 10. Odometry and filter centroid (shape and slip combined) for the test track run. The odometry tracks the true position very well in the beginning of the run, but deviates during the end along the direction of travel, presumably due to wheel slip.

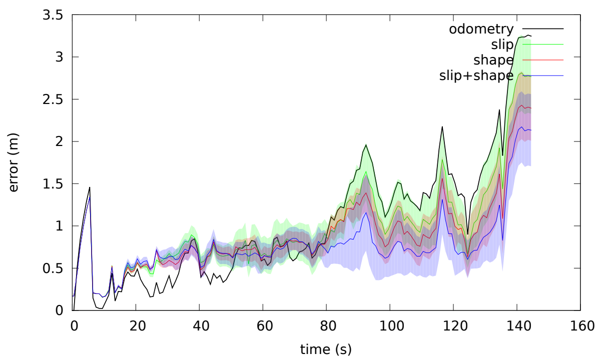


Fig. 11. Errors compared to reference trajectory, for test track run. For each of the different measurement models, the filter was run 5 times using different random seeds. The mean and error bars are plotted over time. Both error models improve over odometry and even more so when used together.

IV. CONCLUSIONS

In this paper, an alternative information source for localisation and mapping was presented. Based on visual and proprioceptive terrain classification, a method was introduced to perform visual to embodied data association. A novel approach based on SVMs was used to classify slip events for the underlying terrain. The method has shown to differentiate between two classes of terrain with up to 92% balance accuracy. The method has low latency, only 1/5 of rotation of the wheel-leg is needed for a classification, and each wheel-leg is an independent classifier. The experiments were conducted with autonomous driving, thus the classifier works independent of any human influence and of probing movements. The recognition of the traction force pattern presents better results when the robot dynamic is considered in the feature space vector and when slip is detected, respectively increase in average balance accuracy from 76.00% and 67.75% to 81.75%. Further, a novel method was presented on how to combine visual with embodied data to perform simultaneous localisation and mapping tasks. The information

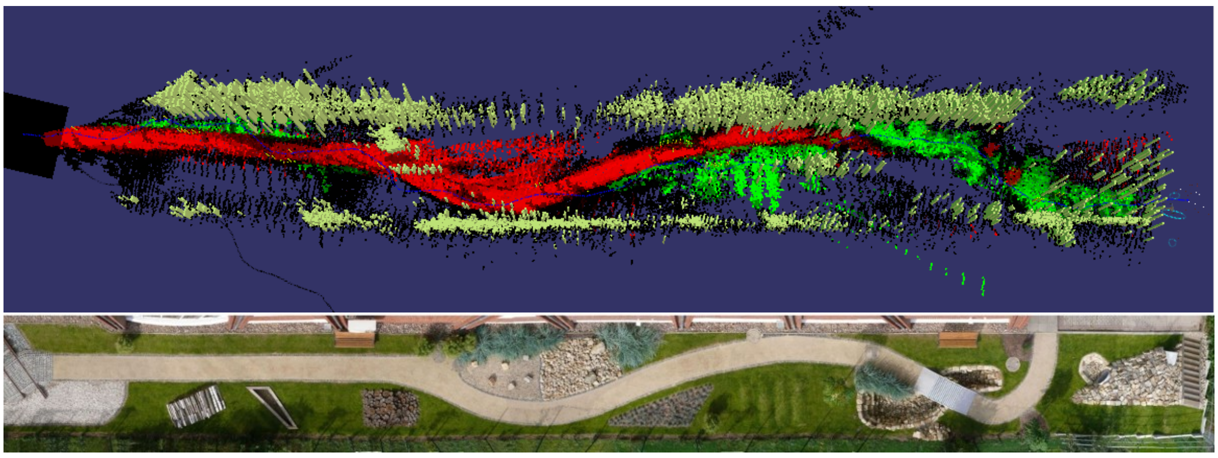


Fig. 9. Resulting map for the test track run (top). Red and green parts are identified terrain types, while yellow indicates walls. Parts that could not be visually classified are black. (bottom) Top view of test track for reference.

from the slip based detection showed an improvement over the odometry, when compared to a reference trajectory. Since this is a different update method, the improvement should be independent of visual-visual correspondences, which could therefore be used in combination. The presented work serves as a proof of concept and should be further investigated. Especially how this type of information influences the position and map errors, and what error rates on the measurements are acceptable are subject to further research, as well as the measurement and error models employed in this work.

V. ACKNOWLEDGMENTS

The work presented here is part of the Project "Intelligent Mobility", which is funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907.

REFERENCES

[1] J. Schwendner and F. Kirchner, "eSLAMSelf Localisation and Mapping Using Embodied Data," *KI - Künstliche Intelligenz*, vol. 24, June 2010.

[2] J. Schwendner and S. Joyeux, "Self Localisation using Embodied Data for a Hybrid Leg-Wheel Robot," in *IEEE International Conference on Robotics and Biomimetics*, 2011.

[3] S. Chitta, P. Vernaza, R. Geykhman, and D. Lee, "Proprioceptive localization for a quadrupedal robot on known terrain," in *The IEEE International Conference on Robotics and Automation*, 2007.

[4] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," in *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002, pp. 425–437.

[5] C. A. Brooks and K. Iagnemma, "Self-Supervised Terrain Classification for Planetary Surface Exploration Rovers," *Journal of Field Robotics*, pp. 1–24, 2012.

[6] K. M. Wurm, R. Kummerle, C. Stachniss, and W. Burgard, "Improving robot navigation in structured outdoor environments by identifying vegetation from laser data," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1217–1222, Oct. 2009.

[7] R. Hoffman and E. Krotkov, "Terrain mapping for outdoor robots: robust perception for walking in the grass," in *IEEE International Conference on Robotics and Automation*, 1993, pp. 529–529.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[9] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ International*, 2006.

[10] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Fast terrain classification using variable-length representation for autonomous navigation," *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[11] I. Halatci, C. A. Brooks, and K. Iagnemma, "A study of visual and tactile terrain classification and classifier fusion for planetary exploration rovers," *Robotica*, vol. 26, no. 6, pp. 767–779, 2008.

[12] Y. N. Khan, P. Komma, and A. Zell, "High resolution visual terrain classification for outdoor robots." in *ICCV Workshops*. IEEE, 2011, pp. 1014–1021.

[13] T. Ojala, M. Pietikinen, and T. Menp, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[14] M. Swain and D. Ballard, "Indexing via color histograms," in *Proc. Third International Conference on Computer Vision*, 1990, pp. 390–393.

[15] M. a. Hoepflinger, C. D. Remy, M. Hutter, L. Spinello, and R. Siegwart, "Haptic terrain classification for legged robots," in *IEEE International Conference on Robotics and Automation*. Ieee, May 2010, pp. 2828–2833.

[16] K. Iagnemma, S. Kang, H. Shibly, and S. Dubowsky, "Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers," *Robotics*, vol. 20, no. 5, 2004.

[17] C. Weiss, N. Fechner, M. Stark, and A. Zell, "Comparison of different approaches to vibration-based terrain classification," in *European Conference on Mobile Robots (ECMR)*, 2007.

[18] S. Joyeux, J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig, "Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI," *KI - Künstliche Intelligenz*, Feb. 2011.

[19] U. Saranli, M. Buehler, and D. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[20] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner, "Asguard: a hybrid legged wheel security and sar-robot using bio-inspired locomotion for rough terrain," in *IARP/EURON Workshop on Robotics for Risky Interventions and Enviromental Surveillance*, 2008.

Map Segmentation based SLAM using Embodied Data

Jakob Schwendner

Abstract— **Autonomous mobile robots offer the prospect of extending our knowledge of remote places in the solar system or in the ocean. They also have the potential to improve everyday life with ever increasing adaptability to a large variety of environments. One of the key technological elements is the ability to navigate unknown and uncooperative environments. A range of solutions for the simultaneous localisation and mapping (SLAM) problem have emerged in the last decade. One factor which is often neglected is the fact that the robot has a body which interacts with the environment. In this paper a method is presented, which utilises this information and uses visual and non-visual correlations to generate accurate local map segments. Further, a method is presented to combine particle filter based local map segments and constraint graph based global pose optimization to a single coherent map representation. The method is evaluated on a Leg/Wheel hybrid mobile robot and the resulting maps compared against high resolution environment models generated with a commercial laser scanner.**

I. INTRODUCTION

Mobile robots have a large potential to improve our everyday life. By not being limited to a single place, single or multi-robot systems can provide flexible services in situations that would be dangerous, uncomfortable or not economically viable for human presence. They are becoming more and more suited to perform tasks autonomously, that before were only possible to do for a human, or at least with human intervention. One important factor is for the mobile robot to be able to navigate its environment, so to get from A to B. From an engineering point of view, how difficult this is largely depends on the environment and the resources and abilities of the robot. Navigation usually requires a knowledge of the surrounding (*mapping*), a knowledge of the position of the robot within the surrounding (*localisation*) and means to convert this information into actuator commands to reach goals within it (*path/motion planning*). While path/motion planning also has an influence on both mapping and localisation by posing requirements with regards to the accuracy of both, the focus in this paper is on the localisation and mapping part.

In structured and well known environments many of the problems of navigation already have well working solutions. Robots that are being operated in buildings have the advantage, that the map of the building is most likely static and can be acquired a-priori. Sometimes it is possible (but often not desirable) to adapt the environment in order to aid the robot navigation (e.g. RFID tags [1]). Further, indoor environments usually suffice to be modelled in two dimensions (single floor [2]) or a collection of 2D models with interconnections (multi-floor with stairs [3]). Unstructured outdoor environments often require a more complex modelling because fewer assumptions can be made, and are generally more difficult to sense because of varying environmental conditions like lighting etc. For this reason, it is often desirable to perceive the environment using different sensors [4] and information processing chains in order to be more robust to such changes in perception quality.

German Research Center for Artificial Intelligence (DFKI), Robotics Innovation Center (RIC), Bremen, Germany
jakob.schwendner@dfki.de

Given a more complex environment, chances are, that the interaction of the mobile robot with the environment will become richer. A robot on a plane does not receive a lot of feedback from the interaction with its environment. When introducing physical walls, collisions with one of them can already provide crucial information. More unstructured terrain has a varying slope with respect to gravity, or may posses certain soil properties that affect the slip of the system. Such information can be a rich source of clues when they vary with localisation and even more so when they can be perceived with multiple modalities. This class of information, which is connected to the physical entity of the robot has been termed embodied data in the previous publications of the Author [5]. Unlike exteroceptive data it is only available and depends on the type of the physical presence of the robot (hence the term *embodied*). In [6] it was shown that such information can be used to localise without the help of exteroceptive data in an area of 40 x 30 m, and a distance travelled of over 100 m, to within a 1 m accuracy. In this paper it is presented that correlating this embodied data with exteroceptive data, it is also possible to perform simultaneous localisation and mapping tasks (SLAM).

The SLAM problem has received widespread interest and while many great solutions have already been proposed it is still of great interest to the robotic community [7], [8]. One of the most successful branches for handling the SLAM problem is the application of Bayesian statistics. The pose (position and orientation) of the robot and maps are modelled as probability distributions, which get updated by changes in the robot pose and updates to the environment using sensor information. The art is to formulate a state space description which can model the problem well enough and is scalable to the size of the problem. One common trick is to divide the problem into more tractable parts, and without loosing too much information. FastSLAM [9] uses a method called Rao-Blackwellization to factor the probability distribution. It uses a particle filter, for which each particle holds its own map. For large maps however and when loop closing (which is performed implicitely) this approach suffers from what is known as the particle depletion problem. Multiple good solutions within the realm of the particle filter have been proposed (e.g. [10]) to mitigate particle depletion. A different approach to the SLAM problem, which performs particularly well at loop closing is to partition the map into individual sub-maps, which are then connected by position constraints with uncertainty. These constraints can form a pose graph which can be optimized using some form of global relaxation [11], [12]. Graph based SLAM solutions are usually scalable and perform particularly well at loop closing. Another way to improve the particle depletion problem is called SegSLAM [13]. The principle idea here is to segment the global map into local sub-maps, but still use a particle filter. Instead of discarding old map parts of particles which have been eliminated in a resampling process, SegSLAM keeps these parts to perform recombination of local sub-maps. In this way the effective number of particles can be drastically improved while reducing the computational effort.

In this paper, a solution is proposed, which combines some of the ideas of FastSLAM, GraphSLAM and SegSLAM mentioned in the previous paragraph in order to perform SLAM in an unstructured environment which is modelled in 3d. The two major contributions are (1) the generation of locally consistent maps using correlation of visual and embodied data. And (2) combination of FastSLAM with GraphSLAM based on interpretation of particle distribution as Gaussians.

The next sections describe the proposed idea in more detail, and set it into context with existing methods. The approach is verified in an experiment on the Asguard v3 system, navigating a (partially) unstructured environment. Based on the experimental results, and the discussions, a conclusion is given and thoughts are provided on how to improve on the proposed method.

II. METHOD

A. *overview*

The principle idea proposed in this paper is to divide the 3D SLAM problem into multiple subparts and make a few assumptions.

Firstly it is assumed that the method is aplied on a vehicle that is in contact with a solid surface (e.g. Rover or similar). The second assumption is that there is a gravity vector, which can be estimated up to sufficient accuracy (around 1 degree) using inertial sensors. The inertial measurement can estimate the heading of the rover, but will do so with an unbounded drift error. Earth magnetic field corrections are not explicitly considered. Partly because they can be misleading in the vicinity of ferromagnetic structures or may not be present (lunar exploration) at all.

Local sub-maps are generated using a FastSLAM based particle filter, which samples in the x and y position and heading. Each particle contains a map and an estimator for the z position of the robot. The robot position is propagated using an odometry model. The map is updated using a probabilistic sensor model. The map itself is represented as a multilevel surface map [14], which stores patches as mean and variance of height in a grid structure. The position of the robot in z is updated using an environment contact point model (based on [6]), which also provides relative weighting for the particles.

While SegSLAM uses heuristics to decide when to segment a map, a simple threshold mechanism is used in this approach. Further, all particles of the filter are segmented at the same time. The result of this segmentation is a structure with an origin, at which all the particles start and a collection of trajectories and maps, one for each particle. The final poses of the particles in the segment are used to estimate the parameters of a Gaussian distribution.

These segments are now used as nodes in a classical GraphSLAM setup, where consecutive segments are given constraints based on the Gaussian distribution from the final pose of the particles. This structure in itself does not provide any additional value, it allows however for explicit loop closing to be performed. When a loop is closed, a pose graph optimizer is used to relax all the constraints. This results in an updated final pose for each of the segments. For each segment, the closest actual pose is then used as the candidate for the current best guess at the final map.

B. *Local Maps*

The full robot state $\check{x}_k = [x\, y\, z\, \theta\, \phi\, \psi\, m]$ is the position and orientation in three dimensions, plus the map m of the environment. Representing an uncertainty distribution over the state as is required by all Bayesian filters is non trivial because of the relatively large number of dimensions. Because of the assumption that we can measure the orientation against the gravity vector, the two angles pitch and roll given by ϕ and ψ do not need to be represented, which leaves us with a state representation of

$$x_k = [x\, y\, z\, \theta\, m] \tag{1}$$

for which we want to find the full slam posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$, where z are the measurements and u are the control inputs to the system for the timesteps 1 to t. We now use a Rao-Blackwellized particle filter [15] to represent this posterior. Because of the nature of the problem, the maps posterior largely depends on the robot pose, as the pose error is the largest contributor to the map error. Further, because of the assumption that there is a surface on which the robot is in contact with, the part of the position parallel to the gravity vector (z) is closely connected to the map. For this reason, the full slam posterior is factored into the part which is represented by a particle distribution

$$x_k = [x\, y\, \theta] \tag{2}$$

and the map including z-position $m_n = [z m]$, which for the z position is represented as a Gaussian and for the map as a regular grid structure in x and y, where each cell can have multiple surface patches which are also represented as Gaussians (see [14]). The full posterior is now estimated as

$$p(x_{1:t}|z_{1:t}, u_{1:t}) \prod_{n=1}^{N} p(m_n|x_{1:t}, z_{1:t}). \tag{3}$$

Each of the N particle has a map and a z position estimator.

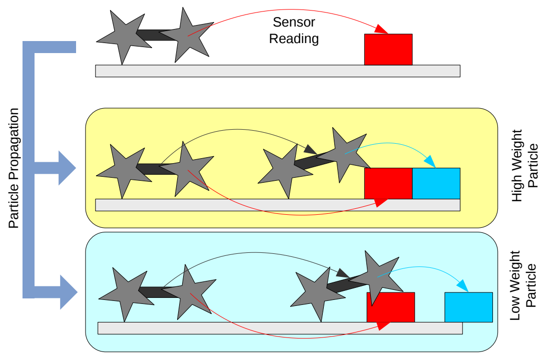


Fig. 1. The particle filter used in this paper samples over x, y and heading. Each particle has the map and z position factored. Sensor readings are added to the map according to the particle position (top). The particles are propagated according to the odometry function. A contact model determines how well the contact points of the robot match the map, and weigh the particles accordingly (center, bottom).

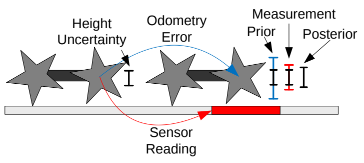


Fig. 2. Uncertainty in z position is represented by a normal distribution per particle. Map cells also have an uncertainty associated, which is updated based on the uncertainty of the robot and a measurement uncertainty (red). Errors in the odometry model (blue) can in this way be corrected using the assumption that the robot is in contact with the environment. Only the difference between measurement and odometry error can be used for correction, since the other part of the error is dependent.

Estimating the map and the z position is a subproblem which can be considered independently. New sensor readings are added to the map based on the current position and orientation of the robot. While x and y part of the position are part of the particle filter, and are marginalized for the map, the z position is not. Since both map and z-position are represented as Gaussians, a Kalman filter could be used to estimate the errors between the map and the z position. Performing this for the full slam problem, including all poses and map cells is computationally involved. For performance reasons, the z position and map heights are estimated in single variate Kalman filters instead. The initial estimate for for z and the variance of z, σ_z^2 is zero. The odometry model provides a relative z difference and a variance for it, which are added to z and σ_z^2. New measurements are added to the map cell with $(z+z_m, \sigma_z^2+\sigma_m^2)$, where z_m is the measurement (which is always relative to the robots position) and the measurement uncertainty σ_m^2. Updates to the z position are handled using the assumption that the robot is in contact with the environment. A contact model $(z_G, \sigma_G^2) = G(c,m)$ provides the z position and variance according to the contact with the environment [6], based on the map m and the set of environment contact points c on the body of the robot. Here a rough approximation of the full covariance is used. Applying (z_G, σ_G^2) directly to the state estimate of z neglects that large parts of the error have a common origin (see Fig. 2), only the independent error that has been accumulated between when the map was written and the current pose can be corrected. So instead of applying the Kalman update steps to (z, σ_z^2), it is applied to $(z, \sigma_z^2 - \sigma_b^2)$, where σ_b^2 is the average variance of the map cells, which were used in the contact model calculation.

The Particle Filter used is a Sequential Importance Resampling (SIR) [16] filter. All particles are initialized to zero mean and covariance at the beginning of each sub-map. Particles are weighted according to the measurement probability function given in [6] (see Fig. 1). In this way a correspondence is created between visual information, which is stored in the map and non-visual information, which is used to both update the z-position and estimate the weight of the particles. Robot poses, which match the terrain well receive heigher weights compared to poses which do not match well with the terrain. When the effective number of particles is below a threshold, the particles are resampled according to their weight. Pose updates are performed based on the odometry model, while measurements are done if the distance to the previous measurement is above a certain threshold. This improves conditional independence between measurements.

C. Pose Graph

For larger travel distances the position error will grow unbounded, since there are no absolute reference measurements. Once previously seen places are revisited, it is possible to reduce the position error with respect to the original position. This is called loop closing, and has seen extensive research already (e.g. [10], [9]). Particle filters can implicitely close loops by reinforcing particles with higher global consistency [9]. In practice this is a problem for larger runs, since the number of particles is limited by CPU and memory. When the uncertainty grows, the particle density is reduced, so that the chance that one of those particles represents the correct loop shrinks with growing distance. This problem is referred to as particle depletion [15].

In this approach the sub-maps, which are called segments, are generated using the approach described in Sec. II-B. Each segment starts at a position and orientation uncertainty of zero, as everything within the segment is represented relative to its origin. After one of the particles passes a certain threshold in either x or y position relative to the segments origin, the current state of the particle filter is frozen in a segment representation $S = (T, \Sigma, p_{1:N})$, where T is the mean pose transform from the final state of the robot to the origin of the segment, and Σ its covariance. For each of the particles a state $p_n = (z_{1:t}, m)$ is recorded, which holds the z position of the robot for each timestep in the segment, as well as the final map. T and Σ are extracted from the final particle poses and their relative weighting.

The individual segments are organised in a graph structure G. Segments are represented as vertices, and constraints between segments are the edges of the graph. Let $x =$

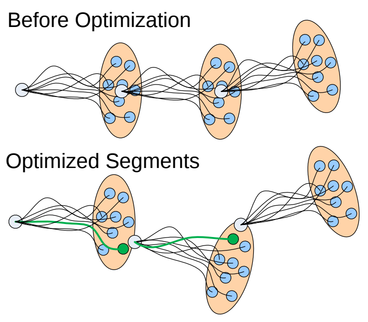


Fig. 3. Individual map segments consists of the particles and their associated trajectories and maps. Constraints between segments are modelled as the normal distribution over the weighted particle poses. After including other constraints e.g. from loop closing, the relative alignment of the graph is changed. The final map is evaluated from the map pieces, which have the smoothest transition to the next segment (green).

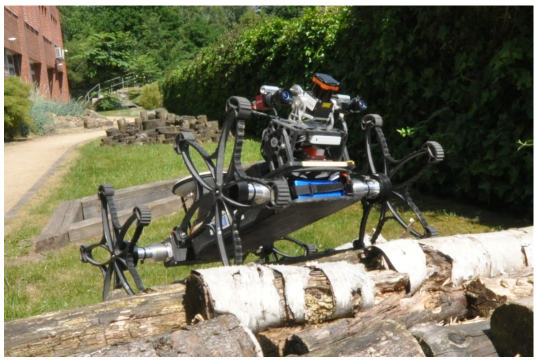


Fig. 4. The Asguard v3 [17] platform has a similar mechanical design compared to the Asguard v2 [18], but includes additional sensors and onboard processing. The main sensors used for this work are the wheel and joint encoders and an IMU, as well as the laser range finder.

$(x_1,\ldots,x_n)^T$ represent the pose x_k of each vertex with respect to a common frame. The sum of all errors given by the constraints can be given as

$$F(x) = \sum_{<i,j>\in C} e_{ij}^T \Omega_{ij} e_{ij}.$$

A method like presented in [12] can be used to find $x^* = \arg\min_x F(x)$ such that the error is minimized. The information matrix Ω_{ij} is the inverse of the constraint covariance matrix stored in the graph, while e_{ij} is an error function setting pose i and j into relation (see [12] for more details).

The first segment added to the graph G is added with an anchoring constraint to the origin of the common reference frame. Any consecutive segment S_{k+1} is added with at least one constraint, relating it to the previous segment S_k. Mean and covariance of the constraint are extracted from S_i as T and Σ. Apart from these sequence constraints additional constraints can be added between S_{k+1} and any Segment S_i with $i \leq k$. The number of constraints that have to be checked grows linearly with the number of segments. Checking for constraints between two poses, depending on the method, usually is fairly costly. For this reason, we use a heuristic based on the extends (bounding box) of the map content for each segment S_i, which gets extended by the transformed sigma points of the current pose uncertainty for that segment. If the extends for S_i calculated with this method overlap with those from S_{i-1}, these two segments can be evaluated for correspondence.

The actual correspondence checking between two poses is currently performed using visual means. SURF features are extracted from a stereo camera, and are compared to the features from the other segment. Outliers are rejected using an isometry constraint between the 3D positions of the features extracted from the stereo correspondence. As this is not the focus of this paper, no further details are provided here.

D. Final Maps

The pose graph G now provides all the necessary information in order to generate a current best estimate of a global map. After running the pose graph optimizer on the graph, each segment S_i has a pose information x_i which is a transformation from the segments frame to the common reference frame. The relative transformation between two consecutive segments, which is given by $x_i^{-1}x_{i+1}$ is then compared to the Gaussians (T,Σ) in the segment S_i.

Once the best set of poses has been found, for each of the segments, the map with the associated final pose for which the error $e_n = |p_n - x_i^{-1}x_{i+1}|$ is minimal for all n is selected as the map representative for that particular segment (see Fig. 3). Since the z position component of $x_i^{-1}x_{i+1}$ provides the z position of the pose at the end of the segment S_i, this information is used to improve the z error of the selected map. Since the z component of the positions was stored for each particle, the final error e_z is linearly distributed over the time steps $0:t$. Each map cell has an association of the time step k at which it was last updated. The mean m_μ for each map cell is therefore updated as $\hat{m_\mu} = m_\mu - e_z k/t$. If necessary for the application the corrected map segments can be converted into a single large map representation.

III. Experiments

The method proposed in Section II has been experimentally verified on the Asguard v3 system (see Fig. 4). The robot weighs around 14 kg and has a distance of 17 cm from the ground to the main axis. The rear axis can be freely rotated around the main directional axis of the system (twist joint). This makes sure that the Asguard is in contact with the environment with all four wheel/feet at any time. For sensing the environment, the system uses encoders on the wheels and the body twist joint. An inertial measurement unit from XSense provides orientation with respect to the gravity vector. The system uses a 2D laser scanner (Hokuyo UTM-30LX) which for these experiments was fixed at at a 30 degree forward facing angle. There a two cameras with a 25 cm baseline, each with a resolution of 768 x 480 pixels. The robot is equipped with a Core2 Duo running at 1.5 GHz and 2 GB of memory. The software installation uses the Rock

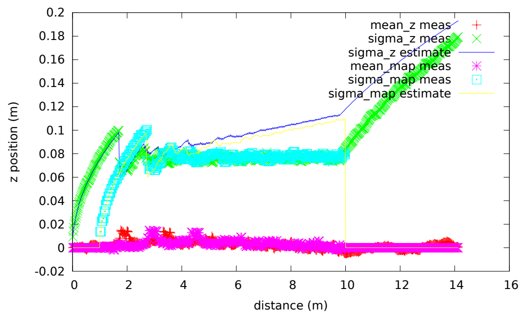


Fig. 5. Simulation run averaged over 500 iterations, which shows the estimation of the z position and map height, as well as the estimated and measured variances. Map surface patches at height 0 are perceived between distance 1 m and 10 m. In this region the map height could be compensated and the variance grows less slowly. It can be seen that the variance for both z-position and z-height do not match exactly, which is to be expected because of the approximations used, but is close enough to be useful.

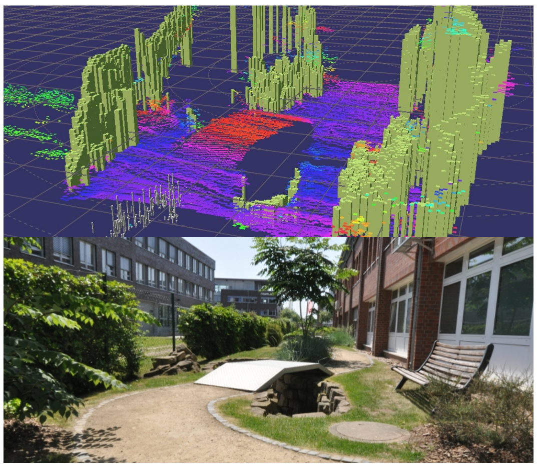


Fig. 6. In this experiment the Asguard v3 system was crossing a bridge on the test-track (bottom), which caused significant wheel slip. The method proposed for generating local maps using embodied data is able to smooth the result and provide a locally accurate map (top) of the environment.

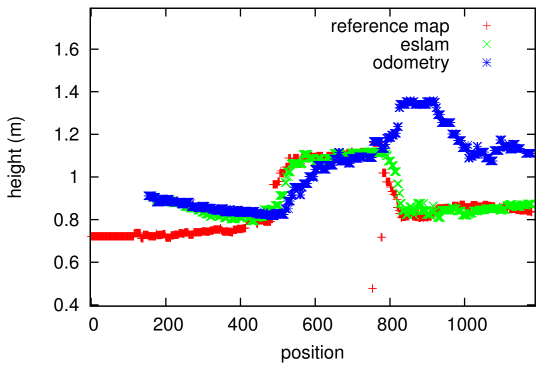


Fig. 7. Height profile of the map, which was generated passing the test-track bridge. The results from the embodied SLAM method are compared to the reference scan and the odometry.

framework[1]. Graph optimization is performed using the Hog-Man[2] framework. SURF feature extraction is done using the opencv image processing library.

The robot traverses a distance of ca. 250 m around the DFKI lab, parts of which is the robotic test track with a number of obstacles. The resulting map is compared to a map of the building which was acquired using 13 scans with a commercial Leica Geosystems HDS 6000 Laser Scanner. The reference map density is 0.5 pixel/cm, with an estimated alignment error less than 5 cm.

IV. RESULTS

The results section is split into three parts. Firstly, the method for estimating the z-position and the map cell heights is verified in a simulation environment. Secondly, the local map building is verified for a single map segment. This experiment was performed on the Asguard v3 robot on the DFKI testtrack. As a final result, a loop around the DFKI building was performed with the Asguard v3 robot, and the resulting map compared against a high-resolution scan of the building.

The method for estimating the z-position of the robot which was proposed in Sec. II-B has been evaluated in a simulation environment. The Asguard system was modeled on a flat surfaced environment. The position was perturbed by odometry noise, and map measurements by measurement noise. Fig. 5 gives the averaged results of a simulation run, which shows that the approximations made are viable. The pose and map variance are approximated within tolerable boundaries. In this result, the position mean has a slight bias, which is due to the geometry of the Asguard system.

In order to evaluate the generation of local maps using the visual to embodied data correspondence, a metal bridge was traversed on the test-track. During the run, the wheels had a significant amount of slip. Using the contact model from [6] in the mapping process, this slip was compensated, as can be seen in the map generated by the particle with the highest weight in Fig. 6. The height profile of the trajectory was compared against the reference map from the commercial laser scanner and the map generated by pure odometry in Fig. 7. While the odometry fails to track the terrain profile and generates a height error of ca. 0.6 m, the embodied SLAM stays within 5 cm accuracy.

In the final experiment, the Asguard v3 system traversed a loop around the DFKI building. The maps in Fig. 8 show the map before and after loop closing. The visualization shows the individual map segments and the robot path for each individual particle with reference to the local frame. The segments are connected such that each new segment starts at the mean of the final positions of the previous segments particles. After loop closing, the segments relative position is updated according to the constraint graph. The final map is generated based on the maps of those particles that generate the smoothest path. Fig. 9 shows the transition between two

[1]http://www.rock-robotics.org
[2]http://openslam.org/hog-man

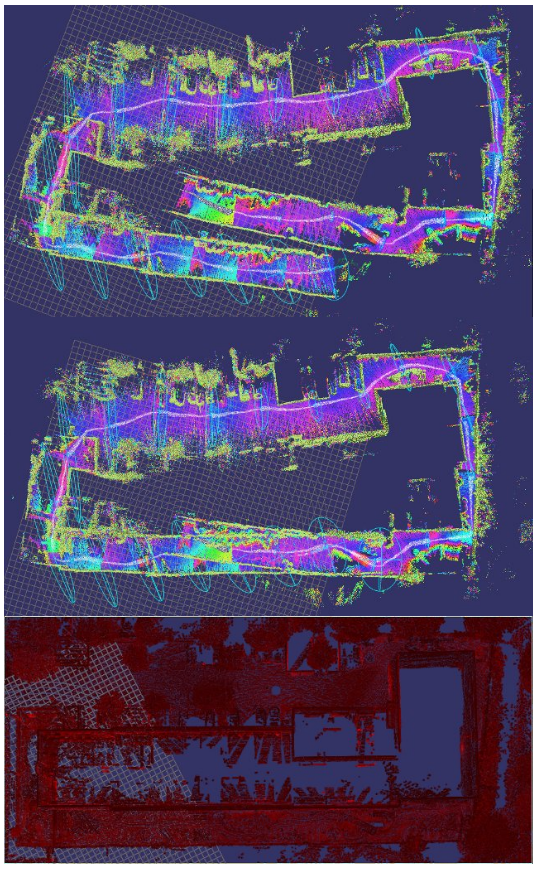


Fig. 8. Loop around the DFKI building before (top) and after (center) loop closing. Loop closing is performed by matching stereo image features. When compared to a scan of the building (bottom) shows that the optimized map is locally consistent, but contains distortions.

map segments in greater detail.

V. CONCLUSION

In this paper, the problem of simultaneous localisation and mapping in difficult outdoor terrain was addressed. It was proposed to solve the problem of representing the full state space by factoring the problem into multiple parts. A FastSLAM based particle filter was used to factor x and y position, heading and maps. The estimation of z position of the robot and the map cells for each particle is performed using a Kalman filter. For larger scenarios these local maps are split into map segments, which are connected using a pose constraint graph. A method was proposed to generate globally smooth maps based on selecting the optimal particle for representing the segment. In this way it is possible to generate locally as well as globally consistent maps. The particle depletion problem of FastSLAM is avoided, while at the same time keeping the ability to generate locally consistent map. For the generation of the local maps, a method for associating visual with embodied data was developed, which has a low computational requirement and provides an

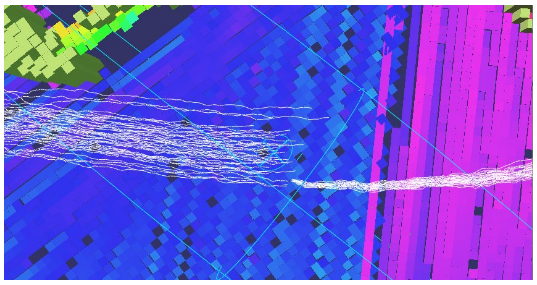


Fig. 9. Enlarged top view of a transition between two segments of the globaly optimized map. The particle trajectories of the previous segment (left) are connected to the trajectories on the following segment (right). Map parts are selected from the individual segments after optimization in such a way that the robot path is kept smooth. This ensures smooth map transitions and globally matching maps.

additional means of generating map constraints.

Currently the constraints between the local maps are generated by calculating the mean and covariance of the particles poses. In some cases this might be a poor approximation of the distribution. On possible way to extend the proposed method would be to use a mixture of Gaussian representation and use an iterative method to pick the most likely Gaussian of the mixture for the graph constraint.

Another limitation is the storage required by the proposed method. All maps for all particles have to be kept, compared to FastSLAM, which can remove maps from eliminated particles over time. In some cases it might be advantageous to only store the trajectories and sensor inputs, in order to regenerate the maps required for the final map evaluation.

VI. ACKNOWLEDGMENTS

The work presented here is part of the Project ”Intelligent Mobility”, which is funded by the Federal Ministry for Economics and Technology (BMWI) through the German Space Agency (DLR) grant number 50 RA 0907.

REFERENCES

[1] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, “Mapping and localization with rfid technology,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 1015–1020.

[2] J. Gutmann and K. Konolige, “Incremental mapping of large cyclic environments,” in *Computational Intelligence in Robotics and Automation, 1999. CIRA’99. Proceedings. 1999 IEEE International Symposium on*. IEEE, 1999, pp. 318–325.

[3] M. Karg, K. Wurm, C. Stachniss, K. Dietmayer, and W. Burgard, “Consistent mapping of multistory buildings by introducing global constraints to graph-based slam,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5383–5388.

[4] K. Berns, K.-D. Kuhnert, and C. Armbrust, “Off-road Robotics - An Overview,” *KI - Künstliche Intelligenz*, 2011.

[5] J. Schwendner and F. Kirchner, “eSLAM - Self Localisation and Mapping Using Embodied Data,” *KI - Künstliche Intelligenz*, 2010.

[6] J. Schwendner and S. Joyeux, “Self Localisation using Embodied Data for a Hybrid Leg-Wheel Robot,” in *IEEE International Conference on Robotics and Biomimetics*, 2011.

[7] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (slam): Part ii,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[8] U. Frese, “Interview: Is slam solved?” *KI - Knstliche Intelligenz*, vol. 24, pp. 255–257, 2010, 10.1007/s13218-010-0047-x.

[9] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the AAAI National Conference on Artificial Intelligence*. Edmonton, Canada: AAAI, 2002.

[10] C. Stachniss, D. Hahnel, and W. Burgard, “Exploration with active loop-closing for fastslam,” in *Intelligent Robots and Systems (IROS)*, vol. 2. IEEE, 2004, pp. 1505–1510.

[11] S. Thrun and M. Montemerlo, “The GraphSLAM algorithm with applications to large-scale mapping of urban structures,” *International Journal on Robotics Research*, vol. 25, no. 5/6, pp. 403–430, 2005.

[12] G. Grisetti, R. Kummerle, and C. Stachniss, “Hierarchical optimization on manifolds for online 2D and 3D mapping,” in *IEEE International Conference on Robotics and Automation*, 2010.

[13] N. Fairfield, D. Wettergreen, and G. Kantor, “Segmented SLAM in Three-Dimensional Environments,” *Journal of Field Robotics*, vol. 27, no. 1, pp. 85–103, 2010.

[14] R. Triebel, P. Pfaff, and W. Burgard, “Multi-level surface maps for outdoor terrain mapping and loop closing,” in *Proc. IEEE IROS*, 2006.

[15] A. Doucet, S. Godsill, and C. Andrieu, “On sequential Monte Carlo sampling methods for Bayesian filtering,” *Statistics and computing*, 2000.

[16] N. Gordon, D. Salmond, and A. Smith, “Novel approach to nonlinear/non-gaussian bayesian state estimation,” in *Radar and Signal Processing*, vol. 140, no. 2, 1993.

[17] S. Joyeux, J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig, “Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI,” *KI - Künstliche Intelligenz*, 2011.

[18] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner, “Asguard: a hybrid legged wheel security and sar-robot using bio-inspired locomotion for rough terrain,” in *IARP/EURON Workshop on Robotics for Risky Interventions and Enviromental Surveillance*, 2008.

Leaving Choices Open in Planner/Planner Integration[*]

Sylvain Joyeux and **Frank Kirchner**
DFKI Bremen, Robotics Innovation Center
Robert-Hooke-Strae 5
D-28359 Bremen

Abstract

In this paper, we analyze the shortcoming of today's hierarchical planner/planner integration methods. We will show that, in these integration methodologies, one planner has usually to take decisions that are not limited to its "area of expertise" (path planning, low-level planning, high-level goal scheduling, . . .). It can therefore make ill-informed decisions and make choices that could have been left open and used to improve planning and scheduling. We propose a different view on planner/planner integration, where one planner does not provide a single solution but instead filters out qualitatively non-optimal solutions, thus returning a set of near-optimal solutions. Making the actual choice is then delegated to either other planners, or during execution when missing relevant information is made available to the system. We will support this approach by outlining our own work in the domain of path and task planning integration.

1. Introduction: Decision-making in Robotic Systems

What is decision making in robotic architectures is a difficult question, and we do not pretend to give a definitive answer here. This section will instead try to give, through examples, a feeling for the following three categories that can be found in a robotic system:

- components definitely *not* taking decisions.
- components definitely taking decisions.
- a "grey area" where it is not so clear-cut

No decision making Low-level behaviours like "go forward" or "turn left" are obviously *not* decision-making components. Another example is a module to control motors, or a pilot module for an UAV. It does not *choose* anything: it takes a very short-term command (for instance the desired path the robot should follow) and tries to make the system stick to that command regardless of external influence (in case of side wind for an UAV for instance).

[*]Work done in the Intelligent Mobility project, funded by the German DLR, with funds from the Ministry of Science (BMBF).

Decision-making components Action planning deals with deciding the future actions of the robot, given its goals, its current state, a model of its environment and a model of how it can change its own state in this environment. Action planning is definitely in the domain of decision making: it chooses the combination of actions that are the most like to make its reach its stated goal(s) *among all the possible combinations of actions*.

Grey area In there lies *path planning*, or – more generally – all processes which involve choosing the place the robot should move. This might be, for instance, a perception planner which tells where perception should be done. These components get a goal and deal with choosing the right path for the robot to reach it. One could argue that given the environment cost function and the geometrical model of the robot, there is only one optimal path. The theoretical optimality of a path in this context of a cost function, though, is often in practice one among a set of *qualitatively equivalent* solutions. Thus, from the robot mission point of view, most path planning algorithms choose a single path in all the possible equivalent ones. They make a decision which impacts the whole system.

Definition This can be seen as a definition of decision-making. Any component which *chooses* one of the possible, sensible courses of actions of the robot makes a decision. As such, they are all *planners*: they do a projection of the robot into its possible futures and choose one (or many) among them. As we will see in the next section, it is important to understand the interaction between the different decision-making components – and to allow them to have rich interactions – as these interactions are critical for the overall robot performance.

2. Problem Statement

In this section, we will outline issues related to how planners are usually integrated in robotic systems. Then, we introduce our proposed approach through the example of path- and task- planning integration. Finally, we will describe this methodology in a more general context.

In most systems, path planners are integrated using what one could call the "MoveTo(goal)" paradigm. As we outlined in the previous section, they are asked to reach a place, and choose one solution that would allow the robot to reach that place. From a system point of view, this would not be a problem if the path planner was informed enough to take into accounts all the mission parameters. That's the main issue: the path planner has to concern itself with problems that go out of the "robot motion" domain, as for instance possible robot-robot interactions or scheduling issues.

For instance, a good solution for the classical rover-AUV navigation scenario – where the rover has a goal to reach and the AUV can provide some mapping information to it – is to have the rover take into account the AUV's task scheduling in its navigation decision, i.e. decide to take a route over another based on *if* and *when* the AUV will be able to provide him some data about the possible routes.

This problem also appears between action planners: in architectures where action planning is split into hierarchies – as for instance in (Alami et al. 1998; Alami and Botelho 2001) – the lower layers have only a partial knowledge of the plans of the upper layers: their knowledge is limited to the actions that should be executed right now. In these schemes, there is not the possibility to choose, in the lower layers, the optimal actions *given the future plan of the upper layer*. The lower layers can therefore over-constrain the upper layer.

To summarize our point here, decision are not always made in the component that is the most well-informed to make it. Note that this issue remains in systems where the planning and execution layers are tightly integrated through a common plan model as CLARAty (Volpe et al. 2000; 2001; Nesnas et al. 2005), IDEA (Muscettola et al. 2002; Finzi, Ingrand, and Muscettola 2004) and the Concurrent Reactive Plans (Beetz 2000). It remains, because for the planning problem to be tractable, one *must* split it into smaller sub-problems and, until now, that involves using some components tailored for a specific level of details and/or domain. And, in the current integration schemes, these components will take into account parameters unrelated to their particular focus. These components are then making choices that should have been done by a more informed or better suited layer.

2.1 The Example of Path Planning Integration

In this section, we will be concerned only with coarse-grained path planning, meaning approaches that are commonly used in outdoor navigation when the whole system's configuration does not need to be taken into account. As we already identified in the previous section, most path planners search for an optimal solution in the geometrical space (for instance in traversability maps) while, given the scale they are operating at, searching optimality is actually meaningless. Indeed, the uncertainty on the travel time or energy consumption is *far greater* than the difference between the optimal path and the next-to-optimal ones.

We therefore advocate to start seeing a path planner as a *filter*: it would take the whole geometrical (or configuration) space and finds out the *set of solutions* that are qualitatively equivalent from the system's task plan point of view. Indeed, these solutions will be, in practice, equivalent at execution time and therefore a choice cannot be made by pure "travel-oriented" concerns (time, energy).

So, in our scheme, the path planner does not output one path anymore but a set of solutions. Other components in the system then take care of choosing one actual path in these sets. The advantage is then that one can take into account concerns that would be difficult to put in the path planner's cost function. For instance:

- in multi-robot contexts, it would enable opportunistic behaviors during path execution: for instance, taking images of a place that another agent requires, without impeding the main robot's task.
- generation of contingency plans, where the task planner takes into account both a nominal movement and the ability to use alternate paths if the nominal path is blocked.
- the proper handling of unknown regions. In outdoor environments, the robot badly knows the environment itself. It must therefore decide whether it should explore an unknown (or badly known) region or if it should take a known path that is potentially longer. This decision is currently implicitly made by the system's path planner through the use of *a priori* values for the unknown parts of the map. In other words, the robot designer assumes the unknown to be of a certain type. An integrative approach could instead use the task planning system that has access to both the mission profile (is there enough time and energy for exploration?), and the consequences of a failure (what would be the cost of a failure?) to make that decision.

Finally, the *stability* of a path plan is of great importance for their integration in task plans. Optimal approaches, like D*-based approaches, become unstable around "crossroads" (where two completely different paths of almost equal costs meet each other). Indeed, little changes in the robot's environment model, or position estimate, can make the path costs change in a way that make the algorithm switch between them. The inability of these path planning approaches to provide a stable plan makes them hard to use in multi-robot contexts: there is no way to assume how a robot is committed to its currently chosen path. The authors have already observed the consequences of such instability in a bi-robot setup where the UAV uses the rover's path to generate the regions to map (Joyeux, Lacroix, and Alami 2007). Again, that problem would probably be mitigated if the multi-robot cooperation component becomes aware of the solution set from which the execution system will picks the executed path.

2.2 Proposed Integration Scheme

What we basically advocate in this paper is to change the way planners are integrated with each other. Through the path planner example, we will see that a hierarchical planner/planner integration can be achieved by having the planners acting as *filters*. In this scheme, one planner does not output one solution anymore, but instead a set of solutions that seem sensible at the level of detail and given the domain of this particular planner.

The following section will further develop the example of path- and task- planning integration, by presenting an implementation of the proposed scheme. Then, the next section will outline how the added information can used during plan building and execution to improve the robot's overall behaviour. Finally, we will conclude by ways to generalize the proposed path planning approach to other types of planning like MDP, and describe some future work.

3. Choices in Plans

The matter of integrating choices in plan representations is not something new in itself. One can see conditional planning and MDP as allowing the representation of open choices by defining "choice tasks" which have multiple outcomes.

The issue, in our opinion, is not to represent the action of making choices, but to represent in the plan the information that is needed to link the actions and the decisions. Indeed, one unique feature of representing open choices in plans is that one can then represent the actions that are required to make these decisions well-informed. Planners would then have to stop seeing the plan as a static problem (static forward model) but really as a dynamic problem where tasks can influence decisions. The GPGP/TAEMS (Lesser et al. 2004) is an example of such a plan model.

In this section, we will see how it is possible to represent and generate such choice plans in the case of the path planning problem, and how those plans have been integrated into our plan manager (Joyeux et al. 2009; Joyeux 2008).

3.1 Corridor Plans

The basic idea behind our representation is that, at the level of abstraction where path planning works, searching for an *optimal* trajectory is meaningless: first, especially in outdoor environments, having a very precise map is impractical for dynamic systems, and therefore the maps the path planner has to work on are imprecise. Second, it is actually expected that the underlying motion planner will seldom follow the generated trajectory, instead using it as a guidance. Our representation is instead based on the idea that the navigation should be *qualitatively* optimal: the goal is not to try following the optimal trajectory, but to represent and structure the regions where the robot should progress so that the overall movement is near-optimal. The resulting plan will therefore be both a geometrical representation of this region and the necessary information to direct the robot in it.

At the geometrical level, this region is segmented in "corridors": a *median line* and two boundary curves. Each corridor has strictly two end zones, represented by the line between the ends of the median line and the associated ends of the boundary curves (Fig. 1). From the execution point of view, a corridor represents one step in the navigation: if a robot enters a corridor from one side, it will exit it at the other side.

These corridors are structured into a topological plan, which represents all desired navigations (sequences of corridor passing) in a compact way. In these plans, a corridor C can be either directed or bidirectional. In the first case, all plan executions (corridor sequences) that include C travel through it always in the same direction. In the second case, there are plan executions that travel through C in one direction and other in another.

To represent the topological plan, we chose to link the corridor sides with edges of two types:

- edges that represent the possible direction of travel inside the corridor itself (intra-corridor edge). There is one intra-corridor edge in directed corridors and two edges for bidirectional corridors (one for each direction).
- connections between sides of two different corridors, which represent the possibility to go from one corridor to the other.

In this representation, a well-formed *path*, that is a corridor sequence, is an alternance of edges of the two types. In other words, once an inter-corridor edge is taken, the robot must take an intra-corridor edge (i.e. cross the corridor), and vice-versa. In other words, the robot cannot go back into the corridor it just crossed.

While similar to topological maps (see (Thrun 1998) for both an interesting method around topological maps and the related work in that field), it is not a representation of a map, but of a plan: it is a compact representation of the set of corridor sequences (plan executions) that lead to the robot's goal in a near-optimal way. Our contribution here is that we are able to generate such a plan in an unstructured outdoor environments, using the cost function to structure the space. The downside of it being that, like all plans, the topological representation may change if the goal changes, while maps will only be modified if the environment itself changed.

3.2 Generation of Corridor Plans

For the plan generation algorithm, we assume that the environment is represented as a navigation function sampled on a grid map. The navigation function gives the optimal cost to the goal for each point in the map and can for instance be generated by D* or one of its offspring (we're using D*). The first phase of our algorithm is to transform the navigation function into a function returning the total cost to go through that point of the grid. For each point, the algorithm computes the added cost of first going to that point and then going from that point to the goal. The algorithm can then very simply find the "navigation region", which includes all points whose *total* cost is within a given margin of the optimal cost. That "cost margin" can be set in different ways, as for instance (i) a fixed value which represents the allowed margin on the overall cost, (ii) a ratio of the optimal cost to goal or (iii) based on the uncertainty of the optimal path (if uncertainty informations is available).

Then, structure is extracted from the navigation region by using a Voronoi skeleton extraction algorithm. Voronoi diagram extraction is a widely studied subject, and we therefore do not describe it here. See for instance (Li, Chen, and Li 1999).

However, the Voronoi diagram only generates an *undirected* graph of corridors. There is still the need to generate a directed graph. The algorithm is as follows:

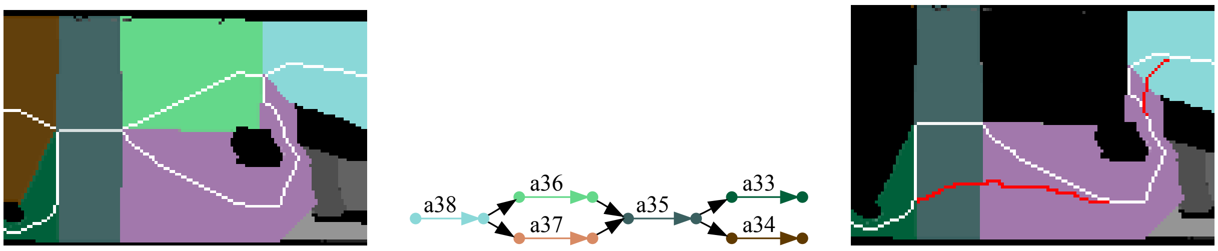


Figure 1: Detail of a corridor plan which gets around an obstacle. The geometrical representation is on the left, with the obstacle in black and the median lines in white. The topological representation is on the middle: the plan dictates that the robot should come from a38 (blue) to a35 through either a36 or a37. On the right, a specific corridor sequence has been chosen for execution by the motion planner and the median line is updated accordingly (red parts).

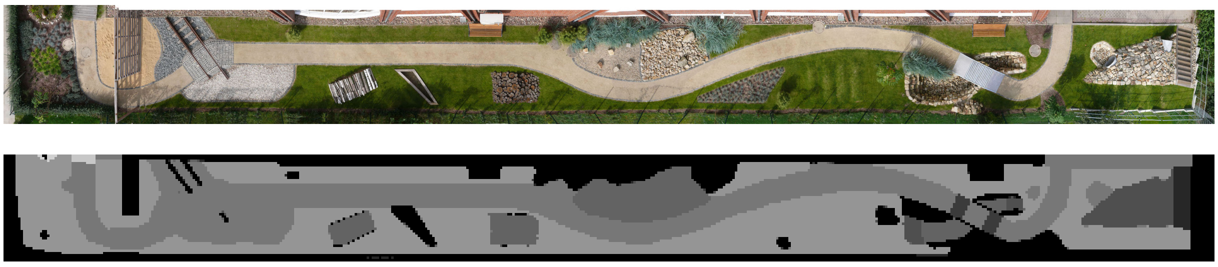


Figure 2: Up: photo of our outdoor test track at DFKI. Down: the hand-made classification per terrain type. Terrain types are internally associated with average speed values. The cost is time to travel.

- for the algorithm to work, the cost along the corridor median lines need to be monotonic – in which case we do the approximation of considering that the whole corridor is monotonic w.r.t. the cost to target. Corridors for which it is not the case are split into two new corridors.
- the corridors are oriented w.r.t. cost to target: the entry point whose cost is higher is the entry point of the corridor and the other one the exit point (remember that navigation is following the down-gradient of the navigation function).
- then, a depth-first search starting at the robot's start point allows to find all the paths that reach the goal in a near-optimal way. The basic idea is to limit the *cost overhead* of any explored path, that is the accumulated cost of traversing a corridor backwards with respect to the orientation determined in the previous step.

The result of this process can be seen on Fig. 3. A detailed explanation, that can help interpreting this plan, is on Fig. 4.

3.3 Choices in the Plan Manager

In our architecture, the plan manager is a component that allows to represent a global system plan, execute it and also modify it while it is executed. We will not dwelve into much details in this paper, as it is not required to understand the principle of the performed integration. One should refer to (Joyeux et al. 2009; Joyeux 2007) for more details about the model, how the plan manager executes its plans and how it is tailored for multi-robot execution.

In a few words, our plan model is made of two types of objects: *events* and *tasks*. The first represents timepoints and the second represent processes. These objects are structured into graphs (mostly DAGs), to form a plan. These graphs, or *object relations* allow to represent:

- the execution flow by linking events together
- how tasks interact with each other thanks to various task relations. Examples of such relations are the *dependency relation* where one task depends on having another achieve something (reach an event). In our scheme, the *influence relation* is used, which represents the ability of one task to positively or negatively impact the execution of another.
- the links between the different levels of abstraction by a mixture of task relations, event relations and an object-oriented model.

Choices in our plan manager are a special kind of event. That event represents that, once this particular timepoint is reached, one of the possible plan executions (the event that follow the choice node) has to be chosen (Fig. 5). At all times, if choice nodes are still present in the plans, it means that *at this stage of execution*, there was no compelling reasons to prefer one possible execution path against another.

Finally, the influence of other robot activities on the ability to perform that choice is represented by an *influence* rela-

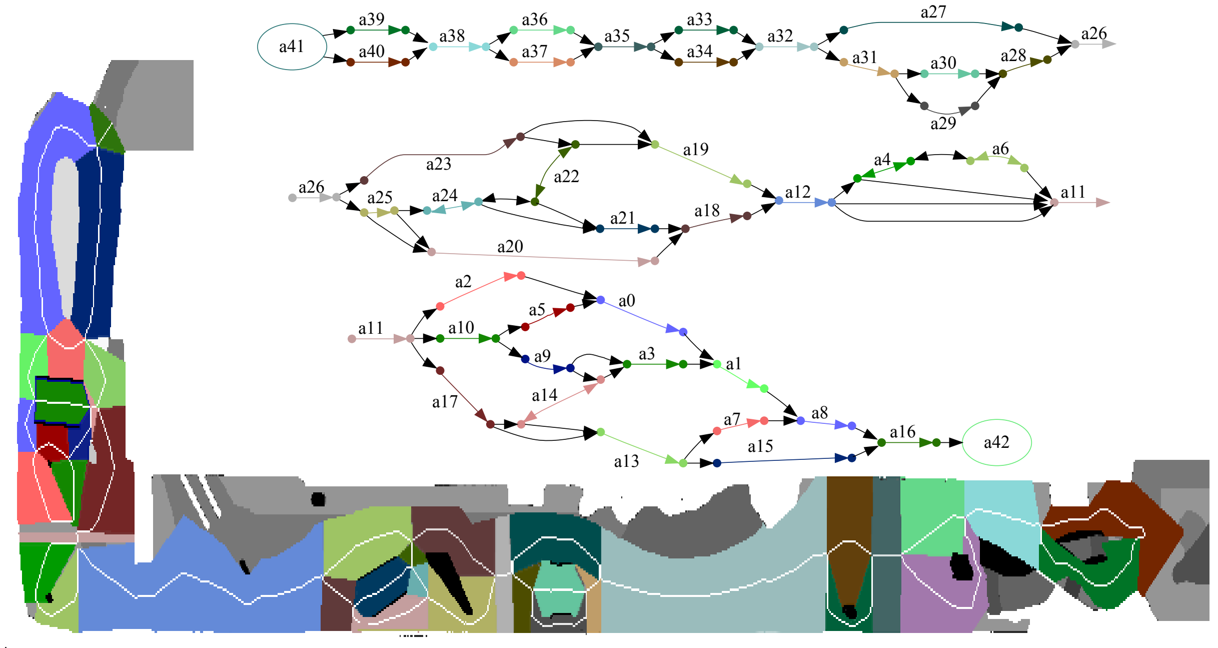


Figure 3: Generated corridor map on the full terrain, for a navigation starting at the bottom far right (on the line between the green and brown regions). The target point is up-left, in the green region. The resulting topological map is presented on the top. Note that each line continues the previous one (for instance a26 is connected to a28 on the right of the first line and to a25 on the left of the second line). The corridor names have no meaning. Detailed explanation of some complex features is presented on Fig. 4.

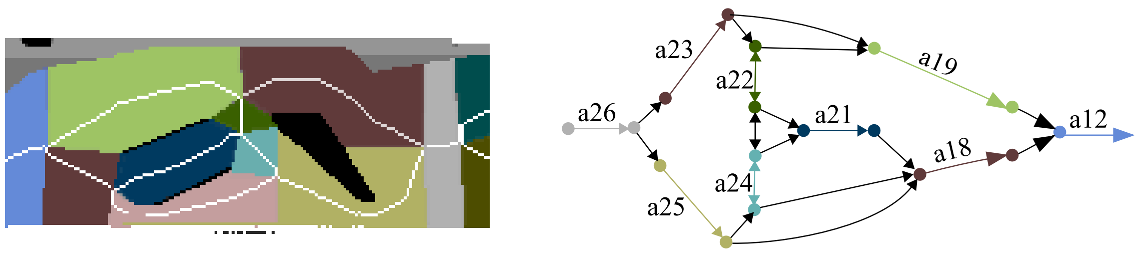


Figure 4: On this detail of the global plan, the concept of bidirectional corridors can be explained. A robot that comes from a23 has first two choices: either go straight to a19 or to go through a22. If it crosses a22, then it has only two choices: either go through a24 and a18 or a21 and a18. It cannot go back a22 because that plan would be ill-formed (see section 3.)1. Symetrically, a robot that comes from a25 can cross both a24 and a22 to go up and reach either a21 or a19. a24 and a22 are therefore bidirectional.

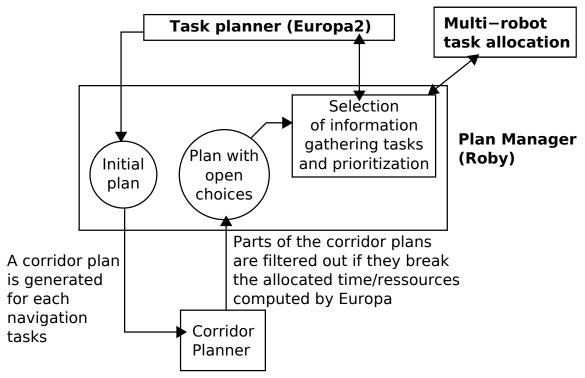

Task planner (Europa2)
Multi−robot task allocation
Selection of information gathering tasks and prioritization
Plan Manager (Roby)
Initial plan
Plan with open choices
A corridor plan is generated for each navigation tasks
Parts of the corridor plans are filtered out if they break the allocated time/ressources computed by Europa
Corridor Planner

Figure 6: Integration scheme of our path planner with an actual task planner through our plan manager.

tion borrowed from the GPGP/TAEMS model (Lesser et al. 2004). This relation marks that some of the outcome tasks are influenced by the task that will provide the necessary information.

4. Exploiting Decision Information

The most straightforward use of the decision information is actually static: it can be integrated in a standard planner to take into account concerns of safety (based on how wide and/or well-known a corridor is) and redundancy (how many alternate paths exist in a subgraph of the corridor plan, to commit the robot as little as possible).

A more interesting approach we already mentioned is the ability for the robot to add information gathering tasks when they provide the information influencing the robot's decision. What we then want is to add and schedule, in the plan itself, those tasks that would improve the robot knowledge about its available options. The scheme we plan to develop is presented on Fig. 6. We're planning to use Europa2 as a task planner, the algorithm presented in this paper for path planning and our Roby plan manager. How we would model the influence of one task on the choice node is still open to us.

The problem with this scheme is that Europa2 itself will not be able to take into account the corridor plans. The issue is that Europa2's plan database, to our knowledge, does not handle conditional plans. It will therefore not be possible to use the availability of choices in Europa2. We will instead handle it in the plan manager itself. Nonetheless, we assume it will allow us to handle interesting cases, especially in multi-robot contexts.

Finally, based on these corridor plans, we could extend the approach to making contingency plans for navigation. Assuming that it is possible to derive a likeliness that traversing a given corridor fails, it would be possible to build a relationship between both the *nominal* plan and the plan where traversing fails, thus allowing decisions to be taken based on an online estimate of the cost of failure. We already developed the corridor update operator that is needed to implement this approach, but its description is out of this paper's scope.

5. Conclusion

In this paper, we argued that a richer approach for planner/planner integration is desirable. We used there the path planning as the primary example, because the handling of path plans stemmed our reflexion on the subject. We presented how we achieved to build a prototype for a path planner that supports that scheme, and how we plan to integrate it further in a complete robotic system.

The principle behind our path planner implementation is actually applicable to other types of planning, and in particular MDP planning. In MDP policies, one has a representation of the optimal cost of reaching the goal starting from each of the policy's state. Therefore, the idea of computing the set of paths in the policy that would allow to reach the goal in a near-optimal way is valid. Such a scheme could be used to make cooperative decision-making between MDP and classical planners, or more generally use the MDP planner as a plan-space filter to be integrated in a classical framework like our plan manager.

What we generally propose is to start thinking of each planner as a filter instead of being a decision-maker. In our point of view, one planner should concern itself only with the parameters that are relevant to its level of detail and domain of application, and produce not *the* optimal plan (given its cost function) but the set of plans that are qualitatively equivalent to the optimal one, thus leaving the actual choice to the upper layers (if any).

References

Alami, R., and Botelho, S. 2001. Plan-based multi-robot cooperation. In *Advances in Plan-Based Control of Robotic Agents*, Lecture Notes in Computer Science. Springer.

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research* 17(4):315–337.

Beetz, M. 2000. *Concurrent Reactive Plans*. Springer-Verlag.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Robot action planning and execution control. In *Proceedings of IWPSS*.

Joyeux, S.; Philippsen, R.; Lacroix, S.; and Alami, R. 2009. A plan manager for multi-robot systems. *The International Journal of Robotics Research*.

Joyeux, S.; Lacroix, S.; and Alami, R. 2007. A plan manager for multi-robot systems. In *Proceedings of FSR*.

Joyeux, S. 2007. *A Software Framework for Plan Management and Execution in Robotics: Application to Multi-Robot Systems*. Ph.D. Dissertation, ISAE. `http://tel.archives-ouvertes.fr/tel-00283086/fr/`.

Joyeux, S. 2008. The roby plan manager. `doudou.github.com/roby`.

Lesser, V.; Decker, K.; Wagner, T.; Carver, N.; Garvey, A.; Horling, B.; Neiman, D.; Podorozhny, R.; NagendraPrasad, M.; Raja, A.; Vincent, R.; Xuan, P.; and

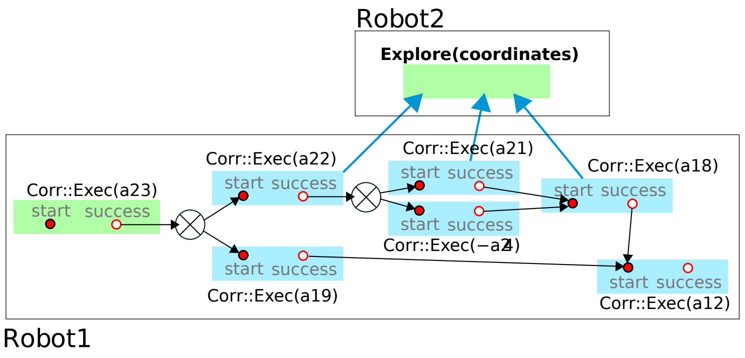


Figure 5: Example plan with choice nodes and influence relation. Parts of the corridor plan of Fig. 4 has been translated into our plan manager's model. The circles are the events, the crossed circles the choice nodes and the event-to-event arrows precedence relations (possible execution traces). Then, a hypothetical robot-robot interaction is set up where the second robot influences the first robot's choice by performing some perception task. This relation is represented by the three arrows pointing at Robot2's Explore task (these arrows should be read "parent is influenced by child"). In our manager's scheduler, the choices are removed when done, allowing to simplify the plan as the choices. This is of course a trade-off in itself, as new gathered information cannot modify these choices anymore.

Zhang, X. 2004. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems* 9(1):87–143.

Li, C.; Chen, J.; and Li, Z. 1999. A raster-based method for computing Voronoi diagrams of spatial objects using dynamic distance transformation. *International Journal of Geographical Information Science* 13.

Muscettola, N.; Dorals, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. IDEA: Planning at the core of autonomous reactive agents. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Nesnas, I.; Simmons, R.; Gaines, D.; Kunz, C.; Diaz-Calderon, A.; Estlin, T.; Madison, R.; Guineau, J.; McHenry, M.; Shu, I.-H.; and Apfelbaum, D. 2005. Claraty: Challenged and steps toward reusable robotic software. *International Journal of Robotic Research* 3(1):23–30.

Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence* 99(1):21–71.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2000. Claraty: Coupled layer architecture for robotic autonomy. Technical report, NASA Jet Propulsion Laboratory.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2001. The claraty architecture for robotic autonomy. In *Aerospace Conference*, 121–132.

Towards a better integration of path planning for outdoor environments

Sylvain Joyeux and Prof. Frank Kirchner

Abstract In this paper, the shortcomings of the currently used integration methods for path planning in outdoor environments are analyzed, and a representation aimed at correcting them is proposed. We build upon the notion of topological maps widely studied in indoor environments to create a representation that allows the use of a cooperative decision-making process where a long-range path planner acts as a middle-man between the task plans on the one hand and the short-range motion planner on the other hand.

1 Introduction

In general, by *autonomous navigation*, one means the process of autonomously moving the robot to achieve a task, be it through planning-based or behavior-based approaches. This paper focuses on planning-based approaches, where the robot must reach a certain location to achieve a task. In the literature, two classes of algorithms are developed to solve this problem.

The first class handles the process of generating a set of precise robot commands which take into account all the geometrical constraints induced by the environment and the actuation limitations of the platform. In general, it is expected that the execution of the motion plan (usually generated in configuration space) will very precisely follow the generated plan [8, 7, 9, 10]. In this paper, these algorithms will be called *motion planning algorithms*.

The second class is the process of generating an object – usually in geometrical space – that the robot should follow as closely as possible. Unlike for motion planning, it is used as "guideline" for the motion execution subsystem, which is expected to follow it "well enough" for the navigation to succeed. In this paper, these algo-

DFKI Bremen, Robotics Research Group, Robert-Hooke-Strae 5, D-28359 Bremen e-mail: firstname.lastname@dfki.de

rithms will be called *path planning algorithms*. The most widely used algorithms of this class are A*, its cousin D* [14] and their derivatives [12, 1].

1.1 Robust long range navigation

On the one hand, motion planning algorithms are usually very costly in memory and computation time, and therefore can only be applied for short-range movements. On the other hand, path planning algorithms cannot be used to tackle difficult situations where the whole robot body constraints should be taken into account. Therefore, a common way to handle long range navigation (more than 100m) in unstructured environments is to combine motion planning and path planning algorithms.

Three ways to do this integration exist in the literature:

- a *distance to the path* is integrated in the motion planner's cost structure. This can be done either through a sampling of waypoints [4] or by trajectory following techniques [3].
- having the path planner generate a navigation function, which is then used by the motion planner as the "cost to target" of each considered motion [2]. This way, the motion planner is in theory choosing a motion that is globally optimal.

These approaches are known to lead to *decision conflicts*, where the path and motion planners take conflicting decisions, leading to undesired behaviors (Fig. 1). In our opinion, this is mainly because the two cost structures differ: in the first two methods, the generated path needs to be integrated in the motion planner's cost structure, which should therefore integrate both motion planning considerations (short range motion difficulty) and the cost element of moving away from the reference path. It is a difficult balance to find. One could think that the third case is immune to this problem. However, if one would use the same cost (for instance time), the order of magnitude of the long-range path far outweighs the order of magnitude of the motion planner's cost. Therefore, the order of magnitude of the *error* in the long-range cost is of the same order as the actual *cost* of the motion plans. Therefore, a balance still needs to be found.

More recently, in [13], one planner (a D* algorithm) is working on a graduated-fidelity graph in which some section are of low-details and some other sections are developed to include the motion constraints of the system. There is therefore only one system doing both path and motion planning. Given that it allows one to have an adaptative level of detail, the aforementioned problems are mitigated. Nonetheless, the graph complexity must be kept below a certain level for the problem to be tractable, and our second point (the weight of long range vs. short range costs in the plan) will still hold. Moreover, it is still subject to plan instability, which is discussed in the next section.

1.2 Integrating task plans and autonomous navigation

From a task planning point of view, planning approaches are integrated as “point-to-point”: the planner generates a task of the form “get to point A”, and when needed calls the navigation subsystem to perform that task. It is a very limited form of interaction, and limits the robustness one can get from a more advanced supervision of the navigation tasks. In our opinion, integration approaches should tend to optimize the use of required motions, so that the robot rarely has to unnecessarily return to a point it was at before. For instance by, in multi-robot contexts, enabling *opportunistic* behaviors during path execution. Or by allowing the task-level supervision to build contingency plans where the various navigation possibilities are taken into account.

Moreover, in outdoor environments, the environment itself is badly known by the robot. The robot must therefore decide whether it should explore an unknown (or badly known) region or if it should take a known path that is potentially longer. This decision is currently implicitly made by the path planner because unknown parts of the map are assigned an *a priori* value – in other words, the unknown is assumed to be of a certain type by the robot designer. Obviously, a better decision would be achieved by a symbolic decision making system that has access to both the mission profile (is there enough time and energy for exploration ?), and the consequences of a failure (what would be the cost of a failure ?).

Finally, the *stability* of a path plan is of great importance for their integration in task plans. Optimal approaches, like D*-based approaches, become unstable around “crossroads” (where two completely different paths of almost equal costs meet each other), because little changes in the robot’s environment model, or position estimate, can make the algorithm switch between the two possible paths. The inability of these path planning approaches to provide a stable plan makes them hard to use in multi-robot contexts: there is no way to assume how a robot is committed to its currently chosen path. The authors have already observed the consequences of such unstability in a bi-robot setup where the UAV uses the rover’s path to generate the regions to map [5].

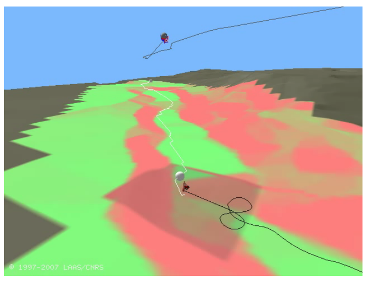


Fig. 1 Simulation result of a navigation task where the path (generated by a D^* algorithm) is regularly sampled, each point being fed as a subgoal to a motion planner. The “eight” structure is the result of a decision conflict between the two levels. The authors have seen similar artifacts appear on approaches using different algorithms but similar integration methodology.

1.3 Problem statement

This paper proposes a plan representation through which the aforementioned problems can be addressed, algorithms that are able to handle this representation and ways to integrate it with both a path execution system (which can be a motion planner) and a task executive. Namely, the following points are addressed:

1. the path planner provides the *frame* in which the motion planner can take its own decisions without impairing the long-range path. Our goal is therefore to stop providing an "educated guess" in the form of a trajectory or a navigation function, but really a set of hard constraints that the motion planner must follow. Of course, if the path cannot be followed, a mechanism must exist that allows for an update (cooperative decision-making).
2. the path planning algorithm acts as a *filter*, producing a symbolic representation of the possible navigation solutions, thus allowing symbolic reasoning on it. More generally, path planning only has to reason on navigation optimization criteria (time, energy), allowing higher level tools to inject other criteria (cooperation possibilities, contingencies).

2 Using corridor graphs as a path representation

This section describes the proposed representation and how this representation helps to address the problems listed in section 1.3: how it can be used to integrate motion planning and then how to integrate task plans. The next section will then present the fundamental algorithms that allow this representation to be built and manipulated.

2.1 Description

The basic idea behind our representation is that, at the level of abstraction where path planning works, searching for an *optimal* trajectory is meaningless: first, especially in outdoor environments, having a very precise map is unpractical for dynamic systems, and therefore the maps the path planner has to work on are imprecise. Second, it is actually expected that the underlying motion planner will seldom follow the generated trajectory, instead using it as a guidance. Our representation is instead based on the idea that the navigation should be *qualitatively* optimal: the goal is not to try following the optimal trajectory, but to represent and structure the regions of space where the robot should progress to be near-optimal. The resulting plan will therefore be both a geometrical representation of this region, but also the necessary information to direct the robot in it.

At the geometrical level, this region is segmented in "corridors": a *median line* and two boundary curves. Each corridor has strictly two end zones, represented by the line between the ends of the median line and the associated ends of the boundary curves (Fig. 2. From a navigation point of view, a corridor represents one step: a robot enters a corridor from one side to exit it at the other side of the same corridor. The topological plan links the corridor sides with two types of edges:

- edges that represent the possible direction of travel inside the corridor itself (intra-corridor edge). Corridors can either be directed or bidirectional. In the first case, there is only one possible direction of travel and one intra-corridor edge. In the second case, there is two edges (one for each direction).
- connections between sides of two different corridors, which represent the possibility to go from one corridor to the other.

While similar to topological maps (see [15] for both an interesting method around topological maps and the related work in that field), it is not a representation of a map, but of a plan: it is a compact representation of the set of corridor sequences (plan executions) that lead to the robot's goal in a near-optimal way.

In this representation, a well-formed *path*, that is a corridor sequence, is an alternance of edges of the two types. In other words, once an inter-corridor edge is taken, the robot must take an intra-corridor edge (i.e. cross the corridor), and vice-versa (i.e. the robot cannot go back into the corridor it just crossed).

2.2 Link with short-range motion planning

In the frame of the path planner/motion planner communication, the corridor plan is no more a graph: a sequence of corridors is already chosen (up to a certain horizon). The motion planner therefore only sees one single corridor, in which the median line can be updated to account for the merged corridor geometry (Fig. 2).

Two methods are envisioned to integrate the underlying motion planner within the frame of our representation:

- using a trajectory-following method. Our representation would not only allow to represent the ideal trajectory in the form of the corridor's median lines, but would

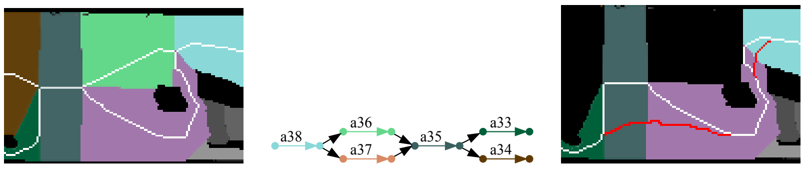


Fig. 2 Detail of a corridor plan which gets around an obstacle. The geometrical representation is on the left, with the obstacle in black and the median lines in white. The topological representation is on the middle: the plan dictates that the robot should come from a38 (blue) to a35 through either a36 or a37. On the right, a specific corridor sequence has been chosen for execution by the motion planner and the median line is updated accordingly (red parts).

also provide the necessary constraints that the trajectory follower should meet in the form of the corridor's boundaries.
- using the path planner to prepare a *planning problem* for the motion planner. For a given robot position, a set of acceptable target configurations can be generated in a given planning horizon (Fig. 3). These configurations would be associations between position, speed and heading, and would be computed based on the corridors shape and a model of the robot's ability to maneuver. Additionally, an optimization constraint would have to be given, as for instance time to travel, or speed maximization.

2.3 Link with task plans

From a symbolic point of view, the nodes of the built graph represent decision points: points where a route should be chosen and others abandonned. Thanks to our representation, the following parameters can be considered in decision-making:

- the probability of failure, which can be derived from various parameters: the minimum corridor width, the terrain type and from the system's confidence in the associated information. For instance, a small obstacle that is not in our map will not have a big impact on wide corridors but would block the navigation in narrow ones. Nonetheless, a narrow passage *that has already been traversed by the system* can be considered safe. Moreover, by considering the cost of getting around said (hypothetical) obstacle, it is possible to take an informed decision about the risk of taking one route or the other.
- given two equivalent paths, the one that has the most options would be chosen, because it is also the one that has more redundancies (if one of its child paths is closed, then there are ways to recover).

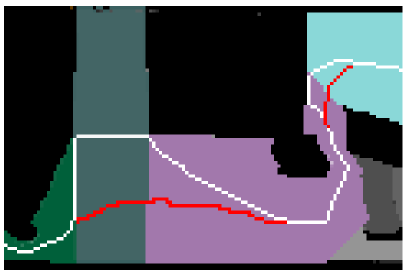


Fig. 3 In this example, the path planner generates a set of target configurations. These sets of allowed $(position, speed, heading)$ values are generated for a certain planning horizon (dotted lines). Black arrows are allowed speeds and headings, red ones are forbidden.

2.4 Fault tolerance

During execution, through the interaction between path and motion planner, the motion planner can suggest an update to the corridor that allows it to overcome an obstacle not foreseen in the path plan. During execution, that “plan repair” would have to be negotiated between path and motion planners. In the current approaches, the motion planner would simply take its own decision based on the information of the navigation function. Because that choice can have effects on the task plan as well, this approach seeks to avoid it. Moreover, thanks to this representation, it is possible to take into account errors in localization: the navigation of a wide corridor would be more robust to navigation errors than the one of a narrow corridor.

Second, the graph-based representation enables the implementation of a contingency planning approach for both path and task plans. As stated before, it is possible to derive the probability that traversing a given corridor fails. What is proposed here is a method to represent the relationship between both the *nominal* plan and the plan where traversing fails, thus allowing decisions to be taken based on an online estimate of the cost of failure. This approach, called contingency planning, is quite common in task planning systems.

3 Implementation

In this section, an algorithm is proposed which can produce a corridor plan based on the analysis of a navigation function. Then, the result of applying this algorithm on a representation of our test track that has been classified by hand (Fig. 4) as a proof of concept. Finally, we will show how a plan update algorithm allows the refinement of an existing plan to either integrate new information or to take into account possible plan faults.

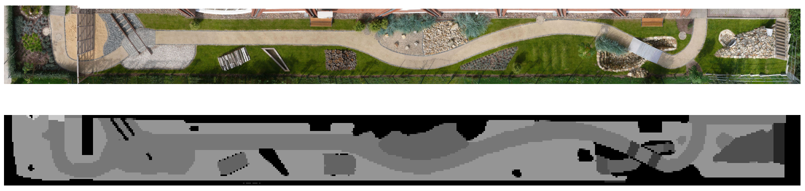


Fig. 4 Up: photo of our outdoor test track at DFKI. Down: the hand-made classification per terrain type. Terrain types are internally associated with average speed values. The cost is time to travel.

3.1 Generating corridor plans

For this algorithm, we assume that the environment is represented as a navigation function sampled on a grid map. The navigation function gives the optimal cost to the goal for each point in the map and can be generated by D* or one of its offspring (we're using D*). The idea of the algorithm is to first find the "navigation region", which includes all points whose *total* cost is within a given margin of the optimal cost, the total cost being defined as the cost for the robot to go from its position to its goal while passing by the considered point. That "cost margin" can be set in different ways, as for instance (i) a fixed value which represents the allowed margin on the overall cost, (ii) a ratio of the optimal cost to goal or (iii) based on the uncertainty of the optimal path (if uncertainty informations is available).

Then, a first step is to segment the generated zone by using a Voronoi skeleton extraction algorithm. Voronoi diagram extraction is a widely studied subject, and we therefore do not describe it here. See for instance [11].

However, the Voronoi diagram only generates an *undirected* graph of corridors. There is still the need to generate a directed graph. The algorithm is as follows:

- for the algorithm to work, the cost along the corridor median lines need to be monotonic – in which case we do the approximation of considering that the whole corridor is monotonic w.r.t. the cost to target. Corridors for which it is not the case are split into two new corridors.
- the corridors are oriented w.r.t. cost to target: the entry point whose cost is higher is the entry point of the corridor and the other one the exit point (remember that navigation is following the down-gradient of the navigation function).
- then, a depth-first search starting at the robot's start point allows to find all the paths that reach the goal in a near-optimal way. The basic idea is to limit the *cost overhead* of any explored path, that is the accumulated cost of traversing a corridor backwards with respect to the orientation determined in the previous step.

The result of this process can be seen on Fig. 5. A detailed explanation, that can help interpreting this plan, is on Fig. 6.

3.2 Merging plans

In this paper, the process of merging plans is a way to represent the links between two given corridor plans: build an unified representation of the two source plans along with the relationship between the source plans and the unified representation, thus showing which parts are different and which parts are common. For our purposes, this has two important uses:

- plan updates needed when the map changes. Along its navigation, the rover will add new information to its map, and therefore the plan will change. What is important from an integration point of view is to be able to determine what part changed, in order to check if there is an impact on the task plan as well.

- contingency planning. When computing a contingency plan, it would be possible to integrate the contingency plan with the nominal plan, allowing the task planners to take these contingencies into account. At the navigation level, it would also allow to know the parts of the plan that are critical because they are part of the nominal *and* contingency plans. In multi-robot cases, these parts could get a higher attention from scouting robots.

As we mentioned before, the object we represent is *not* a topological map but a topological *plan*. Because of that, two corridors are merged if they represent a similar general motion, sharing a geometrical zone is not enough.

The central merge algorithm is a two-corridor merge $twoWayMerge(left, right)$, which is outlined on Alg. 1. This algorithm assumes that the points in the medians are ordered in the direction of travel (i.e. that moving from point i to $i+1$ means progressing along the corridor). The general plan merge is then a recursive application of this algorithm on all pairs of corridors. A detail of two merged plan is presented on Fig. 8.

The two use cases presented above differ by the choice of the $mergedSlice$ function. This function, in the two-corridor merge algorithm, generates a common slice based on two slices that are merged. In the “plan update” case, that would return the right slice as in the end we’re only interested by the new plan. In the case of

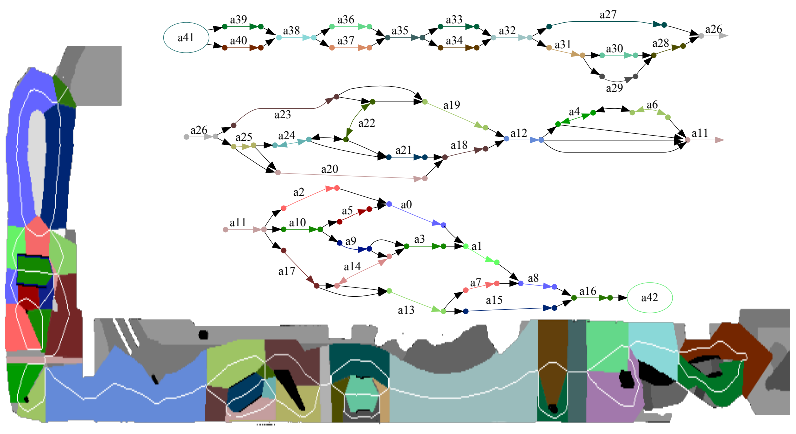


Fig. 5 Generated corridor map on the full terrain, for a navigation starting at the bottom far right (on the line between the green and brown regions). The target point is up-left, in the green region. The resulting topological map is presented on the top. Note that each line continues the previous one (for instance a26 is connected to a28 on the right of the first line and to a25 on the left of the second line). The corridor names have no meaning. Detailed explanation of some complex features is presented on Fig. 6.

contingency planning, this would be an average representation (getting the middle of the median points, and the middle lines for the borders).

Theoretically, one can see in the plan update case that the plan could actually drift in time: the application of updates could slowly change the plan, leading *in fine* to a dramatic change that would be missed by the navigation system. In practice, we do not believe that it would be the case as, from our experience with D^* algorithms, the system mostly moves from one stable structure to another stable structure and will rarely have a continuous change from one to the other. Nonetheless, we're currently working on an extension of this algorithm to do a three-way merge that would fix this problem.

4 Conclusion and future work

The main contribution of this paper is to extend the notion of topological plans and maps to the realm of outdoor, unstructured, environments, thus proposing a novel way to integrate long-range path planning in a complete system which has both an advanced motion planning executive and a high-level reasoning layer. In our opinion, this method and the algorithms we presented here can yield significant improvements in both mono and multi-robot systems, since it allows a more integrated

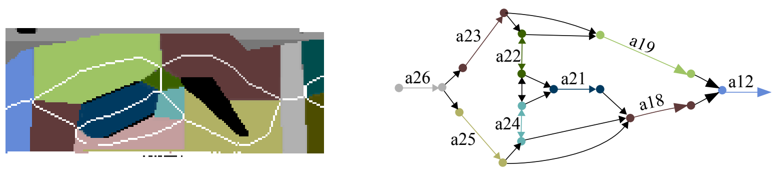


Fig. 6 On this detail of the global plan, the concept of bidirectional corridors can be explained. A robot that comes from a23 has first two choices: either go straight to a19 or to go through a22. If it crosses a22, then it has only two choices: either go through a24 and a18 or a21 and a18. It cannot go back a22 because that plan would be ill-formed (see section 2.1). Symetrically, a robot that comes from a25 can cross both a24 and a22 to go up and reach either a21 or a19. a24 and a22 are therefore bidirectional.

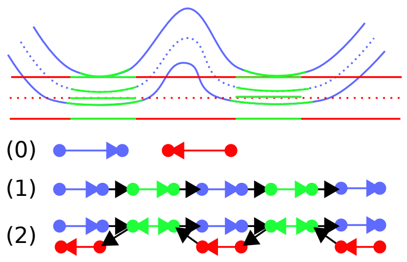


Fig. 7 Simple two-corridor merge. The blue corridor is traversed left to right and the red right to left. (0) represents the original plan, (1) the result of the left pass and (2) the result of the right pass.

approach for decision-making. Indeed, the algorithms presented in this paper enable the use of complex navigation-related information in the symbolic decision-making process.

While we believe that it will provide us a good foundation for autonomous navigation on our own robots, it is not integrated yet in our systems. Future work is therefore, to test in in real conditions. We plan to use various approaches for local path execution, approaches selected based on the navigation context (difficult or "soft" terrains, wide or narrow corridors, ...). On the task planning side, we plan to integrate that approach in a generic plan management component [6], allowing us to, later on, reuse the symbolic algorithms that reason on these corridor plans in conjunction with various task planners.

```
foreach p median point of left do
    if candidate = findBestCandidate(p, right) then
        if last point was not merged then lastCorridor = newCorridor()
        push mergeSlices(p, candidate) in lastCorridor
    else
        if last point was merged then lastCorridor = newCorridor()
        push p in lastCorridor
    end
end
foreach p median point of right do
    if p has been merged in corridor m then
        if last point was not merged then connect(lastCorridor, m)
        lastCorridor = m
    else if candidate = findBestCandidate(p, left) then
        if last point was not merged then
            mergeCorridor = the corridor in which candidate has been merged
            connect(lastCorridor, mergeCorridor)
            lastCorridor = mergeCorridor
        end
    else
        if last point was merged then  lastCorridor = newCorridor() /* create a
            connect to lastCorridor                                          */
    end
end
```

Algorithm 1: Algorithm computing the merge of two corridors. The `connect` function creates a connection from *a* to *b*. In the first pass, this connection is directed if *left* is directed and bidirectional otherwise. Same behavior in the second pass with *right*. Additionally, merge corridors are marked as bidirectional if *left* and *right* don't cross them in the same direction. See Fig. 7 for a simple two-corridor merge

References

1. Ferguson, D., Stentz, A.: Using Interpolation to Improve Path Planning: The Field Dˆ* Algorithm. JOURNAL OF FIELD ROBOTICS **23**(2), 79 (2006)
2. Howard, T., Green, C., Kelly, A.: State space sampling of feasible motions for high performance mobile robot navigation in highly constrained environments. In: Results of the 6th FSR Conference, vol. 42, pp. 585–593. Springer (2008)
3. Howard, T., Kelly, A.: Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. The International Journal of Robotics Research **26**(2), 141 (2007)
4. Ingrand, F., Lacroix, S., Lemai-Chenevier, S., Py, F.: Decisional autonomy of planetary rovers. JOURNAL OF FIELD ROBOTICS **24**(7), 559 (2007)
5. Joyeux, S., Lacroix, S., Alami, R.: A plan manager for multi-robot systems. In: Proceedings of FSR (2007)
6. Joyeux, S., Philippsen, R., Lacroix, S., Alami, R.: A plan manager for multi-robot systems. The International Journal of Robotics Research (2008). Available at roby.rubyforge.org, to be published
7. Lamiraux, F., Bonnafous, D., Lefebvre, O.: Reactive path deformation for nonholonomic mobile robots. IEEE Transactions on Robotics **20**(6), 967–977 (2004)
8. Lamiraux, F., Sekhavat, S., Laumond, J.: Motion planning and control for Hilare pulling a trailer. Robotics and Automation, IEEE Transactions on **15**(4), 640–652 (1999)
9. LaValle, S.: Planning Algorithms. Cambridge University Press (2006)
10. LaValle, S., Kuffner Jr, J.: Randomized Kinodynamic Planning. The International Journal of Robotics Research **20**(5), 378 (2001)
11. Li, C., Chen, J., Li, Z.: A raster-based method for computing Voronoi diagrams of spatial objects using dynamic distance transformation. International Journal of Geographical Information Science **13** (1999)
12. Philippsen, R., Siegwart, R.: Smooth and efficient obstacle avoidance for a tour guide robot. In: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, vol. 1 (2003)

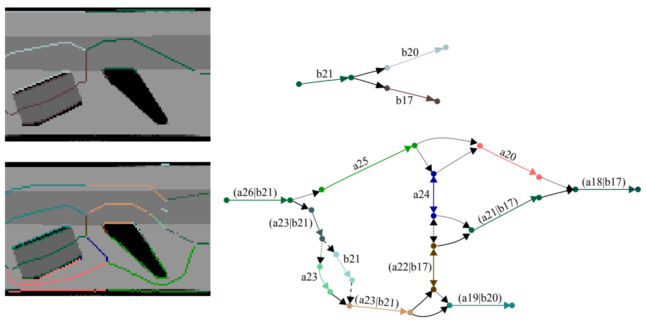


Fig. 8 Example merge showing contingency planning capabilities. In the plan of Fig. 5, a26 has been closed and a new plan generated, this time starting at the entry of that corridor. Then, both plans have been merged. The detail here is the same zone than on Fig. 6. Top: the detail of the plan in the blocked case. Bottom: the merged plan, in which the inter-corridors edges are marked to represent if they are only part of the original plan (dotted), only part of the updated plan (dashed) or part of both (plain). The $(x|y)$ notation shows which corridor of both sides have been merged.

13. Pivtoraiko, M., Kelly, A.: Differentially constrained motion replanning using state lattices with graduated fidelity. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pp. 2611–2616 (2008)
14. Stentz, A.: Optimal and efficient path planning for partially-knownenvironments. In: Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, pp. 3310–3317 (1994)
15. Thrun, S.: Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence **99**(1), 21–71 (1998)

High-level specification and online adaptation of a component layer

Sylvain Joyeux and Frank Kirchner

***Abstract*— Many component-based architectures exist nowadays, to provide the needed flexibility and reconfigurability on robotic systems. Though, very few approaches exist in the robotics community, that tackle the system definition and online adaptation that is the key to creating versatile systems. This paper presents such an approach. Our approach abstract away the complexity of the functional layer configuration in terms of high level services. enabling upper layers to dynamically control this functional layer in terms of high-level requirements instead of detailed component network configuration.**

I. INTRODUCTION

A. Component-Based Architectures in Robotics

Component-based design has been established for quite a long time now as one of the mainstream engineering paradigm to integrate software on robotic systems [1], [2], [3], [4], [5]. The robotic community, by adopting this methodology, acknowledged the progress of the more general field of software engineering, in which component-based system design and, later service oriented architectures has seen a widespread adoption.

Robotic researchers tackled the component-based integration problem by developing many software frameworks to standardize the notion of "components". These frameworks focus (and differences) has always been the definition of the basic services that were perceived as important to integrate robotic software. However, very few contribution exist that allows to easily compose the existing components into a well working system. Indeed, in most frameworks, the "transformation" of a set of components into a working system is manually done by the system designer, with no or little help from software tools.

We feel that this is a very important topic, as the easy *composition* of components is critical to their reuse. One should not have to understand the intrinsics of a component to be able to integrate it into his own system. Ideally, one should have only to require the use of a component in the context of a specific robotic system to see that component integrated in the data and control flow of the robot's functional layer.

Moreover, system-level definition of a component network is also critical to integrate higher-level layers: these upper layers should not have to deal with the specifics of the components, but instead should be able to require high-level services, requirements that are then translated in a dynamic reconfiguration of the functional layer.

Approaches do exist that allow to *verify* that a given system design is valid. The BIP component system [6] allows to model and compose components at the levels of their behaviours, interactions and priorities and verifies that the resulting composition is valid by generating a model-checked controller. The software engineering community also, obviously, concerns itself with the topic of checking that components are compatible at both the *interface* and *protocol* levels [7], [8].

Basu et al. [6] mention that the aim of BIP is to support the complete range of component interaction, from hard real time, synchronous interactions to decoupled asynchronous ones. We argue that, at the level that is our concern – the functional layer of a robotic system – the components fall into the *decoupled* category, making the use of an integration strategy less intrusive than the one used by BIP possible. More specifically, we assume that the following properties hold for the components in the system:

- *validating*: a component should always validate its inputs, to check if they correspond to a set of expected inputs.
- *decoupled outputs*: a component behaviour is not influenced by the fact that its outputs are used or not, and by whom they are used. It is possible, that the module will *not* generate a certain data structure if that data is not used (i.e. outputs not conneced). What we assume is that one output not being connected does not change the value of the other outputs, everything else being the same.
- *"good" behaviour is independent of data availability*: in case the component lacks data on one of its inputs, will either have a sensible default behaviour or will go into a well-defined fault mode. In no case the module should silently ill-behave.

B. Our Approach

Our approach has the following design criteria:

1) easy and flexible specification of a component network. The root idea is that the "user" of the functional layer should only have to require high-level services, our system being able to configure the network to realize these services.
2) the generated system model should be usable for supervision. In other words, it should provide enough dependency information to assess whether the functional layer is in a good state or not. As we will see later, this requires that composition is done *with sharing*: the same components can be shared by different services, leading to a composition graph which is not a tree as it is common in composition models.

DFKI Robotics Innovation Center, 28359 Bremen, Germany
firstname.lastname@dfki.de

3) reusability of the models. Once models are created for a set of components, it should not have to be modified to fit a specific robotic systems.
4) dynamic reconfigurability of the component network, to fit changing requirements from the "user" of the functional layer.
5) static verification: it is impractical to always have to use the platform to check that the system networks will function as expected. Instead, we argue that it is more practical to know the interfaces of the components statically, and thus being able to verify offline that the compositions (i) are valid and (ii) that it is possible to dynamically switch between them.[1].

Two approaches from the software engineering community are related to ours. In the Fractal component model, Bruneton et al. [9] tackle the problem of composition with sharing. While being very related to Fractal, our approach extends it by using a type system more extensively that provides a more powerful system building algorithm and dynamic reconfiguration. In [**?**], XXXX et al. propose a composition model in which compositions are seen as *patterns*, i.e. abstract networks, that are used to instantiate the overall system network. We also extend this approach by extending their type system, in order to support an easy to use system specification and model reusability, as well as having dynamic network reconfiguration.

The remaining of this paper has the following structure. Section II will present the underlying component implementation, our system model and how the combination of the two allows to create a system specification. Then, in section III, the core algorithm will be described, that allows to instantiate this system specification into a working component network as well as reconfigure it online. Section IV will be a wrap-up of our solution, as well as an outlook on possible extensions of the approach.

II. SYSTEM-INDEPENDENT MODELLING OF A COMPONENT LAYER

As just mentioned, one of the requirements of this approach is that the models on which we are based should be system-independent. To achieve this, we abstract away the component-based implementation (section II-A) into a service-oriented description (section II-B) (see the introduction of [10] for a nice parallel between component-oriented and service-oriented architectures).

It is then possible to specify the building blocks of the component networks in a mixture of concrete components and abstract services (section II-D) and finally associate these abstractions to the actual concrete implementations that are available on the system to obtain a system definition (II-E).

A. The RTT Component Model

The Orocos RealTime Toolkit [11], [3] is a C++ component implementation. Its main originality does not lie in its

[1]this does *not* preclude the existence of dynamic creation of inputs and/or outputs on the components. Our approach simply assumes that the norm are static I/O and that dynamic port creation is the exception

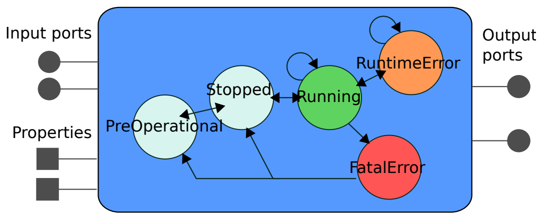


Fig. 1. Overview of a RTT component. Input and output *ports* are the interfaces for data flow I/O. *Properties* form the configuration interface. A method call interface also exists, but is not depicted here as it is not relevant for the concerns of this paper. External software can control the module's state machine.

interface, which is by now a quite standard interface (Fig. 1), but by these three things:

- the functionality is integrated in a standard state machine, which allows external supervision (or coordination) components to control it. To our knowledge, this is also only seen in the GenoM modular framework [2].
- the task context, as components are called in the RTT, are not tied, at design time, to a particular thread and/or triggering mechanisms. That triggering mechanism (which is called an *activity*) is chosen at *deployment time*. Ergo, it is possible to integrate the same functionality in completely different system dynamics *and* the dynamics of the component is known to the external world. As we will see, this last part is important for automated configuration.
- the RTT itself is not tied to a particular communication strategy: it is chosen at runtime, to fit the needs of the controlled system. In particular, the RTT offers by default two communication policies. The *data* policy ensures that the reading component will always get the most up-to-date sample. The *buffer* policy, by contrast, is a fixed-size FIFO. The fixed size is important for decoupling reasons, as one cannot know if a dynamic-sized FIFO will actually grow forever (if the input generates data faster than the output can process).

To ease the development of RTT components, we developed the oroGen component specification and code generation tool [12]. This tool, greatly influenced by the GenoM tool [2], provides the specification language that our approach will use. Based on this specification, it generates the core of the component, which *(i)* makes the development of RTT components much easier and *(ii)* ensures that the component's implementation actually matches its specification.

B. Abstract Service Models

In our approach, a *service* is either a *data source*, that injects data into the system (usually coming from hardware), or a *data processor*. In both cases, we characterize the service by its abstract type, its inputs and its outputs. A service is therefore not solely defined by its inputs. Indeed, for instance for a data processor, the type of processing that is made does matter. In the same way, the same data type can be used to

```
interface 'Position' do
  output_port 'position_samples',
    '/wrappers/samples/RigidBodyState'
end
interface 'Orientation' do
  output_port 'orientation_samples',
    '/wrappers/samples/RigidBodyState'
end
interface 'Pose' do
  output_port 'pose_samples',
    '/wrappers/samples/RigidBodyState'
  end
  provides Position, Orientation
end
```

```
class Dgps::Task
  driver_for 'Mb500', :provides => Position
end
```

Fig. 2. **Top:** abstract definition of positioning services in our system. The syntax is the one of the Ruby programming language, as we use it for our implementation. Since most communication infrastructures do not support type slicing, we use the same data type to represent pure positions, orientations and complete poses. The `Pose` service is then declared as providing both `Orientation` and `Position`, the necessary port name mapping begin handled automatically since there is no ambiguity. **Bottom:** Declaration of a device driver for a MB500 differential GPS board from Magellan. This declares that the driver provides a `Position` service. This driver definition implicitly defines a *device type*, that can be used in the robot definition file to enumerate the robot's devices (see Fig. 6).

represent different informations, and therefore the data type alone is not enough to characterize a data source service.

To cover these aspects, a type system is used to describe these services and their relations to concrete components. A *service* type is therefore a triple (T, P, I, O) where T is the service type, $Parents$ an optional set of service types that T implements, $I = [(Name, DataType), \cdots]$ its set of inputs and $O = [(Name, DataType), \cdots]$ its set of outputs. The obvious constraint is that $\forall Parent \in Parents, \quad I_{Type} \subset I_{Parent}$ (idem for outputs).

As an example, the positioning-related interfaces in our system are defined on Fig. 2.

Dynamic creation of inputs and outputs is an extension of this service definition: among the I and O sets, some ports are definition of *dynamics ports*: they are not a $(Name, DataType)$ pair anymore, but a $(NamePattern, DataType)$. They are used in the following way: during deployment, if a connection is requested and the associated port does not exist, dynamic ports are checked for matching definitions and a *dynamic port instance* is created. It is then assumed that the underlying component, at runtime, will be configured so that it creates the requested port. These dynamic ports are most useful when integrating multiplexing/demultiplexing components, as for instance drivers for hardware busses (in our system, a CAN bus).

C. Services and Components

In our implementation, the component models are loaded directly from the oroGen specification files, and are mapped onto Ruby classes (as our implementation uses Ruby as its interface language). For instance, a `Task` component model defined in a `dgps` oroGen project will be mapped onto a `Dgps::Task` class on the Ruby side.

They can then be extended by declaring what services they provide. This declaration will allow, at deployment time, to determine what concrete component can be used in what context. Moreover, instead of adding a service to a component, one can declare that components are *data sources*. On a robot, these would be device drivers, when offline, they would be log replay components. These data sources are submodels of generic services, as for instance a device driver for a GPS that will – trivially – be a submodel of the `Position` service we presented (Fig. 2).

More complex cases are also possible: some components may provide the same service multiple times. In this case, services should be given an unique name that can be used at deployment time to refer to a specific service. For instance, a stereocamera would have both a `PointCloud` service and two `Image` services that could be named “left” and “right”.

D. Composition Models

Representing aggregations of components is a classical way to hierarchically build a more complex system from the atomic building blocks that the components are. We adopt this methodology, but extend it to match our type system.

More specifically, in our framework, a *composition* is a set interfaces, concrete components or compositions, with the associated connections. Since the composition is meant to represent a composite service as well, it can be declared to export ports from components inside the composition to the outside, and thus provide services. Finally, as for services, a composition can derive from parent models in order to provide a composition generalization mechanism (Fig. 3 and 4).

Compositions feature an automatic connection process, in which case the connections are created *within the boundaries of the composition itself.* This way, one avoids the main issue of system-wide automatic connection schemes: on large systems, it becomes hard to be certain that undesired connections will actually not be added and/or desired connections will be added in the final network.

E. Putting Everything Together: System Specification

In our scheme, building a *system specification* requires three sets of information:

1) what data sources are available on the system. This is normally provided by a robot description file, which maps names to data source models. Offline, it can also be generated by mapping data streams in log files onto services.
2) what is required. The general syntax is add(Model[, :as => 'name']), where `Model` can be a service model, a composition or a concrete component.
3) disambiguation information if multiple solutions exist to map the abstract requirements of (2) to concrete

```
composition 'ControlLoop' do
  abstract
  add Hbridge::Task
  add Controller

  specialize Controller, MotionController,
    :not => FourWheelController do |ctrl|
    export ctrl.motion_command
    provides MotionController
  end
  # also specialize Controller on FourWheelController

  default_specialization 'Control',
    MotionController
  autoconnect
end
```

Fig. 3. Control loop composition model. This definition uses concrete motor drivers directly (`Hbridge::Task`), as there is only one existing for now. However, there are two possible command types. First, a classical *motion command* which provides longitudinal and rotational speeds. Second, a method aimed at controller development: the *four wheel command* which controls the four wheels separately.
The `specialize` statement creates a submodel of ControlLoop that will be selected at deployment time, based on whether the Controller child provides the MotionController or the FourWheelController services (FourWheelController specialization not detailed). Moreover, specialization where Controller implements both services is not generated (`not` specifications). Moreover, if a ControlLoop element accepts both motion and four wheel commands, then the motion specialization should be used (`default_specialization`). Finally, only specializations of this composition can be used in practice (`abstract` statement)

```
composition 'ManualDriving' do
  abstract
  add Compositions::ControlLoop, :as => 'Loop'

  specialize 'Control', MotionController,
      :not => FourWheelController do
    add MotionCommand
    autoconnect
  end
  # also specialize Control on FourWheelController
end
```

Fig. 4. Composition that ties a ControlLoop composition (Fig. 3) with an actual command source (either a joystick for motion commands or a self-made sliderbox for the four wheel commands).

components. This information is a mapping from either a composition's child name or a service type to a concrete component model or a data source name. During the composition instantiation, this selection will be applied to any matching composition child, replacing the abstract service requirement by a more concrete element.

As an example, let's consider the deployment of a `ManualDriving` composition, as defined on Fig. 4. On our system, we have three different motor controllers, that are adapted to different types of environments, and two different types of input commands. If one would try to build a manual driving service without additional information, there would be an ambiguity on the type of controller to use.

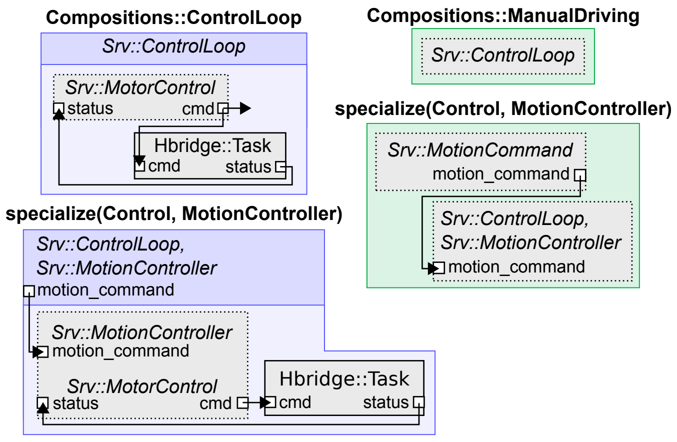


Fig. 5. Graphical representation of the compositions of Fig. 3 and 4

```
Robot.devices do
  device Mb500
  device Joystick
  device Sliderbox
  device Motors
end

add(ManualDriving).
  use('Loop.Controller'
   => PIVController)
```

Instantiate(Compositions::ManualDriving)
MotorControl elements matches selection
`use('Loop.Controller' => Control::PIVController)`

Instantiate(Compositions::ControlLoop)
`use('Controller'=>PIVController)`
default_specialization applies
→ MotionController specialization
select specialization based on the
Loop instance
→ MotionController specialization

Task Allocation
Need a Srv::MotionCommand data source
→ Joystick (the only one available)

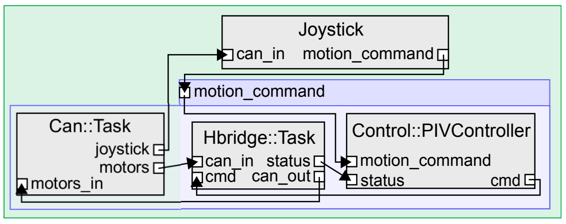


Fig. 6. Composition instantiation. Left: robot definition and system requirement. Right: the corresponding trace of the instantiation algorithm and resulting network. The colors represent the composition type (Fig. 5). The integration of the CAN driver requires a definition in the robot file that says which devices are on which busses, but no modification of the compositions. Its handling during instantiation is trivial and is therefore not presented in this paper. In the final structure, the CAN task is being depended-upon by both compositions, as it is needed by the Hbridge and Joystick components.

Moreover, one needs to adapt the input chain (in the case of ManualDriving, the input device) to the actual inputs the controller accepts. This is the role of the specializations and of the explicit child selection (Fig. 6).

III. INITIAL DEPLOYMENT AND RUNTIME ADAPTATION

A. System Deployment

The root idea of our deployment method is that the behaviour of a data processor is determined by what inputs it will have, and that the behaviour of a data source service is determined by the actual data source it is associated to (e.g. the underlying sensor). By applying this assumption recursively, two components of the same type that are linked to the same set of data sources in the same way will have the same outputs, and it is therefore possible to exchange them.

The core of our algorithm is therefore a `can_merge?(a, b)` predicate, which is true if and only if

- `a` is a supermodel of `b`. If `b` is a set of services, it mean that `a` must provide these services. If `b` is a concrete component or a composition, then `a` must be an instance of a submodel of this component or composition.
- the configuration parameters of `a` are either the same, or are not set on `a` for parameters that are set on `b`
- `a` has the same inputs than `b`, or has unconnected inputs where `b` is connected

If `can_merge?(a, b)` is true, then the `merge(a, b)` operation replaces `b` by `a` by (i) updating its configuration parameters and (ii) adding the missing input and output connections.

We can now outline the system deployment algorithm:

(1) instantiate the required composition and components (from the `add` statements), applying the disambiguation information. Also instantiate all data sources that are declared in the robot declaration file.
(2) allocate the remaining abstract services For each abstract service S, the algorithm finds all existing compositions and concrete service x for which `can_merge?(x, s)` holds. Since explicit selection (`use` statements) have already applied, any remaining ambiguity is interpreted as an error.
(3) recursively merge elements where `can_merge?(a, b)` is true. What this step does is recognize the parts of the network that are equivalent, and bring them back together. We will see that this step is actually the key step of the algorithm, as it allows dynamic reconfiguration as well.
(4) allocate deployed tasks Orocos deployments are the operating system processes in which C++ component classes are actually instantiated, associated with triggering mechanisms and allocated a thread. This stage of the algorithm actually takes the component network and maps it to the actually running components by running (3) again.

The system deployment failed if (i) there are abstract services left or (ii) if a task has no supporting deployment.

B. Dynamic Data Flow Modelling

The Orocos/RTT framework requires that, when a connection is created, a *connection policy* is chosen (section II-A). Our algorithm computes these policies based on the following information:

- the *activity* information, i.e. how each components are triggered (by the underlying device, because of its inputs, at a fixed period), and what is the expected maximum latency between the trigger and the corresponding component execution. This information is part of the component's own deployment.
- whether the component expects certain inputs to use reliable connections (no-loss) or not.
- some details on the communication that will pass on the connection (how much data samples are sent per cycle, is there "bursts", i.e. times where more samples are sent than usual)
- detailed data source information (device period, how much samples per period, ...), given in the robot definition file.

Based on this information, the algorithm picks a buffer policy if the connection needs to be reliable and a data policy otherwise. The buffer size is calculated so that it can accommodate the samples that the source will push on the connection until the next execution cycle of the target component. This information is then propagated in the data flow network until it is computed for all components.

C. Online Management

Our implementation is based on the Roby plan manager [13]. Roby, in our context provides two important tools.

The first tool is a task model to represent and command the component's execution, and a dependency model which allows to supervise the good execution of the component network [14]. Moreover, Roby allows us to represent both the components and the underlying OS processes in which the components run (Orocos deployments), thus allowing to react when a process crashes.

The second tool is the *transaction*. This is a sandbox that allows to modify the plan while it is running. The idea behind transactions in Roby is that neither the running plan should not be modified little by little, nor it is desirable to block the main supervision thread while computing a plan update. Transactions therefore keep track of the modifications that exist between the plan that is being executed and the plan that is being generated. In our case, it means that system adaptation is first computed in a transaction, and is applied only once the algorithm finished.

In order to be able to manage data flow modifications at runtime, extensions to the Roby base model have been implemented. First, a DataFlow task relation represents the required connection. In this graph, connections that go through composition boundaries (as for instance the `motion_command` chain on Fig. 6) are represented as multiple edges. This relation graph is translated into a RequiredDataFlow graph, in which composition ports are removed – i.e. only concrete connections are stored. Finally, an ActualDataFlow graph stores the current connection state of the running components. The needed modifications to the underlying component connections is therefore the difference between the ActualDataFlow and the RequiredDataFlow graphs.

The runtime update process ensures that the component network is never left in a half-configured state, i.e. a state where missing connections and/or non-running components makes the whole system non-functional. To do so, either all required state and connection modifications are applied at once, or none are.

```
(1) add(PoseEstimator).
        use('imu' => XsensIMU)
(2) add(TrajectoryFollowing).
        use('Control.Controller' => PIVController)
```

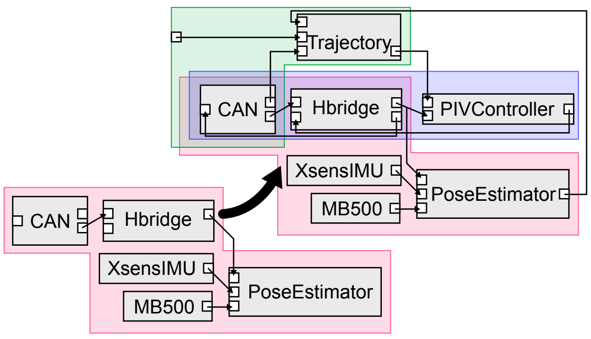


Fig. 7. Transition between two driving modes. **Top:** (1) original requirement and (2) applied modification. **Bottom-left:** the starting point is a low-level driving mode, where the motor hardware is directly controlled through a microcontroller, with navigation aid (position estimation). **Top-right:** the target configuration is an autonomous trajectory following mode (top right). The transition requires the hbridge component to be restarted, so that it can take control of the motor hardware over the embedded microcontroller. The component will go into read-only mode (only the state reading for odometry estimation) if its command input port is not connected, and to read-write mode otherwise. This restart step is handled by our plan manager, which detects it because the command input port of the hbridge component requires static connections. Both CAN and Hbridge components are shared in the final composition, as the hbridge is needed for odometry measurement and motor control.

D. Online Deployment Modification

By deployment modification, it is meant that the user can require some of the serives to be removed, add new services and/or require that some already running services are replaced by another one (in our earlier examples, replacing a manual driving service by a trajectory following strategy for instance).

Basically, the instantiation algorithm works in two steps: *(i)* the network that implements the required functionality is produced (steps 1-3) and then find a mapping from the resulting network to the current setup of the network (step 4). Trivially, it can therefore be extended to adapt running systems by completely disconnecting services that are being removed or replaced before (3) is run.

Indeed, these components will be free to be reused by the merging algorithm in step (3). Then, thanks to the use of our plan manager's transactions, the resulting component network will be compared to the currently running one when the transaction is applied and only the differences will be applied. On our system, this scheme allows us to switch successfully between any combination of motor controller and navigation modality (example on Fig. 7)

IV. CONCLUSIONS AND FUTURE WORKS

This paper presented a scheme for the system modelling, deployment and online adaptation of a component-based functional layer. The main contributions of that scheme is its capacity of abstraction and reusability. *Capacity of abstraction* since the scheme handles all the non-ambiguous cases automatically, requiring explicit selection only in cases where ambiguities exist. *Reusability* since it neatly separates the required specification of the components and compositions from the specifics of a system.

In practice, thanks to this scheme, deploying a component library on a system boils down to (i) describing the devices that exist on this system and (ii) asking for a set of high-level services, while specifying what device and/or specific method to use (for instance: "monocular SLAM service using the front-left camera and the front IMU" on a two-camera, two-IMU system).

One main issue needs to be tackled, though. The deployment algorithm we describe, while sufficient in the non-trivial case of our robotic system, cannot handle the general case of having loops that involve only non-data sources. The reason is that the algorithm is iterative and not global, and therefore these loops would not be recognized as being identical. A global graph matching algorithm would be needed to solve the general case.

Regardless of this flaw, this scheme provides us with a robust base on which to build: it is now in use on the Asguard robot at DFKI, and provides us with the flexibility required to reconfigure the component network. Through the integration in our plan manager, we believe that it will enable us to use online reasoning to drive adaptation and this way solve more difficult problems – in other words *choosing the right tool for the right job.*

V. ACKNOWLEDGMENTS

This work has been done in the frame of the Intelligent Mobility project, funded by the German DLR, with funds from the Ministry of Science (BMBF). Grant Nr. 50RA0907.

REFERENCES

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *The International Journal of Robotics Research*, vol. 17, no. 4, p. 315, 1998.

[2] S. Fleury, M. Herrb, and R. Chatila, "Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1997, pp. 842–848.

[3] P. Soetens, "A Software Framework for Real-Time and Distributed Robot and Machine Control," PhD, Katholieke Universiteit Leuven, May 2006.

[4] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The CLARAty architecture for robotic autonomy," in *IEEE Aerospace Conference*. Ieee, 2001, pp. 1/121–1/132.

[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Open-Source Software workshop of ICRA*, 2009.

[6] A. Basu, M. Bozga, and J. Sifakis, "Modeling Heterogeneous Real-time Components in BIP," in *Proceedings of the IEEE International Conference on Software Engineering and Formal Methods*, 2006.

[7] D. Giannakopoulou, G. Leavens, and M. Sitaraman, "SAVCBS 2001 Proceedings," in *Specification and Verification of Component-Based Systems - Workshop at OOPSLA 2001*, 2001.

[8] A. Speck, E. Pulvermuller, M. Jerger, and B. Franczyk, "Component composition validation," *International Journal of Applied Mathematics and Computer Science*, vol. 12, no. 4, p. 581590, 2002.

[9] E. Bruneton, T. Coupaye, and J. Stefani, “Recursive and dynamic software composition with sharing,” in *Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming (WCOP02)*, 2002, pp. 1–8.

[10] A. W. Brown and S. Johnston, “Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications.”

[11] “the Orocos project,” www.orocos.org.

[12] “the RubyInMotion project,” http://sites.google.com/site/rubyinmotion.

[13] S. Joyeux, R. Alami, S. Lacroix, and R. Philippsen, “A Plan Manager for Multi-robot Systems,” *The International Journal of Robotics Research*, 2008.

[14] S. Joyeux, S. Lacroix, and R. Alami, “A Software Component for Simultaneous Plan Execution and Adaptation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

Literaturverzeichnis

Die Verwendete Fachliteratur ist in den jeweilig angehängten Dokumenten (siehe Sektion 5) zu den einzelnen Arbeitspaketen zu finden.

[illegible]