Context-aware design and motion planning for autonomous service robots

Context-aware design and motion planning for autonomous service robots

Janno Lunenburg

The research presented in this thesis is part of the research programme of the Dutch Institute of Systems and Control (DISC). The author has successfully completed the educational program of the Graduate School DISC.

The research leading to these results is part of the R5-COP project, funded by ARTEMIS and the Bobbie project, funded by NL Agency (Dutch Ministry of Economic Affairs)

A catalogue record is available from the Eindhoven University of Technology library
ISBN: 978-90-386-3870-6

Typeset by the author with the pdfLaTeX documentation system
Cover design: Janno Lunenburg
Reproduction: Ipskamp Drukkers B.V., Enschede, the Netherlands

Context-aware design and motion planning for autonomous service robots

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. F.P.T. Baaijens
voor een commissie aangewezen door het College voor Promoties,
in het openbaar te verdedigen
op woensdag 24 juni 2015 om 16.00 uur

door

Janno Johan Maria Lunenburg

geboren te Veghel

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. L.P.H. de Goey	
promotor:	prof.dr.ir. M. Steinbuch	
copromotor:	dr.ir. M.J.G. van de Molengraft	
leden:	prof.dr. H. Nijmeijer	
	prof.dr. H. Bruyninckx	(KU Leuven)
	prof.dr.ir. E.J. van Henten	(Wageningen University)
	prof.dr.ir. S. Stramigioli	(Universiteit Twente)
	dr.ir. M. Wisse	(Technische Universiteit Delft)

Societal summary

As a result of the aging population, the need for care is increasing while the amount of home care and nursing staff available to provide this care is decreasing. Developing domestic service robots is a possible solution to anticipate this shortage. However, there is still a long way to go before service robots can be part of our daily lives. Both the hardware and the software are not yet at the level that robots can perform a variety of tasks as quickly, reliably, flexibly and robustly as humans do with costs comparable to or even lower than care personnel.

This thesis introduces a new domestic service robot. This is an open hardware design: researchers and developers are free to copy and improve this design. Similar to open source software, this should eventually lead to better robot hardware and lower costs.

Furthermore, a number of advances in motion planning for mobile robots are introduced. These will allow future service robots to move quickly and safely through a domestic or care environment.

The hardware as well as the motion planning developments contribute to the introduction of domestic service robots in our daily lives.

Summary

Context-aware design and motion planning for autonomous service robots

As a result of the aging population, the need for care is increasing while the amount of home care and nursing staff available to provide this care is decreasing. Developing domestic service robots is a possible solution to anticipate this shortage. However, there is still a long way to go before service robots can be part of our daily lives. Both the hardware and the software are not yet at the level that robots can perform a variety of tasks as quickly, reliably, flexibly and robustly as humans do with costs comparable to or even lower than care personnel. There are two important steps to be made: i) make robust, reliable and capable service robots available to a large community of researchers and developers and ii) develop skills for these robots, of which the ability to quickly and safely plan motions through a domestic or care environment is one of the most important ones. This thesis discusses four aspects related to hardware design and motion planning.

To make service robots available to a large community, a new modular, open hardware domestic service robot is developed that has a fully holonomic platform that can robustly cope with indoor surfaces and an upper body that combines the vertical range and stability of a linear vertical actuator with the dexterity of a rotational joint. Furthermore, the robot has a modular setup with clearly defined mechanical and electrical interfaces.

The second main topic of this thesis is motion planning. Over the past fifty years, the amount of literature discussing motion planning has grown so large that it has become extremely difficult to select a combination of representations and search algorithms that are suitable for a particular application. To aid this selection process, a methodology is developed that discusses the relevant criteria to select appropriate environment representation classes, search algorithm classes and a suitable approach to combine these into a navigation system.

Existing navigation systems for domestic service robots usually suffer from the trade-off between being quick and being robust against changes in the environment. To improve this behavior, an existing three-dimensional environment representation that probabilistically fuses sensor measurements to represent the occupancy probability of volumes is extended with a time-dependent occupancy probability model. Additionally, a proactive approach is used to deal with unpredictably moving obstacles. This navigation system is extensively tested in simulations, laboratory experiments and real life experiments.

Finally, it is demonstrated how a single, centralized, object-oriented world model can be used for multiple modules, *e.g.*, task planning and execution, localization, navigation and manipulation, in the software stack of a service robot instead of having separate world models for each module. For the specific use-case of navigation, it is shown how this approach increases robustness and safety: i) while using this object-oriented world model as a basis for the planning representation, ii) while using task context to determine the goal regions with respect to world model objects and iii) while using symbolic knowledge of the task and goal to coordinate the motion planners.

Samenvatting

Als gevolg van de vergrijzing van de bevolking is er een toenemende behoefte aan zorg, terwijl de beschikbare hoeveelheid zorgpersoneel afneemt. Het ontwikkelen van zorgrobots is één van de mogelijkheden om te anticiperen op het toekomstige tekort aan zorgpersoneel. Er is echter nog een lange weg te gaan voordat robots deel kunnen uitmaken van ons dagelijks leven. Zowel de hardware als de software van robots is nog niet op het niveau dat benodigd is om een verscheidenheid aan taken net zo snel, betrouwbaar, flexibel en robuust uit te voeren als mensen dit kunnen. Tevens zijn de kosten voor zorg- of huishoudrobots momenteel nog veel te hoog. De ontwikkeling van robuuste, betrouwbare en capabele robots die op grotere schaal beschikbaar zijn is een eerste stap om deze robots in ons dagelijks leven te introduceren. Vervolgens is één van de belangrijkste vaardigheden van zorgrobots het vermogen om snel en veilig bewegingen te plannen in een huis- of zorgomgeving. In dit proefschrift komen vier onderwerpen aan bod die gerelateerd zijn aan het ontwerpen van robot hardware en het plannen van bewegingen.

Op de eerste plaats is een nieuwe modulaire, open hardware huishoudrobot ontwikkeld met een volledig holonome basis die robuust in een binnenomgeving rond kan rijden en een bovenlichaam dat het verticale bereik en de stabiliteit van een lineaire verticale actuator combineert met de behendigheid van een draaigewricht. Tevens is de robot modulair opgebouwd met goed gedefinieerde mechanische en elektrische interfaces.

Het tweede hoofdonderwerp van dit proefschrift is het plannen van bewegingen. In de afgelopen decennia is er dusdanig veel literatuur over het plannen van bewegingen geschreven dat het erg moeilijk is geworden om een combinatie van omgevingsrepresentaties en zoekalgoritmes te vinden die geschikt is voor een specifiek probleem. Om dit selectieproces te ondersteunen is een methodologie ontwikkeld die de relevante criteria beschouwt om de juiste klasses van omgevingsrepresentaties en zoekalgoritmes te selecteren alsmede een plan van aanpak om deze te combineren.

Bestaande navigatiesystemen voor huishoudrobots moeten vaak een afweging maken tussen tussen snel bewegen en robuust zijn tegen veranderingen in de omgeving. Om dit gedrag te verbeteren is een bestaande drie-dimensionale omgevingsrepresentatie die probabilistisch metingen integreert uitgebreid met een tijdsafhankelijk bezettingskansmodel. Tevens wordt een proactieve aanpak toegepast zodat de robot om kan gaan met onvoorspelbaar bewegende obstakels. Het gedrag van dit navigatiesysteem is uitgebreid getest in simulaties, experimenten in een testlaboratorium en experimenten in de universiteitsbibliotheek.

Ten slotte is aangetoond hoe een enkel, centraal, object-georiënteerd wereldmodel kan worden gebruikt voor verschillende software modules, bijvoorbeeld taakplanning en -uitvoering, lokalisatie, navigatie en manipulatie in plaats van een specifiek wereldmodel voor iedere taak. Met navigatie als specifieke toepassing is aangetoond dat deze aanpak de robuustheid en veiligheid verbetert: i) door dit object-georiënteerde wereldmodel te gebruiken als basis voor de planningrepresentatie, ii) om taak-context te gebruiken om doelregio's ten opzichte van objecten in het wereldmodel te berekenen en iii) symbolische kennis van de taak en het doel te gebruiken om de planners te coördineren.

Contents

Chapter 1

Introduction

1.1 The need for service robots

The worldwide ageing population is putting increasing pressure on the home care system. While the number of elderly people in need of care is increasing, the number of personnel available to provide this care is decreasing. In the Netherlands, the ratio between the amount of people above the retirement age and the potential labor force is expected to increase from 0.27 in 2012 to 0.39 in 2040 (van Duin and Stoeldraijer, 2013). This trend is visible in almost all countries in the European Union (Robustillo et al, 2013) as well as in Japan, Canada, Australia, New Zealand and the United states (Anderson and Hussey, 2000). As a result, there will be a shortage of assistance in daily activities such as cleaning the house, preparing and serving food and doing the laundry (Bugmann and Copleston, 2011). This will make it potentially difficult for the elderly to live independently at home.

Robots are a possible solution to this shortage. Hereto, these robots should be able to robustly perform a variety of tasks in a domestic or a care environment. Furthermore, the costs for such a robot should at most be equal to the costs of care personnel and preferably even lower.

Ever since Unimate joined a General Motors assembly line in 1961, robots have been used in industry to automate production processes such as welding (Figure 1.1a), painting (Figure 1.1b) and pick-and-place tasks (Figure 1.1c). In IFR (2012), it was estimated that at the end of 2011, between 1.2 and 1.4 million industrial robots are being used throughout the world. These robots can perform repetitive tasks quickly, accurately and reliably and therefore do jobs that are, either too heavy, too dangerous or too boring for humans. Due to the repetitive nature of their tasks and due to the invariance of their environments, it is possible to pre-program exact trajectories for each joint of the robot for one specific task. As a result, industrial robots are not able to react to changes in their task or in the environment: even slight changes in the task or the environment requires reprogramming of the robot, which is often a costly and time-consuming task.

More recently, robots have found their way to domestic environments, examples of which can be found in Figure 1.2. Like industrial robots, these have been programmed for a specific purpose, *e.g.*, vacuum cleaning, lawn mowing or pool cleaning. The big difference, however, is that they do not operate in a static environment and hence, their trajectories are not pre-programmed. Often, these robots simply drive randomly in their designated area, under the assumption that they will eventually cover the entire environment this way. Since these robots are not aware of their environment, this approach results in suboptimal trajectories, *e.g.*, robots have cleaned a certain part of the floor many times while another part is still dirty, and robots

(a) A welding robot.

(b) A painting robot.

(c) A pick-and-place robot.

Figure 1.1: Three examples of industrial robots.

can easily get stuck in unforeseen situations.

Therefore, to really support people in their activities of daily living, a whole new type of robot is required: service robots that are not designed for a specific purpose but are designed –both in hardware and software– to perform a wide variety of tasks in an arbitrary domestic environment. Such robots contain a moving base platform to move through the environment and one or more manipulators to perform manipulation tasks. Furthermore, these robots need various sensors to perceive their environment. Examples of such robots can be seen in Figure 1.3.

Unlike industrial robots, it is not possible to pre-program all motions due to the diversity in tasks and environments. On the other hand, the robots cannot complete their tasks by simply executing random motions such as vacuum cleaners. Therefore, it is essential that service robots plan their motions based on the task at hand and the environment in which they operate.

1.2 Putting service robots into practice

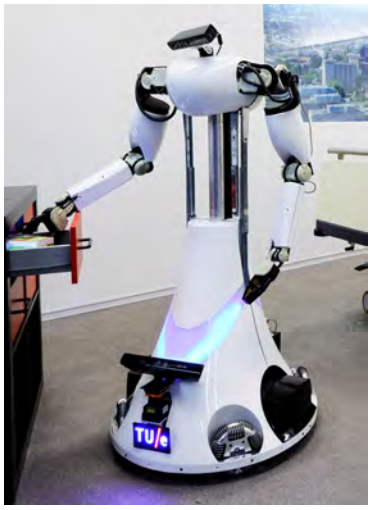
There is still a long way to go before service robots will be part of our daily lives. Significant improvements on both the hardware as well as the software are required before robots can perform a variety of tasks as quickly, reliably, flexibly and robustly as humans do with costs comparable to or even lower than the costs of care personnel. Large-scale availability and the ability to navigate through their environment quickly as well as safely are key prerequisites to put service robots into practice. Therefore, the main problem definition of this thesis is defined as:

(a) A vacuum cleaner robot.

(b) A lawn mower robot.

(c) A pool cleaner robot.

Figure 1.2: Three examples of domestic robots.



(a) The AMIGO robot (Photo courtesy of Bart van Overbeeke Fotografie.)

(b) The PR2 robot.

(c) The Care-O-bot 3 robot.

Figure 1.3: Three examples of domestic service robots.

Design a modular, affordable service robot and develop a navigation system that is robust against the dynamics and uncertainties of the environments a service robot operates in.

In this thesis, four highly relevant issues in robot design and motion planning will be addressed:

- fully functional service robots are not available on a large scale for developers. Many research groups use custom robots that are not available to other researchers and the robots that can be purchased are either too expensive or not adequate.
- despite the large amount of literature on motion planning, it is extremely difficult to select a combination of representations and search algorithms for a specific motion planning application.
- navigation systems of service robots are either too slow or not robust against the uncertainties that are present in the environments they operate in.
- software modules such as motion planning, localization, task planning and execution all usually have their own specific environment representation, limiting performance because of the replication of processing steps and the absence of reuse of information.

The related work concerning these issues is discussed in the following sections, followed by the main contributions of this thesis.

1.2.1 Robot hardware

Today's service robots have a number of common drawbacks:

- Fully functional service robots are only available to large research institutions who can afford to develop one. The robots that are available are far too expensive.
- They are not modular. Modular components with standardized interfaces i) allow the use of hardware components for a specific task, ii) allow a robot to be assembled from components of multiple developers and iii) eases maintenance of the robots.

- They are closed source. Similar to open source software, open hardware benefits from distributed developments and reviews, leading to better quality and lower costs. Furthermore, the availability of open hardware implementations of standardized interfaces makes it more likely that these standards are adopted since these act as reference implementations, as is argued in Reynolds and Wyatt (2011) for open-source software.

- Their manipulation capabilities are inadequate. Today's service robots are able to pick-and-place objects such as bottles and cans. However, their mechanics are not suitable for the fine manipulations skills that are required to, *e.g.*, prepare dinner, nor do they have sufficient strength to physically aid people.

Making robust, reliable and capable service robots available to a larger community is therefore a first step towards the introduction of these robots in our daily lives. Furthermore, the developments of these robots can be encouraged by introducing a standard, modular architecture, in which the components of various robots can be easily exchanged.

Over the past ten years many service robots have been developed. Common examples are the PR2 (Figure 1.3b, based on the design in Wyrobek et al (2008)), Care-O-bot (Graf et al, 2004, 2009) (see Figure 1.3c), Rollin' Justin (Ott et al, 2006; Fuchs et al, 2009), ARMAR-III (Asfour et al, 2006), Dynamaid (Stückler and Behnke, 2009), TWENDY-ONE (Iwata and Sugano, 2009) and HoLLiE (Hermann et al, 2013). Unfortunately, most of these robots are unique research platforms that are not available to other research institutions (Fuchs et al, 2009; Weisshardt et al, 2010). The platforms that are available such as the PR2 (Wyrobek et al, 2008) and the Care-O-bot 3 (Graf et al, 2009) are too expensive for many research institutions.

The importance of modularity has been acknowledged in a number of service robot designs. In Wyrobek et al (2008) it is recognized that a modular approach makes it possible to add specialized hardware and end-effectors. The Care-O-bot 3 also has a modular setup (Weisshardt et al, 2010) and has a modified, existing manipulator connected to an existing end-effector. Rollin' Justin (Ott et al, 2006; Fuchs et al, 2009) also has a separately designed base, torso and arms. Similarly, ARMAR-III (Asfour et al, 2006) and HoLLiE (Hermann et al, 2013) have been designed as a mobile platform with a modular upper body. However, only the PR2 contains open interfaces to use different grippers, arms or sensors and the designs in Stückler and Behnke (2009); Iwata and Sugano (2009) are in general less modular.

In the field of humanoid, walking robots, the importance of open hardware has long been recognized, with the iCub (Metta et al, 2008), DARwIn-OP (Ha et al, 2011) and NimbRo-OP (Schwarz et al, 2014) as well-known examples. These robots, however, are not suitable as service robots: at this point, a driving robot is more suitable than a walking robot and furthermore, the manipulation abilities of these robots are inadequate for service robots.

To boost the development of service robots, a modular service robot is required with standardized interfaces. Furthermore, it must be open hardware so that it can act as a reference implementation for future developments.

Contribution 1 *The first contribution of this thesis is SERGIO: a* modular, open hardware *design for a domestic service robot.*

SERGIO is an acronym for Second Edition Robot for Generic Indoor Operations. SERGIO has a fully holonomic platform for domestic service robots that can robustly cope with rigid, uneven surfaces and smoothly drive over thresholds such as doorsteps. Furthermore, it has an upper body that combines the vertical range and stability of a linear vertical actuator with the dexterity of a rotational joint using only two actuators. Finally, it is equipped with two 7-DoF anthropomorphic manipulators. The mechanical and electrical interfaces between the base platform, upper body and arms are clearly defined, allowing easy exchangeability of these

components. Its CAD drawings, part lists and electrical schemes will be released under an open hardware license on the Robotic Open Platform[1].

1.2.2 Motion planning

As was discussed in Section 1.1, one of the most important skills of a domestic robot is to plan and execute its motions quickly and safely. In practice, a motion planner is typically desired to be *complete*, *i.e.*, to return a solution if one exists to report failure otherwise, and to return a solution that is *optimal* and *correct*. However, there exists no motion planner able to achieve all these requirements in *real-time* while satisfying the *robot constraints* in presence of *moving obstacles* and *uncertainty* (Canny, 1988). Therefore, the motion planning problem at hand is often approximated by a simplified but still realistic version (Latombe, 1990), which introduces a trade-off between the various requirements. Due to the importance of motion planning for robotics, this subject has been thoroughly researched over the past fifty years. In fact, the amount of literature discussing motion planning has grown so large that it has become extremely difficult to select a combination of representations and search algorithms that is suitable for a particular application.

Motion planning consists of two steps: i) representing the robot and its environment and ii) searching a path in the represented environment (Hwang and Ahuja, 1992). In literature, either only a single step of the motion planning problem, *i.e.*, either the representation or the search algorithm, is discussed or, in more cases, not all requirements are discussed as some are considered not relevant for a specific robot or goal. In Ferguson et al (2005) some combinations of algorithms are discussed that, given a specific representation of the environment, find a path. The work in Giesbrecht (2004) gives an overview of algorithms that represent the environment and algorithms that find a path and discusses combinations of both. How the combined algorithms can deal with robot constraints is not discussed. There is also an overview of combinations of algorithms that find a path for robots with certain kinematic constraints (Laumond et al, 1998). It does, however, not consider how to deal with moving obstacles or a partially known environment. Also, motion planners are compared that combine multiple algorithms but with a focus on unmanned aerial vehicles (UAVs) (Goerzen et al, 2010; Dadkhah and Mettler, 2012) instead of mobile robots moving only on the ground plane. Moreover, these references do not consider how combined algorithms provide robustness against a dynamic environment. Next, there is an overview that discusses planners that have 'covering' as a specific goal (Choset, 2001). These planners assume a mostly static environment and thereby exclude robustness against a dynamic environment. Also, robot constraints are not taken into account. There are examples of combinations of multiple algorithms that make a well-performing motion planner in practice for a specific mobile robot (Thrun et al, 1999; Brock and Khatib, 1999; Stachniss and Burgard, 2002; Hsu et al, 2002; van den Berg et al, 2006; Dolgov et al, 2008; Marder-Eppstein et al, 2010). Such motion planners give insight in the choice for a combination of algorithms with respect to predefined requirements, but they do not give generic directions on how to get to this choice.

To design a motion planner for a practical situation, a generic methodology is required to select appropriate representations, search algorithms and an approach to combine these.

Contribution 2 *The second contribution of this thesis is a recipe for the selection of a combination of motion planning algorithms for navigation of mobile robots.*

This recipe consists of three parts: In the first part, an appropriate approach to combine representations and search algorithm classes is selected. In the following two parts a representation class and search algorithm class to accomplish the two steps of motion planning are selected.

[1] http://www.roboticopenplatform.org/wiki/SERGIO

1.2.3 Environment representations for motion planning

Among the motion planning problems the robot faces is navigation, *i.e.*, the ability of the base platform to move through the environment. The environment representation the robot uses for navigation needs to be three-dimensional in order to deal with typical obstacles in a human environment such as tables and chairs. Even if a prior representation is present, this representation needs to be updated using 2D sensors such as laser rangefinders and 3D sensors such as the Microsoft Kinect to account for changes in the environment, *e.g.*, furniture that has moved or people walking around in the environment. The obstacles that the robot encounters generally vary in size and shape and can be static as well as dynamic. Uncertainty typically arises from three sources (van den Berg et al, 2011): i) sensing uncertainty due to noisy and false sensor measurements, ii) uncertainty about the environment due to (partially) unknown parts of that environment and iii) robot position uncertainty due to localization errors and external disturbances acting on the robot. In the presence of this uncertainty, a robot still needs to guarantee that it is safe, *i.e.*, that it can ensure to come to a timely stop when a collision is imminent. As a result, existing navigation systems for domestic service robots usually suffer from the trade-off between being quick and being robust against uncertainties in the environment.

To achieve safe behavior, the objects in this environment are typically inflated with a 'safe' distance that encompasses the robot position uncertainty as well as the robot sensing uncertainty (Hsu et al, 2002; Chung et al, 2004). While this may ensure that a robot keeps a greater distance from obstacles, it may also prohibit a robot to traverse tight spaces. This is undesirable as tight spaces such as doorways are typically present in a domestic environment. Other approaches limit the robot's velocity based on distance to obstacles on the robot's path (Lingemann et al, 2005) or the amount of clearance on both sides of the robot (Fox et al, 1997). The former approach will fail when a moving obstacle emerges from occluded regions, *e.g.*, a doorway in a corridor. The latter approach is conservative in that it will scale down its velocity in a narrow corridor, while in such a confined part of the environment the robot could safely drive at higher velocities.

Instead of these approaches to approximate the uncertainty due to robot position inaccuracy and the limited sensor range, these sources of uncertainty can also be explicitly modeled to prevent collisions. A range of approaches estimate the probability of collision of the robot along its path (Missiuro and Roy, 2006; Burns and Brock, 2007; Guibas et al, 2009; van den Berg et al, 2011; Patil et al, 2012). These approaches plan a path by sampling the environment for feasible robot configurations that reduce the probability of collision during the execution of a path. These approaches result in robot motions with a lower probability of collision by explicitly considering the robot position and sensing uncertainty. However, these approaches do not take into account the uncertainty that arises due to unknown parts of the environment. In Marder-Eppstein et al (2010), a method is introduced that does track the unknown space in three dimensions. To guarantee safe behavior it is ensured that the robot never traverses this unknown space. However, it is assumed that the environment is mostly static, *i.e.*, it can not ensure safe behavior if an obstacle emerges from an occluded part of the environment. Furthermore, once any unknown space is marked as free it will remain free. This makes the representation over-confident in its assumption that space is free as this space can become occupied again in a changing environment. Hence, a time-dependent occupancy probability model is necessary, instead of merely tracking unknown space.

Finally, some approaches explicitly model the uncertainty that arises due to unpredictable moving obstacles, *e.g.*, humans (Philippsen et al, 2006, 2008; Rohrmüller et al, 2008). Moving obstacles are extracted from subsequent sensor readings and their position and velocity is estimated to obtain a probabilistic model that resembles the risk of collision. A moving obstacle can, however, be occluded up to an imminent collision.

Hence, current approaches that deal with uncertainties to allow safe navigation are not fully suited for a domestic environment. To improve on the trade-off between being quick and being robust against uncertainties in the environment, the three sources of uncertainty need to be explicitly modeled and integrated in the environment representation.

Contribution 3 *The third contribution of this thesis is a three-dimensional representation that explicitly models the three uncertainties that are present in a domestic environment to allow a motion planner to decide when it can move at a certain maximum velocity and when it must maintain a slower pace to ensure safety, based on the probability of collision.*

This is analogous to slowing down in hazardous situations in traffic.

1.2.4 The lack of context

Recently developed robotic software architectures allow for the execution of semantically represented action recipes that can deal with the presence of variation in both environment and robot, see, *e.g.*, Di Marco et al (2013); Janssen et al (2013). In the process of grounding the used semantic concepts, knowledge of the environment is needed, which is often represented and integrated in an ad-hoc way, where each of the software components, *e.g.*, for localization, navigation and manipulation, uses and maintains its own specific world model. As is recognized in, *e.g.*, Blumenthal et al (2013), this limits performance as certain processing steps need to be replicated in multiple modules and the reuse of functionality is limited.

A clear example can be found in motion planning. Unlike the world models used for, *e.g.*, task planning and execution, the representations used for motion planning typically do not explicitly distinguish the objects that are present, nor do they associate semantic knowledge with the measurements or track objects. In other words, the motion planning representation lacks *context*. Consequently, all volumes of a representation, *e.g.*, the voxels of a voxel grid or the nodes of an OctoMap, are assumed to be independent. However, this assumption is not valid in practice and limits the performance of motion planning systems: i) obstacles are poorly cleared, resulting in 'clutter', *i.e.*, single voxels that are still modeled occupied although the object they were initially associated with is no longer there, in the representation and ii) movements of obstacles cannot be explicitly accounted for. Furthermore, the robot cannot reason about the appropriate actions to take in case its task is obstructed due to the absence of semantic knowledge of the environment. In this example, the clearing problem would benefit from object tracking and semantic annotations of the obstacle could help the robot decide about the appropriate action.

World models in literature can be roughly divided in object-oriented representations and grid-based representations. In motion planning, it is common to use grid-based representations, examples of which can be found in Section 1.2.3.

Contrary to the world representations for motion planning, world representations that are designed for task planning and execution are usually object-oriented and semantic knowledge is associated with these objects. Examples of semantically annotated world models can be found in Pronobis and Jensfelt (2012); Elfring et al (2013). In the work of Pronobis and Jensfelt (2012), a probabilistic framework for semantic mapping has been presented which abstracts sensor information and integrates it into a semantic map in a probabilistic way. It is shown how this can be used as a topological map for navigation but does not present a complete geometric model that can be used down to the level of local motion planning. The world model in Elfring et al (2013) contains the position of known objects, but these objects have no associated volumetric information and unknown objects are not modeled.

Geometric and semantic information are combined in, *e.g.*, Zender et al (2008); Wurm et al (2011); Chiesa (2013). In Zender et al (2008), a multi-layered spatial representation is introduced. This system has one central world model consisting of different abstraction layers.

Table 1.1: Main contributions of this thesis.

	Contribution	Chapter
1	SERGIO: a *modular*, *open hardware* design for a domestic service robot	2
2	Recipe for the selection of a combination of motion planning algorithms for navigation of mobile robots	3
3	Three-dimensional representation for navigation based on the probability of collision	4
4	Using a *centralized*, *object-oriented* world model, with motion planning as an exemplary application	5

However, the layered approach hides the semantic knowledge from the low-level components such as motion planning. In Wurm et al (2011), the environment is modeled as hierarchies of octrees. Here, it is mentioned that semantic annotations can be used to facilitate data association or to adapt certain map properties and an example of a representation for tabletop manipulation is shown. It is assumed that that there exist methods to segment a point cloud, whereas in our approach the segmentation is actually aided by using the models that are already present in the world model. The method introduced in Chiesa (2013) also adds semantic information to a 3D occupancy grid. However, this approach does not scale beyond small indoor rooms.

The solution proposed in Blumenthal et al (2013) is to have a common and shared world model where each component has access to relevant parts of the environment. In this reference, however, it is not yet shown how this world model can be integrated with other software modules such as motion planning.

Contribution 4 *The fourth contribution of this thesis is therefore to demonstrate how a single, centralized, object-oriented world model can be used for multiple modules in the software stack of a service robot instead of having separate world models for each module.*

As an application, it is shown how this approach increases robustness and safety:

- while using this object-oriented world model as a basis for the planning representation
- while using task context to determine the goal regions with respect to world model objects
- while using symbolic knowledge of the task and goal to coordinate the motion planners

An essential property of this approach is that improvements of the world model directly lead to improvements in other modules such as navigation.

1.3 Outline of the thesis

This thesis is structured according to the contributions introduced in Section 1.2 and summarized in Table 1.1. In Chapter 2, the design of SERGIO is introduced. The design considerations for the base platform, the upper body and the manipulators are discussed and compared with the existing robots introduced in Section 1.2.1. Furthermore, the mechanical and electrical design are shown and the performance of the base platform as well as the reachability of SERGIO is compared with its predecessor AMIGO.

In Chapter 3, the recipe for motion planning of mobile robots is introduced, consisting of the three parts that are mentioned in Section 1.2.2 is discussed. Based on this recipe, possible

solutions to three motion planning problems are developed and compared with the solutions that were developed in practice.

A geometric representation for navigation of a mobile robot is the subject of Chapter 4. Here, the three sources of uncertainty introduced in Section 1.2.3 are explicitly modeled. The resulting representation has been extensively tested in simulation, laboratory experiments and a large scale experiment in the university library.

Based on the experiences in these experiments, a new navigation system has been developed which is introduced in Chapter 5. It is discussed how navigation can benefit from the use of this object-oriented world model as well as taking task-context into account. For comparison, this system was also tested in the university library.

Finally, the main conclusions of this research are summarized in Chapter 6, followed by a number of recommendations for further research.

Chapter 2

SERGIO: the open hardware design of a holonomic, anthropomorphic domestic service robot

Before service robots will be part of our daily lives, large-scale developments on both hardware and software are required. Nevertheless, service robots are currently only available to large research institutions that have the resources to develop one. The service robots that are available are usually far too expensive. Furthermore, the available service robots are not modular and their designs are closed-source, hampering further development. Therefore, a modular service robot is required with standardized interfaces. Furthermore, it must be open hardware so that it can act as a reference implementation for future developments. This chapter introduces SERGIO: a modular, open hardware design for a domestic service robot.
This robot has a fully holonomic platform with compliantly suspended Mecanum wheels, an upper body that combines the vertical range and stability of a linear vertical actuator with the dexterity of a rotational joint using only two actuators and two 7-DoF anthropomorphic manipulators. The mechanical and electrical interfaces between the base platform, upper body and arms are clearly defined, allowing easy exchangeability of these components. Its CAD drawings, part lists and electrical schemes will be released under an open hardware license on the Robotic Open Platform[1]*.*

2.1 Introduction

Robots are expected to play an increasing role in society over the next number of years. They can, *e.g.*, reduce the pressure on the home care system by assisting people with mobility impairments so that they can live independently at home for a longer time. Furthermore, robots will also increasingly work alongside people in industry. This illustrates the need for the development of versatile, reliable, robust and affordable robots.

As is mentioned in Weisshardt et al (2010), it is expensive and time consuming to develop these robots. Nevertheless, many service robots have been developed over the past ten years. The development of the PR2 (based on the design in Wyrobek et al (2008)) has concentrated on the design and implementation of a fully integrated development platform that is designed to be

[1] http://www.roboticopenplatform.org/wiki/SERGIO

safe and capable in human environments. The long term goal of the Care-O-bot project (Graf et al, 2004, 2009) is to develop a mobile robot able to assist people in their homes and combines the technological aspects with a user friendly design. The two-arm system in reference Ott et al (2006) is developed as a research platform to contribute to the manipulation skills of humanoid robots. Together with the platform introduced in Fuchs et al (2009), it also forms the mobile service robot Rollin' Justin. More humanoid research platforms are ARMAR-III (Asfour et al, 2006), Dynamaid (Stückler and Behnke, 2009), TWENDY-ONE (Iwata and Sugano, 2009) and HoLLiE (Hermann et al, 2013).

Similar to the computer industry, which quickly gained momentum after a standardized, open PC architecture was introduced, the development of service robots can greatly benefit from standardized interfaces that allow modular substitution of components. Modular components with standardized interfaces i) allow the use of hardware components for a specific task, ii) allow a robot to be assembled from components of multiple developers and iii) eases maintenance of the robots.

Figure 2.1: AMIGO (left) and SERGIO (right), without covers and arms.

Modularity has been an important issue in a number of these designs. In Wyrobek et al (2008) it is recognized that a modular approach makes it possible to add specialized hardware and end-effectors. The Care-O-bot 3 also has a modular setup (Weisshardt et al, 2010) and has a modified, existing manipulator connected to an existing end-effector. Rollin' Justin (Ott et al, 2006; Fuchs et al, 2009) also has a separately designed base, torso and arms. Similarly, ARMAR-III (Asfour et al, 2006) and HoLLiE (Hermann et al, 2013) have been designed as a mobile platform with a modular upper body. However, only the PR2 contains open interfaces to use different grippers, arms or sensors and the designs in Stückler and Behnke (2009); Iwata and Sugano (2009) are in general less modular.

As is argued in Reynolds and Wyatt (2011), open-source software implementations of a standard means that the standard is more likely to be of high quality, since these implementations act as reference implementations, and that the standard is more likely to be adopted. Unfortunately, most of these robots are unique research platforms that are not available to other research institutions (Fuchs et al, 2009; Weisshardt et al, 2010). The platforms that are available such as the PR2 (Wyrobek et al, 2008) and the Care-O-bot 3 (Graf et al, 2009) are too expensive for many research institutions.

In the field of humanoid, walking robots, the importance of open-hardware has long been recognized, with the iCub (Metta et al, 2008), DARwIn-OP (Ha et al, 2011) and NimbRo-OP (Schwarz et al, 2014) as well-known examples. These robots, however, are not suitable as service robots.

The AMIGO robot (see Figure 2.1) has been developed as a research platform by the Eindhoven University of Technology. AMIGO consists of a four-wheeled omni wheel platform with an extensible upper body, equipped with two seven degree of freedom Philips Experimental Robotic Arms (PERA, Rijs et al (2010)). The CAD drawings and electrical schemes of this robot will published on the Robotic Open Platform[2] (see Appendix B).Although this robot showed its capabilities in the RoboEarth project (Waibel et al, 2011) and successfully competed in the RoboCup@Home competition (Wisspeintner et al, 2009), with a second place in RoboCup 2014, a number of fundamental limitations of the base platform and the torso have been identified. Furthermore, the arms are not open hardware and the base platform and torso are fully integrated, contrary to the desired modular setup.

The contribution of this chapter is SERGIO: a modular, *open hardware* design of a domestic service robot with:

- clearly defined mechanical and electrical interfaces that enable the exchangeability of components and ease maintainance
- CAD drawings, part lists and electrical schemes that will be released under an open hardware license on the Robotic Open Platform[3]
- a fully holonomic platform for domestic service robots that can robustly cope with rigid, uneven surfaces and smoothly drive over thresholds such as doorsteps
- an upper body that combines the vertical range and stability of a linear vertical actuator with the dexterity of a rotational joint using only two actuators
- two 7-DoF anthropomorphic manipulators

In the next section, the design principles of the base platforms, upper bodies and manipulators of current domestic service robots are discussed, followed by the design of SERGIO's base platform in Section 2.3, its upper body and manipulators in Section 2.4 and electronics and modularity in Sections 2.5 and 2.6. In Section 2.7, the driving performance of the base platform and the reachability of the upper body are compared with AMIGO, followed by conclusions and future work in Section 2.8.

2.2 Best practices in service robots

Most of the robots introduced in Section 2.1 consist of a moving base, a torso with one or more degrees of freedom and one or two manipulators. Although many of the tasks that are

[2]http://www.roboticopenplatform.org/wiki/AMIGO
[3]http://www.roboticopenplatform.org/wiki/SERGIO

currently demonstrated using service robots can be performed with only one manipulator, an anthropomorphic robot with two manipulators and a moving torso has the advantage that it can also perform bimanual manipulation. Therefore, focus will lie on these anthropomorphic robots. In this section, various possibilities for base platforms, torsos and manipulators are discussed.

2.2.1 Base platforms

Base platforms for service robots can be divided in three categories: non-holonomic, semi-holonomic and fully holonomic platforms. A holonomic robot can drive in any direction without having to turn beforehand. Hence, both the number of controllable DoFs and the total number of DoFs equal 3 (two translations and one rotation). A non-holonomic robot has fewer controllable DoFs than the total number of DoFs. For example, a car is non-holonomic: it cannot drive sideways. A semi-holonomic platform is able to drive sideways. However, it first has to turn its wheels.

Non-holonomic robots have two differentially driven wheels and one or two passive caster wheels, *e.g.*, the Pioneer P3-DX, or four or more differentially driven wheels, *e.g.*, the Pioneer P3-AT. As is recognized, however, in Asfour et al (2006); Fuchs et al (2009); Stückler and Behnke (2009), this is not very beneficial for manipulation and maneuvering in tight spaces.

The semi-omnidirectionality of the robots in Wyrobek et al (2008); Fuchs et al (2009); Stückler and Behnke (2009) is obtained using steering wheels, *i.e.*, four individually driven steerable drives. Each drive has either one or two wheels, hence the total number of motors required for a platform is either eight or twelve. This large number of actuators is the main drawback of these platforms.

Therefore, a base using either Mecanum wheels or omni wheels is an interesting alternative. These wheels have small rolls on their circumference, allowing the wheels to move freely in axial direction (in case of omni-wheels) or under an angle (in case of Mecanum wheels). In Wyrobek et al (2008), it was argued that these fully holonomic platforms did not perform sufficiently robust in the presence of doorway thresholds, curbs and extension cords. Nevertheless, fully holonomic platforms are also successfully used in Iwata and Sugano (2009) and Hermann et al (2013).

As is mentioned in Fuchs et al (2009), platforms with more than three wheels are statically overdetermined and hence require some form of suspension in order to maintain good ground contact. The lack of compliance is also the main drawback of AMIGO's omni-wheel platform, which was designed as a Middle-Size League soccer robot to drive on soft carpets. Wheel suspension is not specifically addressed in Iwata and Sugano (2009); Stückler and Behnke (2009); Hermann et al (2013). Rollin' Justin (Fuchs et al, 2009) has an independent wheel suspension system. An example of adding compliance using a rotational degree of freedom between the front and rear axis can be found in, *e.g.*, Bischoff et al (2011). AMIGO is able to drive over extension cords and doorway thresholds, but due to the small diameter of the rollers on the circumference of the wheels, this does not look very smooth. Nevertheless, doorways are usually passed in forward direction and it is therefore expected that using Mecanum wheels results in a much smoother ride.

2.2.2 Upper bodies

Like AMIGO, the successor of the design in Wyrobek et al (2008) and the robot in Stückler and Behnke (2009) are equipped with telescopic spines. This results in a large vertical motion while keeping the center of gravity (CoG) in the middle of the robot. On the other hand, the robots in Asfour et al (2006); Ott et al (2006); Iwata and Sugano (2009); Hermann et al (2013) have rotational DoFs. The upper bodies in Asfour et al (2006); Iwata and Sugano (2009); Hermann et al (2013) each have one actuator per joint, whereas one of the DoFs in Ott et al (2006) is

passive, *i.e.*, the tilt of the chest is coupled to the base via tendons. The main advantage of having a pitch joint is that the robot can move its shoulders further forward, as is indicated in Ott et al (2006). In our experiences with AMIGO, the absence of a pitch joint is indeed a drawback: the robot needs to be positioned very accurately with respect to an object to allow a feasible grasping motion.

For large tilt motions, however, care has to be taken to prevent the robot from tipping over. In Fuchs et al (2009), this is solved by using a variable footprint, increasing the required number of actuators and the complexity and therewith the costs of the system. An alternative approach is presented in Hermann et al (2013), where a stability measure is optimized in motion planning. However, the safety of the robot preferably does not depend on software.

2.2.3 Manipulators

One of the key abilities of a service robot is the ability to transport objects. Hereto, the robots in Section 2.1 are all equipped with manipulators. These manipulators typically have up to seven degrees of freedom and most manipulators are either mechanically compliant or have torque sensors so that the controllers enable compliant control.

The robots in Ott et al (2006); Weisshardt et al (2010) have industrial robot arms with seven DoFs and harmonic drives. The arms in Ott et al (2006) are equipped with torque sensors to allow compliant control. Both arms, however, are too large and too heavy for a dual arm domestic robot. The robot in Hermann et al (2013) has arms with six DoFs using three Schunk Powerballs, lacking the possibility to do compliant control. The arms of the robot in Stückler and Behnke (2009) are actuated by Dynamixel servo actuators. Although these have a compliant mode in which they are backdrivable, true impedance control is not possible. The robots in Wyrobek et al (2008) and Iwata and Sugano (2009) also have seven DoFs but are mechanically compliant.

AMIGO's manipulators (Rijs et al, 2010) have seven DoFs, of which six DoFs are driven by three differential joints. The use of differential joints results in a compace and anthropomorphic design (Le Tien et al, 2007). All joints are equipped with absolute position sensors as well as torque sensors to allow impedance control. Nevertheless, this design is also not open hardware.

Although recent developments show that variable-stiffness actuation is a promising research direction, it is concluded that a rigid arm with torque sensors leads to a simpler design of both hardware and controllers. To keep the compact design and anthropomorphic appearance, a design with differential joints similar to, *e.g.*, Le Tien et al (2007); Rijs et al (2010) is desired. Compared to the current manipulators, a number of possible improvements is identified: i) reduce the backlash in the differential joints, ii) improve the absolute position sensors and iii) redesign the I/O to be compatible with the rest of the robot.

2.3 Mechanical design of the base platform

With the experiences with AMIGO and the considerations mentioned in Section 2.2.1, a base platform was designed which fulfills the following requirements:

- To minimize the number of required actuators for an omnidirectional platform, the base is fully holonomic, *i.e.*, it has either omni wheels or Mecanum wheels. With respect to costs, it is furthermore desired to i) use off-the-shelf parts whenever possible and ii) design parts in such a way that production costs decrease with increasing production volumes.

- Since the robot is supposed to operate in domestic and care environments, it must be able to match human walking velocities up to $v = 2.0$ m/s $= 7.2$ km/h. To come to a timely

stop in case of unexpected events, the robot must be able to accelerate to this velocity in 0.5 s, hence $a = 4.0$ m/s^2. To compute the required motor torques, the weight budget in Table 2.1 has been used.

- The width of the robot is limited to 600 mm to easily fit through doors.
- The robot must be able to drive through wheelchair-accessible areas. This implies that a vertical edge may up to 6 mm and a beveled edge with a slope up to 1 : 2 with a height up to 13 mm are allowed.
- To keep traction even at rigid, uneven surfaces, the wheels must be compliantly suspended.

Table 2.1: Estimated mass of the various bodyparts.

Part	$m_{\text{estimated}}$	Comments
Base	25 kg	
Torso	22 kg	
Arms	14 kg	2×7.0 kg
Head	2.0 kg	
Load	3.0 kg	2×1.5 kg
Total	66 kg	

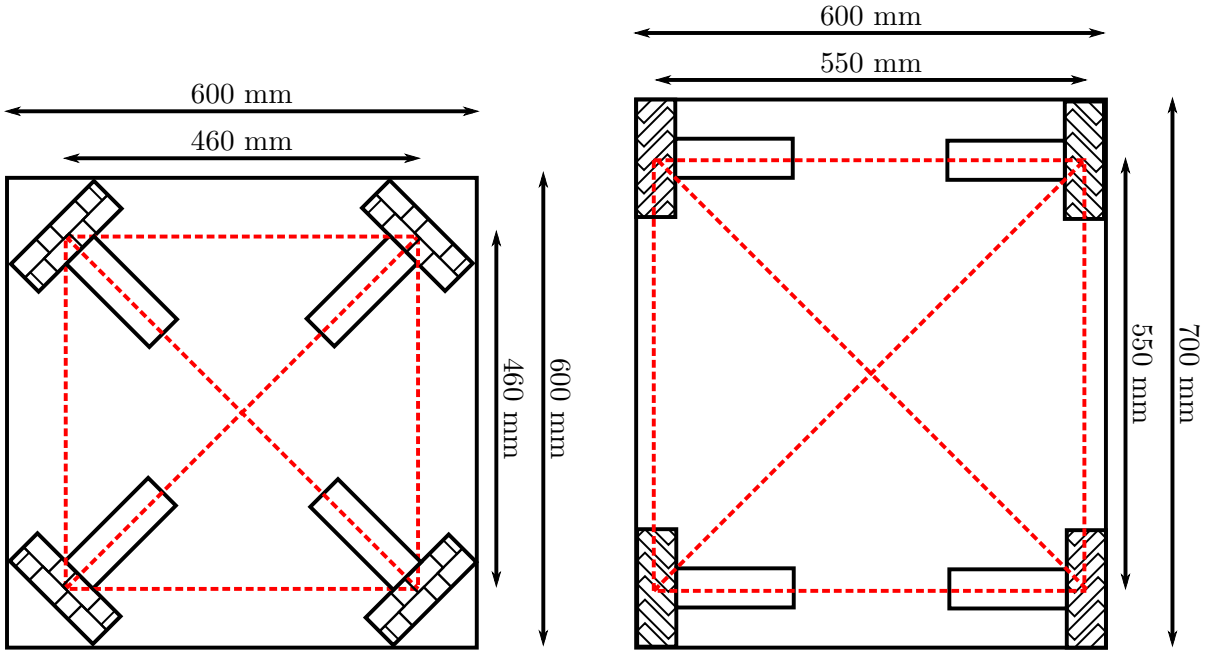
2.3.1 Mechanics

The difference between omni wheels and Mecanum wheels is the angle on which the rollers are placed on the wheel. Compared to omni wheels, Mecanum wheels have a number of advantages: With the same total width and wheel size, a Mecanum wheel platform has a wider track and a longer wheelbase (0.55 m) compared to an omni wheel platform (0.46 m), resulting in a more stable platform (see Figure 2.2). Together with a more favorable position of the motors this also results in more space for peripheral equipment. Finally, driving over doorway thresholds is smoother because the wheels roll along their circumference instead of the circumference of the small rollers.

Although some of the robots in Section 2.2.1 are equipped with a spring damper system to handle uneven floors, no specific attention has been paid to the resulting driving characteristics. Nevertheless, as is well-known in the automotive industry, suspension design has a great influence on, *e.g.*, the camber and castor angles over the wheel travel, the roll center height and dive and squat properties.

Therefore, various suspension concepts have been considered, with one or both axles mounted to the chassis with a rotational DoF, crossed bars and an independent suspension. Of these concepts, the independent multi-link suspension was selected. The main advantage of this concept is that the wheels will always be perpendicular to the surface, *i.e.*, when the robot is accelerating in any direction as well as when driving over obstacles (see Figure 2.3). This prevents vibrations that are otherwise introduced (see Figure 2.4). A second advantage is that the center of the base platform is not occupied by suspension parts.

The suspension geometry has been optimized using extensive dynamic simulations of the robot accelerating and driving over doorsteps of 15 mm. During the doorstep simulations, both compression and rebound of the wheels reached 20 mm but the camber angles remained below 0.3°. During normal use, *i.e.*, driving over a smooth floor with a maximum acceleration of



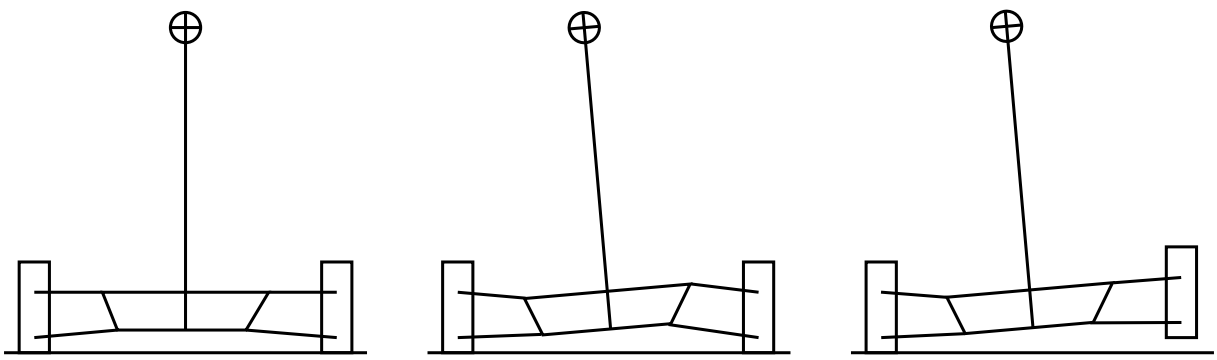
(a) Omni-wheels **(b)** Mecanum wheels

Figure 2.2: Schematic layout of an omni-wheel and an Mecanum wheel platform. As is indicated by the red dashed lines, the Mecanum wheel platform is more stable and has more space available for peripheral equipment compared to the omni wheel platform.

2.5 m/s^2, wheel compression and rebound are around 5 mm and as a result the camber angle stays below 0.1°.

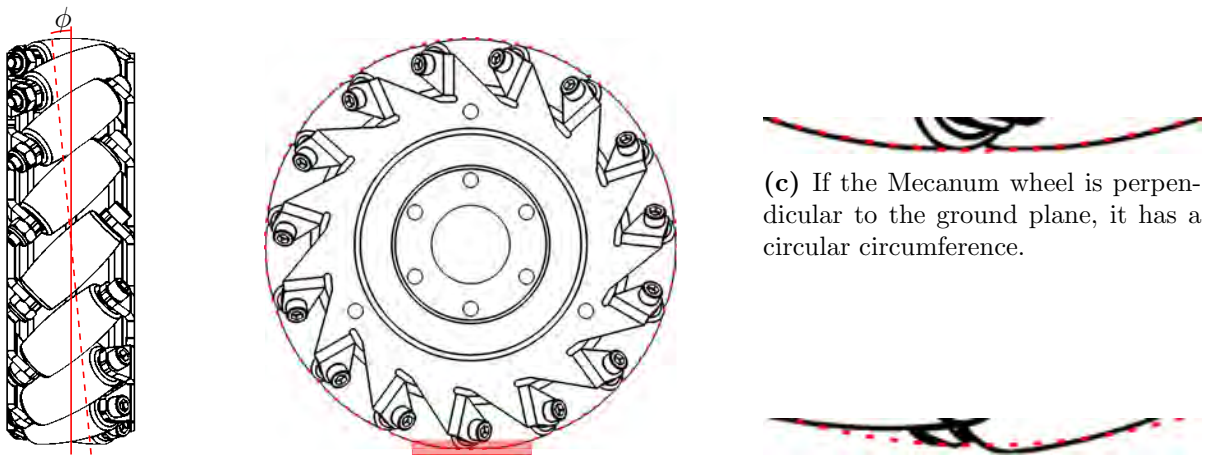
The multi-link suspension design in shown in Figure 2.5 with the six rods (blue) to constrain all six DoFs of the wheel upright (red), the drive train (yellow) and the spring and damper (green). The maximum forces in the suspension rods are estimated at 500 N based on the doorstep simulations. Using aluminum (Young's modulus $E = 69$ GPa) tubes with inner radius $r_i = 2$ mm and outer radius $r_o = 3$ mm, the buckling load of the longest tubes of $l = 0.186$ m is computed using Euler's buckling load formula:

$$F = \frac{\pi^2 EI}{l^2} = 1.0 \times 10^3 \text{ N} \tag{2.1}$$



(a) Neutral position **(b)** Accelerating sideways **(c)** On uneven surface

Figure 2.3: Rear view of the base: the multi-link suspension is designed to keep the wheels perpendicular to the ground in all three situations.



(a) Mecanum wheel. ϕ denotes the camber angle.

(b) Side view of a Mecanum wheel. Figures 2.4c and 2.4d zoom in on the red box at the bottom.

(c) If the Mecanum wheel is perpendicular to the ground plane, it has a circular circumference.

(d) With a 6° camber angle, the circumference is not circular , thereby introducing vibrations.

Figure 2.4: Front view, side view and bottom part of the side view of a Mecanum wheel to illustrate the importance of keeping the wheel perpendicular to the floor. A red dotted circle is show for reference.

which is two times the expected maximum force. The vertical stiffness of the suspension is provided by an extension spring, while the rubber bushing provides damping. All rods are made of aluminum tubes with off-the-shelf rod ends screwed into the end of the tubes, keeping the costs of these parts to a minimum. The only part requiring extensive machining is the wheel upright.

The drivetrain consists of 24 V, 90 W Maxon motors with 1 : 19 gearboxes. Based on the continuously permitted output torque and velocity of this motor and gearbox, $v_{\max} = 2.9$ m/s and $a_{\max} = 5.9$ m/s and hence this combination meets the requirement with some margin. To reduce the moving mass of the wheels and thereby improving the dynamic behavior of the robot, the motors and gearboxes are mounted in the chassis. The torque is transferred to the wheels using an off-the-shelf slip shaft with two cardan joints.

The blue suspension rods in Figure 2.5 are connected to the chassis (see Figure 2.6). This is a box structure made out of 0.5 mm steel sheets, resulting in a light but stiff and inexpensive construction. Merlons are used at the intersections of the sheets to allow accurate assembly. Sandwich constructions are used to create a stiff connection between the suspension rods and the chassis.

2.4 Mechanical design of the upper body

With the considerations of Section 2.2.2 in mind, a body was designed with the following requirements:

- Increase the kinematic reachability with respect to AMIGO by combining the vertical range and stability of a linear actuator with a rotational joint.
- The body must be strong enough to lift not only its own mass but also the mass of two fully stretched manipulators holding a load of 1.5 kg. Furthermore, it must be able to move from its lower to its upper position in approximately 4 s.
- A human-like appearance is sensible since the robot will operate in an environment designed around humans. Furthermore, a partly anthropmorphic appearance encourages

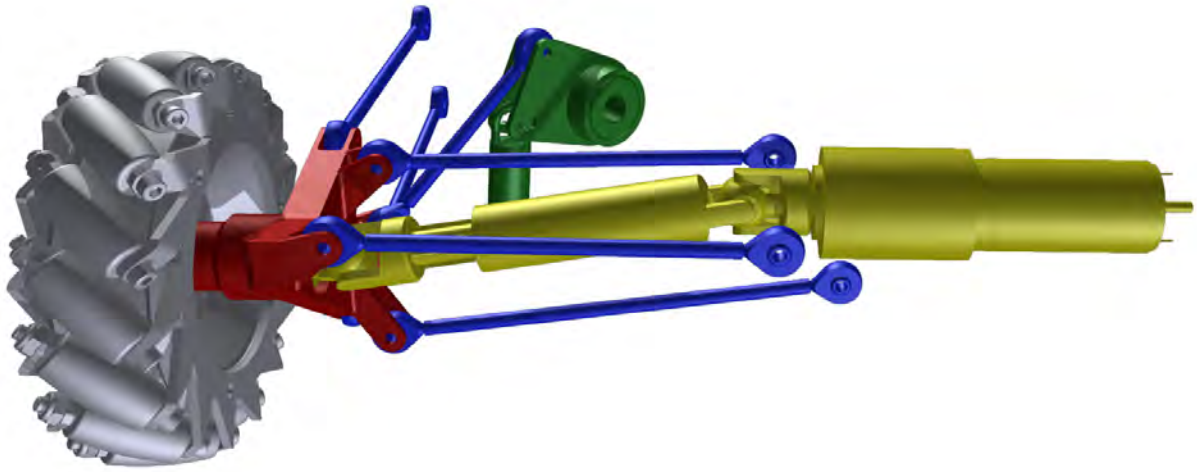


Figure 2.5: Overview of the suspension and drive train for one wheel with the suspension rods in blue, the spring and bushing in green, the drive train in yellow and the wheel upright in red. The vertical stroke of the wheel is approximately 4 cm.

people to interact (Mori, 1970; Hermann et al, 2013).

2.4.1 Mechanics

Again, a number of concepts for a body with a vertical hip motion and a chest rotation was investigated. Eventually, a four-bar mechanism (see Figure 2.7) was selected for the leg of the robot since this has fewer joints compared to a double parallelogram construction and is more compact to build than a telescopic setup. This mechanism can be seen as a lower leg and an upper leg which are kinematically coupled. The resulting near-vertical stroke of the hip is 684 mm, with a motion of only 50 mm in x-direction (see Figure 2.7c).

The tilt of the hip resulting from the leg motion can be compensated by the rotational hip joint. The stroke of 75° results in a shoulder motion of 0.31 m in forward direction. The resulting concept is shown in Figure 2.7. The human-like appearance with lower leg, upper leg and trunk is obvious in this view.

The mechanical design based on the concept in Figure 2.7 is shown in Figure 2.8. Like the chassis of the base platform in Section 2.3, the chassis of the upper body, the lower rear leg, the upper leg and the chest are inexpensive sheet steel box structures (see Figure 2.8a), again accurately assembled using merlons. The lower front leg consists of two rods similar to the suspension rods but with a larger diameter. Small safety bars are placed on the upper leg and the torso. In case something blocks the mechanism, these bars will trigger switches to stop the motors and to apply the brakes.

Both actuated joints are driven by a ball screw (see Figure 2.8b), which can provide a large force with high precision with a limited volume. Both ball screws are also driven by 24 V, 90 W Maxon motors, in this case equipped with a brake. By placing gas springs parallel to the ball screw, the effective force is increased since the actuator can now be used to either push or pull the mechanism and the required force when standing still in an upright position is decreased. Digital calipers are used to obtain an accurate absolute position measurement for both DoFs.

2.4.2 Manipulators

The manipulator of SERGIO is inspired by existing anthropomorphic 7-DoF arms, see, *e.g.*, Le Tien et al (2007); Rijs et al (2010). The main points of improvement compared to the

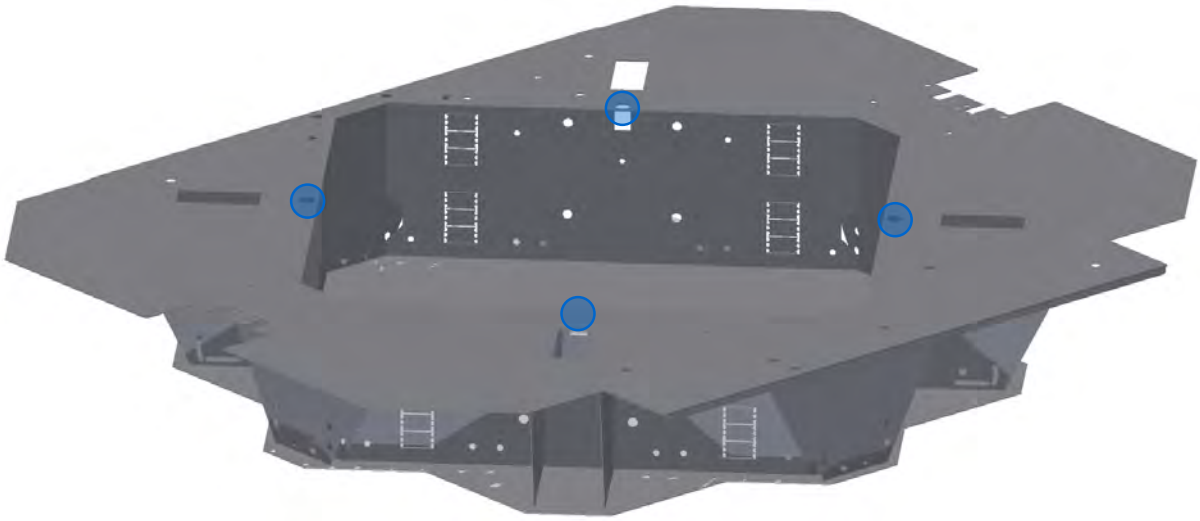


Figure 2.6: The chassis is built as a box structure made out of 0.5 mm steel sheets, resulting in a light but stiff and inexpensive construction. The blue circles indicate the mounting points for the upper body.

current arm that were identified in Section 2.2.3 were reducing the backlash of the differential joints and improving placement of the absolute position sensors. Therefore, the requirements can be defined as:

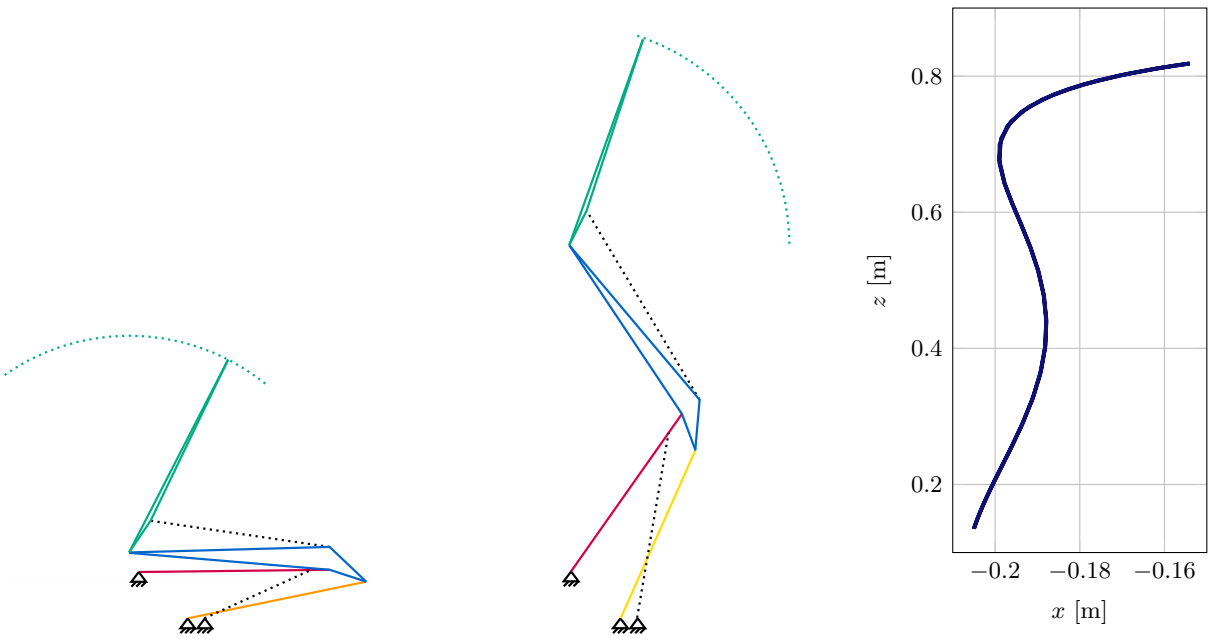
- Anthropomorphic arm with seven DoFs. The desired reach between shoulder joints and tool center point (TCP) is equal to the reach of the AMIGO arm: 0.73 m.
- A payload of 1.5 kg is desired while limiting the moving mass of the arm to 4.5 kg and the stationary mass to 4.0 kg. This is also similar to the AMIGO arm.
- The joint velocities must be similar to those of the AMIGO arm as well.
- Torque sensors to allow impedance control and absolute sensors to prevent a homing procedure at startup.

The result of the redesign of the arm is shown in Figure 2.9.

Like the AMIGO arm, the shoulder and elbow of the new arm consist of differential drives, with another rotational joint in the shoulder. The main difference of these joints is the use of spiral bevel gears instead of straight-cut bevel gears, causing the arms to run more smoothly. As a result, the gears are assembled tighter, significantly reducing the backlash. Another important difference is that the shoulder differential is driven by two coaxial shafts. As a result, the upper arm can be connected to this joint using two ears. The wrist of the arm is equipped with two spindle drives, which provide a high stiffness with only a small number of parts.

All joints are equipped with torque sensors to allow interaction with the environment using impedance control. Hereto, the gearboxes are mounted using flexible elements, such that the joint torques can be determined by measuring the deformation of these elements using optical sensors. Furthermore, absolute sensors are present to prevent homing procedures upon startup. Depending on the placement, potentiometers or hall sensors are used to this end.

The complete kinematic structure of the torso and the manipulators can be seen in Figure 2.10. As can bee seen in Table 2.2, the joint ranges of this new design are similar to those of AMIGO. The main difference between the kinematics of the manipulators is the stroke of the shoulder yaw and pitch joint, which has increased from $\pi/2$ rad to π rad (yaw) and from π rad to $9\pi/5$ rad (pitch) to accommodate the pitch of the upper body. Compared to the AMIGO



(a) Upper body in lower position

(b) Upper body in upper position

(c) Hip movement in the $x - z$ plane.

Figure 2.7: Schematic overview of the upper body concept. The lower rear leg (red), lower front leg (yellow) upper leg (blue) and mounting frame form a four-bar mechanism, resulting in a near-vertical stroke of the hip of 0.68 m. The rotational joint in the hip allows the upper body (green) to pitch (the dashed arcs denote the stroke). The ball screws driving both degrees of freedom are depicted as dashed lines.

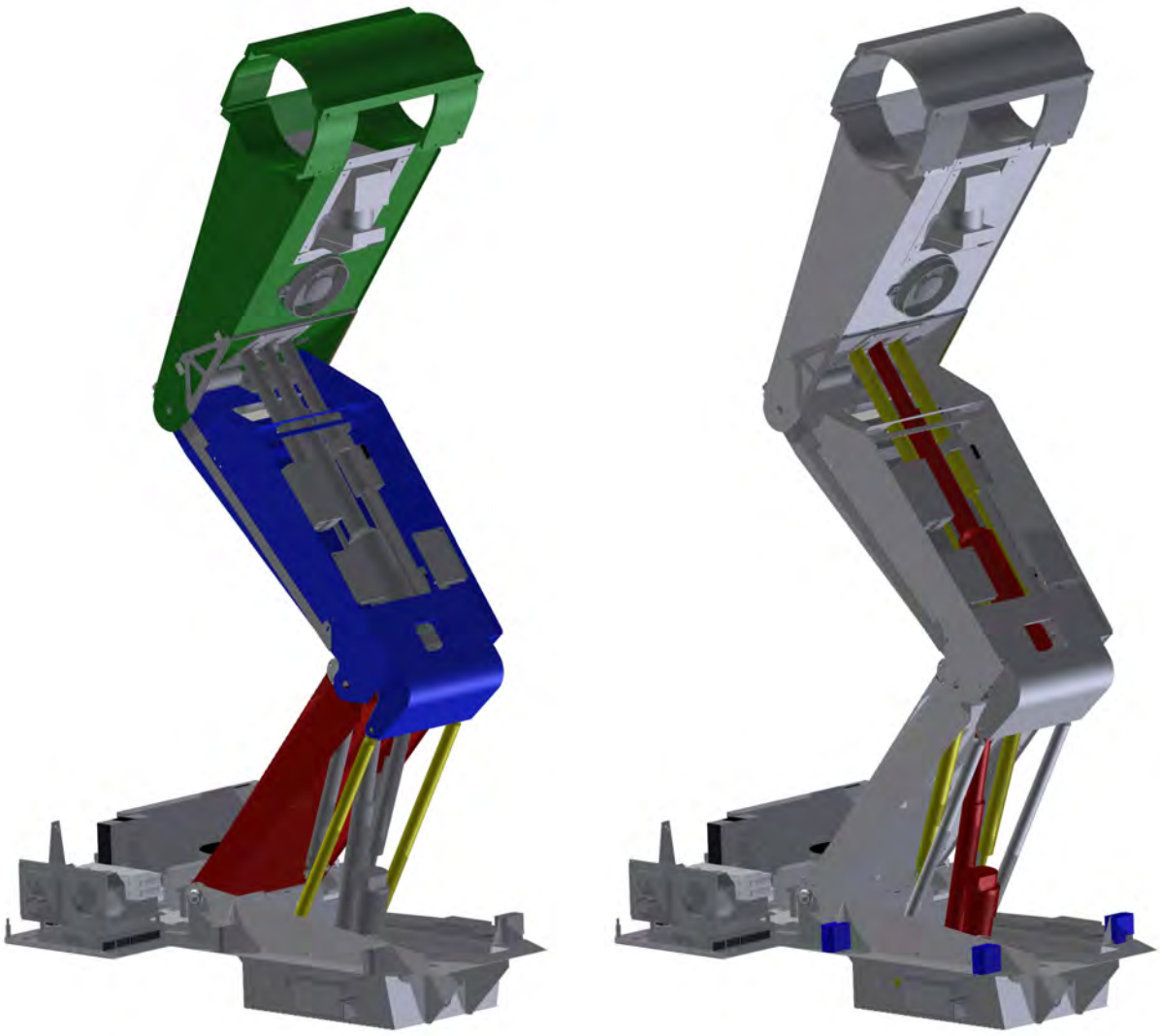
arm, the maximum velocity of the shoulder joints has increased. The maximum velocity of the elbow and wrist joints has slightly decreased.

2.5 Electronics

2.5.1 Input/Output

Domestic service robots typically require many inputs and outputs (I/O), *i.e.*, to control the motors, to read the encoder values and additional sensor inputs and outputs. Compared to industrial robots, however, the space for I/O in the robot is rather limited. Therefore, a small general purpose I/O board has been developed (see Figure 2.11), supporting up to three servo axes. Since the base platform contains four wheel motors, two of these boards are required. Furthermore, the torso contains one board and each arm contains three boards. The internal amplifiers of these I/O boards can supply up to 3 A, which is sufficient for the arms. If more power is required, analog outputs can be used in combination with external amplifiers. This solution is implemented for the base platform and the torso, where the motors are driven by Elmo Violin amplifiers. Furthermore, redundant analog and digital I/O is present for additional sensors, *e.g.*, to read the battery and emergency switch status, read the SPI data from the absolute sensors in the torso and to read the absolute position sensors as well as the force sensors of the arms. The entire board is not larger than 110 mm × 50 mm.

The I/O boards communicate with the computers via EtherCAT, which has been selected to allow control on the PC's with high sampling rates and low jitter. Furthermore, EtherCAT can also supports other protocols such as CANopen and also has solutions for RS232/RS485



(a) Torso design with the lower rear leg (red), lower front legs (yellow), upper leg (blue) and trunk (green).

(b) Torso actuators (red), gas springs (yellow) and base interface (blue).

Figure 2.8: Torso design with the main parts (left) and the drive trains and mount to the base platform (right).

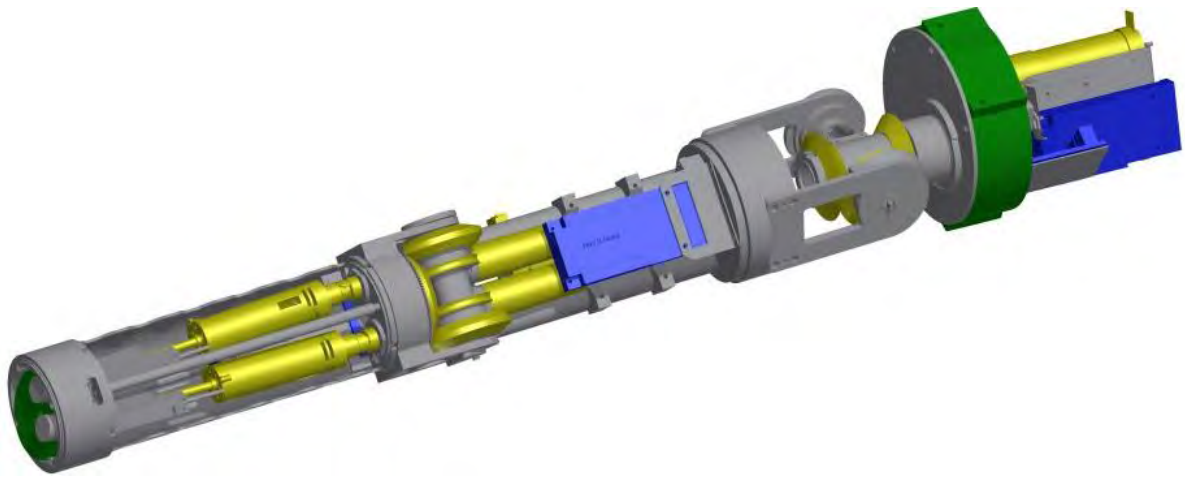


Figure 2.9: CAD drawing of the design for a 7-DoF manipulator for SERGIO with the drive trains in yellow, the electronics in blue and the interfaces to the torso and the end-effector in green.

Table 2.2: Joint ranges (rad) and velocities (rad/s) of AMIGO and SERGIO. The joint directions can be found in Figure 2.10.

	AMIGO			SERGIO		
	min	max	ω	min	max	ω
Shoulder yaw joint (q_1)	$-\pi/2$	0.0	1.47	$-\pi/2$	$\pi/2$	2.81
Shoulder pitch joint (q_2)	$-\pi/2$	$\pi/2$	1.47	$-4\pi/5$	π	2.81
Shoulder roll joint (q_3)	$-\pi/2$	$\pi/2$	2.27	$-\pi/2$	$\pi/2$	2.81
Elbow pitch joint (q_4)	0.0	2.23	2.80	0.0	2.4	2.51
Elbow roll joint (q_5)	-1.83	1.83	2.80	-2.6	2.6	2.51
Wrist pitch joint (q_6)	-0.95	0.95	3.54	-0.79	0.79	3.13
Wrist yaw joint (q_7)	-0.61	0.61	3.54	-0.52	0.52	3.13

communication. This allows to communicate with a large variety of sensors and actuators over a single bus.

2.5.2 Sensors, computers and power

The head of the robot is a Kinect camera mounted on a pan-tilt unit using RX-64 Dynamixel servo actuators. The Kinect is used for 3D navigation and people- and object detection and recognition. Furthermore, a tilting laser range finder (LRF) is present in the torso for people detection. For human-robot interaction, a directional microphone, a speaker and an LED bar are present. The base platform houses a second LRF for localization and navigation.

The robot houses three Core i7 mini ITX PC's with 16 GB RAM, two of which are located in the base while the third, extended with an NVIDIA graphics adapter to allow GPGPU, is located in the torso. The communication between the computers is handled by a Gigabit Ethernet router with a wireless link (both 2.4 GHz and 5.0 GHz) to a base station.

Like AMIGO, SERGIO has four Makita 24 V, 3.3 Ah power tool batteries that can be substituted within seconds. This has proven to be very convenient for a research platform since one never has to wait for the robot to charge.

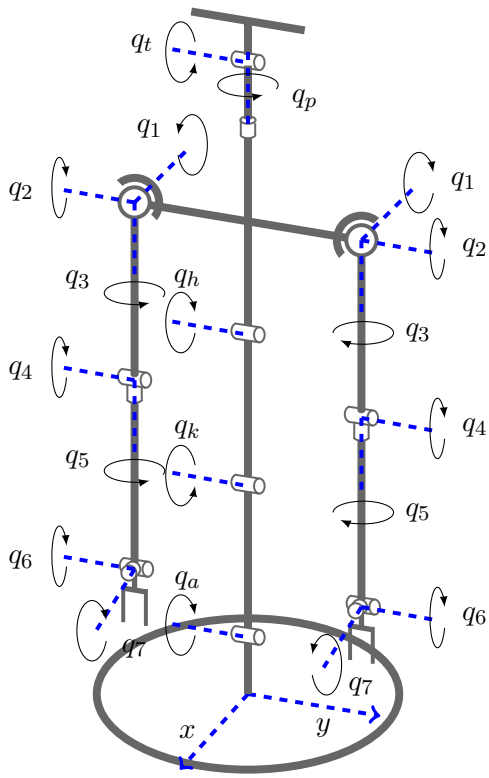


Figure 2.10: Kinematic configuration of the robot. The blue dashed lines and the arrows indicate the axes and direction of rotation. Furthermore, x and y indicate the definition of the x- and y-axis of the robot, q_a, q_k, q_h the ankle, knee and hip joints of the upper body, q_p, q_t the pan and tilt joint of the neck and $q_{1,\ldots,7}$ the manipulator joints (see also Table 2.2)

.

2.6 Modularity

As was mentioned in Section 2.1, modularity has been an important design aspect. In order to exchange body parts of the robot, both the mechanical and electrical connections between these parts needs to be defined. Mechanically, the body connects to the base platform using four M6 threads on a square of 250 mm × 250 mm as indicated by the blue circles in Figure 2.6 and the blue blocks in Figure 2.8b.

Instead of developing a custom electrical interface, the connection between base and body is established using a Harting docking frame with four modules:

1. A power connection with a maximum current of 70 A is present. A raw voltage supplied to the individual components will be 24 V. This raw voltage is only used by the actuators and is cut off in case a safety feature of any of the body parts is triggered, *e.g.*, a kill switch or an end-switch.

2. The second connection provides clean 24 V, *i.e.*, filtered by DC-DC converters, for the remaining hardware components such as computers, sensors and I/O boards. Furthermore, it contains the safety connection that ensures that whenever a safety feature of one body part is triggered, the entire robot stops moving and brakes are applied. A second safety line is present to re-enable the hardware once again.

3. An ethernet connection is present to accommodate a base and a body that both have computers. Having a computer in both parts makes it possible to operate them separately.

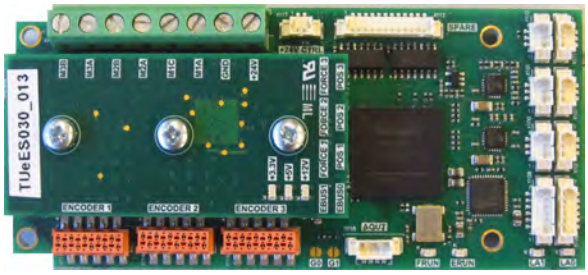


Figure 2.11: A multiple purpose EtherCAT I/O board, with three encoder inputs and three motor outputs to control three servo axis, six analog inputs, two analog outputs and eight digital pins (either input or output).

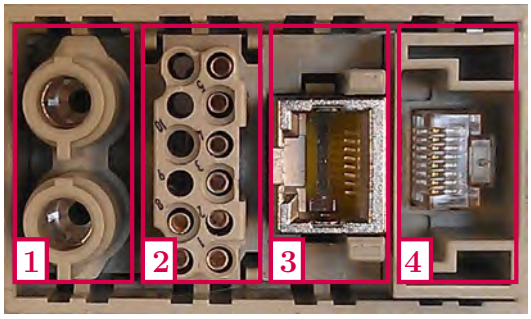


Figure 2.12: Harting docking frame with power connection (1), logic and safety connection (2), ethernet connection (3) and EtherCAT connection (4).

4. An EtherCAT connection is present so that all body parts can be controlled by one PC.

Using this interface, assembling or separating the base platform and the upper body can be performed in a minute.

Figure 2.13 shows a section of the torso-arm interface. Manipulators with either a rectangular mount up to 0.124 m × 0.90 m as well as a circular mount with a diameter up to 0.140 m fit into this connection. If less space is required, an adapter piece can be fitted between the arm and the torso. Not that this shape runs over the entire width of the torso, as can be seen in Figure 2.8. As a result, there is sufficient space to put actuators for the shoulder joint inside the torso. Similar to the base-torso interface, a Harting docking frame is used for the electrical connection. Again, a power connection, (logic) power connection, safety lines and EtherCAT are present.

2.7 Experimental results

In this section, a number of experiments is discussed to assess the driving performance of SERGIO compared to AMIGO. Furthermore, the reachability of the torso and the manipulators is investigated. Since building the arm has only just been completed, its performance has not yet been evaluated.

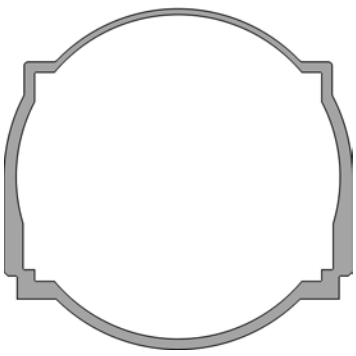


Figure 2.13: The torso-arm interface is suitable for manipulators with a rectangular or round circumference. Due to the size, there is sufficient space available to put actuators for the shoulder joint.

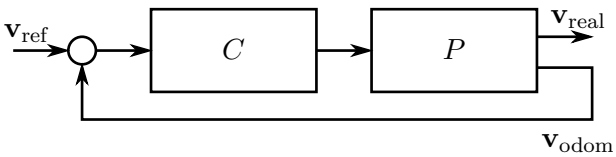


Figure 2.14: The controller C is designed such that the measured velocity $\mathbf{v}_{\text{odom}}$ tracks the reference velocity $\mathbf{v}_{\text{ref}}$. In case of slip, however, the actual velocity $\mathbf{v}_{\text{real}} \neq \mathbf{v}_{\text{odom}}$ and hence $\mathbf{v}_{\text{odom}} \neq \mathbf{v}_{\text{real}}$.

2.7.1 Base platform

As was mentioned in Section 2.1, one of the largest limitations of the AMIGO robot was the slip of the platform while driving over rigid surfaces. Even a slight unevenness can cause one or two wheels to lose traction. This introduces two problems (see Figure 2.14): i) the robot is not able to track the desired velocity, *i.e.*, $\mathbf{v}_{\text{real}} \neq \mathbf{v}_{\text{ref}}$, and ii) the velocity measured by the encoders, or odometry does not accurately represent the actual velocity, *i.e.*, $\mathbf{v}_{\text{odom}} \neq \mathbf{v}_{\text{real}}$. This inaccurate odometry data in turn is a poor input for a localization algorithm, thereby limiting localization accuracy.

To investigate the difference in driving behavior between AMIGO and SERGIO, both robots were set to drive forward and sideways in a straight line with varying velocities. Both robots use a diagonal velocity controller. For SERGIO, the 3×3 diagonal controller acted directly in the v_x, v_y, v_ϕ space. The wheels of AMIGO, however, might loose ground contact due to the lack of compliance and therefore a 4×4 diagonal controller was used that controlled the wheel velocities. Since the performance is based on both hardware as well as controller design, the velocities were measured not only using the wheel encoders but also using an inertial measurement unit (IMU). Measuring velocity using a IMU was preferred over a measurement system using an external sensor setup because this enabled experiments on multiple surfaces. Furthermore, data from absolute localization, *e.g.*, (Fox, 2001), proved to be too noisy. The acceleration measured by the IMU was compensated for gravity effects and integrated. The resulting error $\mathbf{e}_{v,\text{real}}$ was filtered to remove the drift. Since driving forward and sideways showed similar results, only the results for driving in forward direction are presented. The result for $v_x = 0.75$ m/s can be seen in Figure 2.15.

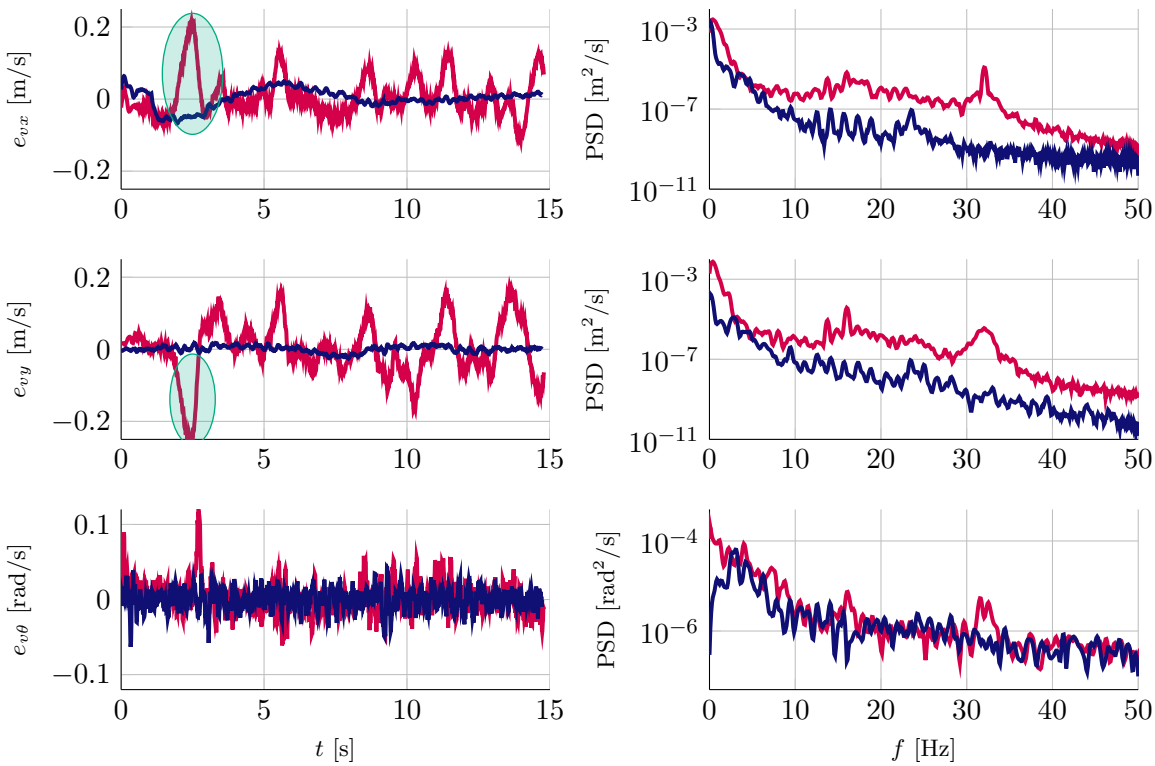


Figure 2.15: Velocity error $e = v_{\text{ref}} - v_{\text{real}}$ in x-, y- and θ-direction for AMIGO (red) and SERGIO (blue). In both x- and y-direction, AMIGO clearly has a significantly larger velocity error. The maximum errors for AMIGO occur at the green ellipses, where the robot 'steps' 0.14 m aside.

In this figure, it can be clearly seen when AMIGO loses traction as this results in an error in both v_x and v_y, *e.g.*, around $t = 2.4$ s (see the green ellipses). The velocity error in y-direction $e_{vy} = 0.28$ m/s at that point. As a result, the robot 'steps' 0.14 m aside at this point. SERGIO, on the other hand, has a maximum error of 0.025 m/s in this direction. AMIGO's loss of traction can also be seen in the x-direction, where the maximum error is 0.22 m/s. SERGIO has a maximum error of only 0.069 m/s. In the frequency domain, the error for SERGIO is lower at all frequencies in both x- and y-direction.

Compared to AMIGO, the RMS value of the error for $v_x = 0.75$ m/s has decreased from 0.057 m/s to 0.026 m/s in x-direction and from 0.077 m/s to 0.009 m/s in y-direction. In x-direction, a low-frequency error is visible at the start of the trajectory. This effect is visible at every velocity but only at the start. Since the acceleration was $a_x = 0.7$ m/s^2 for every experiment, it is believed that this induces a 'rocking' motion of the robot which is inaccurately filtered from the IMU output signal. The error in rotational velocity does not differ significantly for both robots. An overview of the maximum errors of AMIGO and SERGIO at various velocities can be seen in Table 2.3, confirming that SERGIO outperforms AMIGO in this test at every velocity.

Table 2.3: Velocity error $e = v_{\text{ref}} - v_{\text{real}}$ when driving in a straight line. All velocities are in [m/s].

	AMIGO				SERGIO			
v_x	e_{vx}^{max}	e_{vx}^{RMS}	e_{vy}^{max}	e_{vy}^{RMS}	e_{vx}^{max}	e_{vx}^{RMS}	e_{vy}^{max}	e_{vy}^{RMS}
0.25	0.12	0.031	0.12	0.038	0.059	0.016	0.019	0.007
0.50	0.18	0.052	0.25	0.068	0.051	0.014	0.023	0.008
0.75	0.22	0.057	0.28	0.077	0.069	0.026	0.025	0.009
1.00	0.15	0.050	0.24	0.061	0.087	0.027	0.032	0.011

Besides the error $\mathbf{e} = \mathbf{v}_{\text{ref}} - \mathbf{v}_{\text{real}}$ plotted in Figure 2.15, the difference between the real velocity and the velocity measured by the wheel encoders $\mathbf{e}_{\text{odom}} = \mathbf{v}_{\text{real}} - \mathbf{v}_{\text{odom}}$ is important. This is shown in Figure 2.16. These errors show a large similarity to Figure 2.15, confirming that SERGIO indeed provides much more reliable odometry data than AMIGO. This is also confirmed by the experiments with different velocities, as is summarized in Table 2.4. Again, both the maximum errors as well as the RMS values are of SERGIO are much lower than of AMIGO. Furthermore, these results indicate that the errors in Fig 2.15 can be attributed to slip of the platform rather than poor control performance.

Although doorsteps are getting less common in houses and especially care facilities where wheelchair accessibility is important, small thresholds such as cables and doorsteps smaller than

Table 2.4: Odometry velocity error $e_{\text{odom}} = v_{\text{real}} - v_{\text{odom}}$ when driving in a straight line. All velocities are in [m/s].

	AMIGO				SERGIO			
v_x	e_{vx}^{max}	e_{vx}^{RMS}	e_{vy}^{max}	e_{vy}^{RMS}	e_{vx}^{max}	e_{vx}^{RMS}	e_{vy}^{max}	e_{vy}^{RMS}
0.25	0.12	0.032	0.12	0.038	0.068	0.020	0.018	0.007
0.50	0.18	0.048	0.23	0.060	0.047	0.014	0.023	0.008
0.75	0.25	0.062	0.28	0.079	0.073	0.029	0.029	0.011
1.00	0.16	0.047	0.24	0.053	0.052	0.027	0.034	0.027

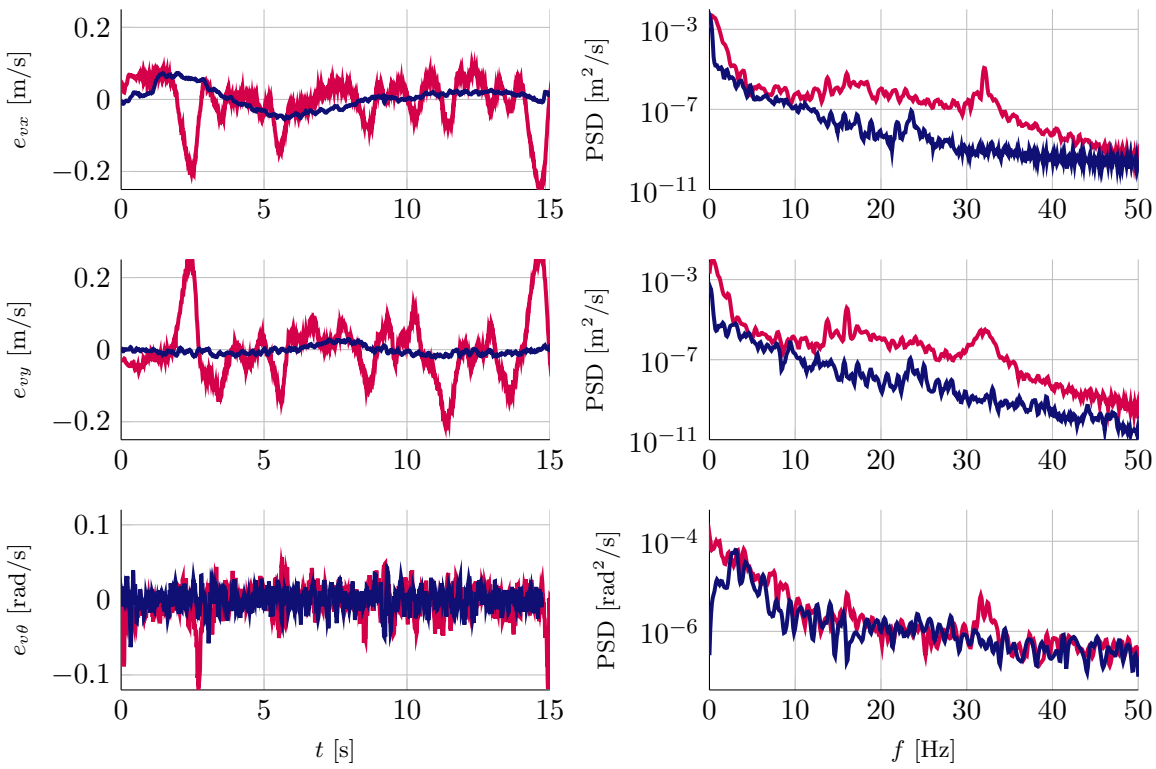


Figure 2.16: Odometry velocity error $e_{\text{odom}} = v_{\text{real}} - v_{\text{odom}}$ in x-, y- and θ-direction for AMIGO (red) and SERGIO (blue). Due to the resemblance to Figure 2.15, it is concluded that the poor performance of AMIGO is attributed to slip of the base platform rather than poor control performance.

1 cm should not pose a problem. In Wyrobek et al (2008), it was argued that fully holonomic platforms do not perform sufficiently robust in the presence of these small thresholds. This is also visible with AMIGO. If it approaches, *e.g.*, a doorstep exactly perpendicular on its driving direction, the small rolls hook onto the doorstep and pull the robot up. Due to the large ground clearance of 38 mm, even large thresholds can be taken this way. Nevertheless, if a doorstep is approached under a slight angle, the diameter of the small rolls is dominant and the robot cannot get over it. This is illustrated in Figure 2.17, where the robots drive over a threshold of 6 mm at a slope of 45°. The approach angle is 60°. As can be seen in the red lines in the upper and middle plot of Figure 2.17, the robot has trouble driving over the threshold. At $t = 2.84$ s (indicated by the green ellipses), it encounters the doorstep and slides along the doorstep for 2 s before crossing it. During this time, it has moved 18 cm along the doorstep, which is clearly undesired.

The drawbacks of AMIGO were known when SERGIO was designed. Under the assumption that the robot usually encounters thresholds when driving forward, *e.g.*, through a door, it is expected that in practice Mecanum wheels perform better than omniwheels because the wheel diameter is dominant rather than the diameter of the rolls. Furthermore, the suspension should lead to better traction. This comes at the expense of a decreased ground clearance of 17 mm. As can be seen in Figure 2.17, the forward velocity can not be tracked exactly and some oscillations occur. Nevertheless, the error in y-direction is much smaller than for AMIGO, indicating that the robot drives accurately in a straight line over the threshold.

Conclusively, it can be stated that SERGIO's compliantly suspended Mecanum wheel platform shows much better driving characteristics than AMIGO's rigid omni-wheel platform. Nevertheless, further experiments will have to demonstrate the effect of the new platform on other tasks. The main challenges that are expected are navigation, due to the large, rectangular foot-

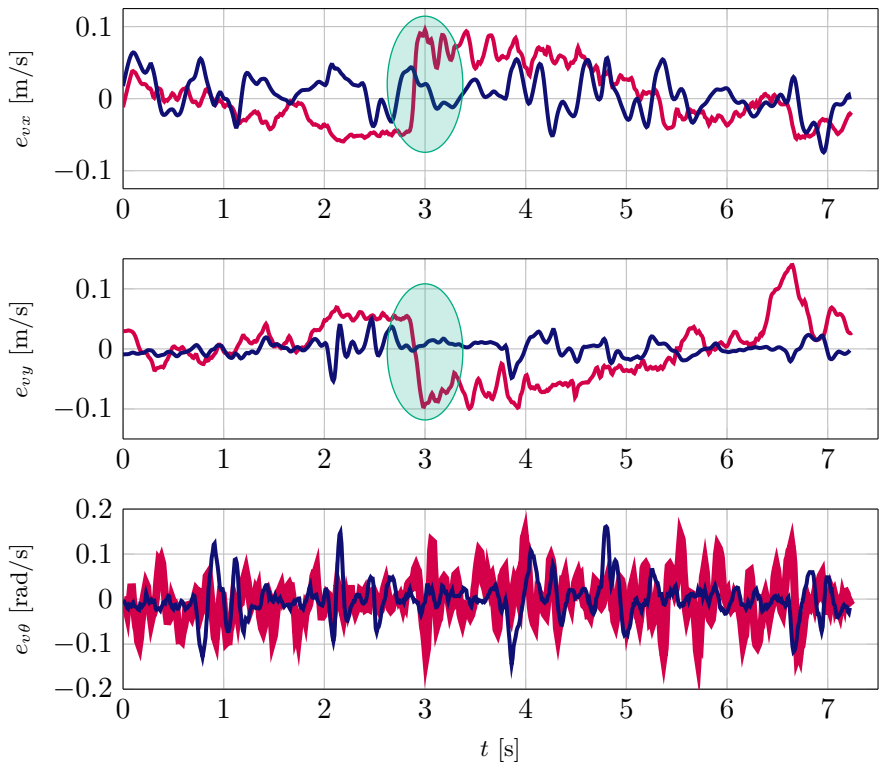


Figure 2.17: Velocity error $e = v_{\text{ref}} - v_{\text{real}}$ in x-, y- and θ-direction for AMIGO (red) and SERGIO (blue) when driving over a small doorstep. In the areas indicated with the green ellipses, AMIGO slides along the doorstep while SERGIO simply drives over it.

print and camera localization, due to the rocking motion that is introduced by the compliance in the suspension.

2.7.2 Upper body

As mentioned in Section 2.1, the new upper body gives SERGIO a much larger reach than AMIGO has. To illustrate this, the workspace is characterized using reachability spheres as is described in, *e.g.*, Zacharias et al (2007); Hermann et al (2013). Hereto, the kinematic ability of the TCP to reach each pose in the spatially and angularly discretized workspace is checked. In each cube of the discretized workspace a sphere with a diameter equal to the width of the cube is inscribed. Subsequently, on each sphere a number N equally distributed points is generated. For each point, it is investigated whether an IK solution can be found for this position of and the orientation towards the center of the sphere (note that a the rotation around this vector is left unconstrained). The colors in the following figures indicate for what percentage of the N points an IK solution has been found (see Figure 2.18). Similar to Zacharias et al (2007), the TCP is located in the wrist. In Figures 2.19a and 2.19b an overview of the results is given, where the increase in reach in every direction is clear. In total, the workspace has increased by a factor three by the new upper body design and increased range of the shoulder pitch joint.

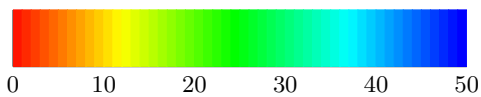
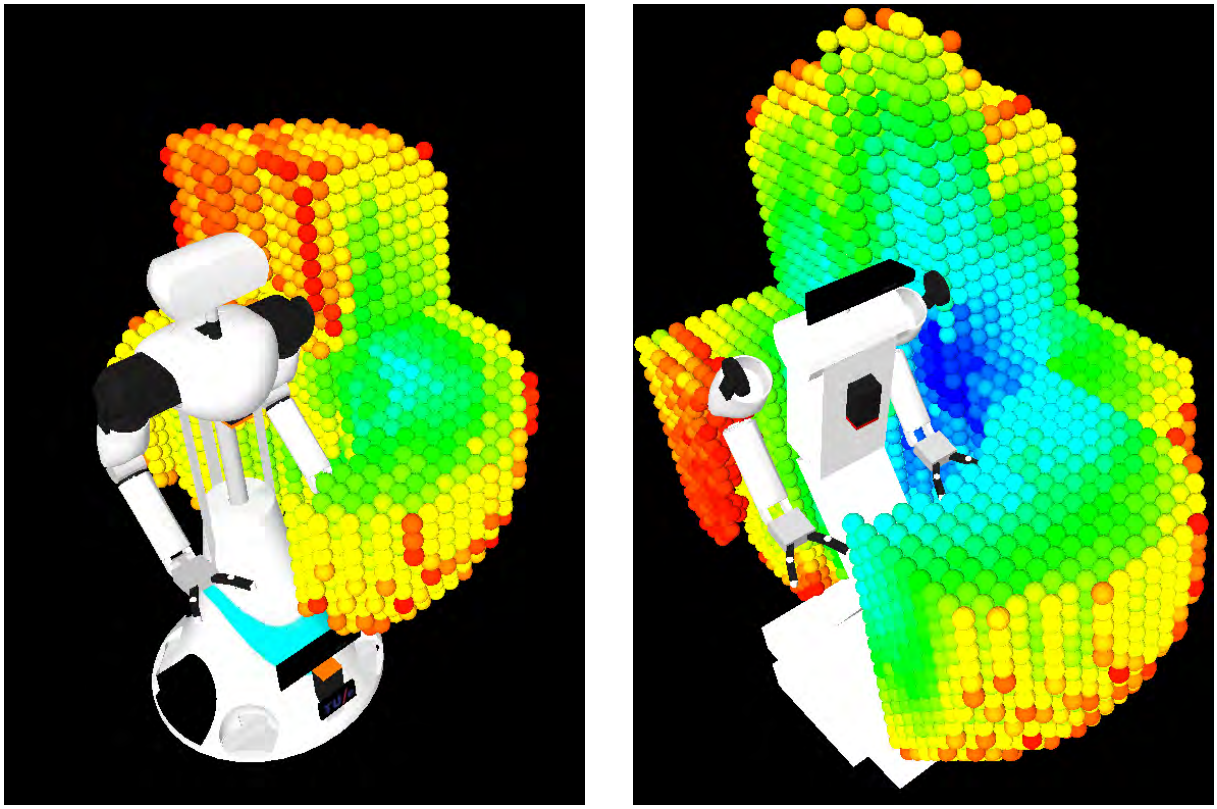


Figure 2.18: Percentage of the points per sphere for which a valid IK solution is found.



(a) AMIGO (b) SERGIO

Figure 2.19: Overview of the reachability of the left arm including torso movements. The TCP is located in the wrist. Obviously, the reachability of SERGIO is much larger than that of AMIGO.

In Figures 2.20a and 2.20b it can be seen that contrary to AMIGO, SERGIO is able to grasp objects from low tables or even from the floor. Note again that for this analysis the TCP is located in the wrist, so the gripper can reach a bit further in front of the robot.

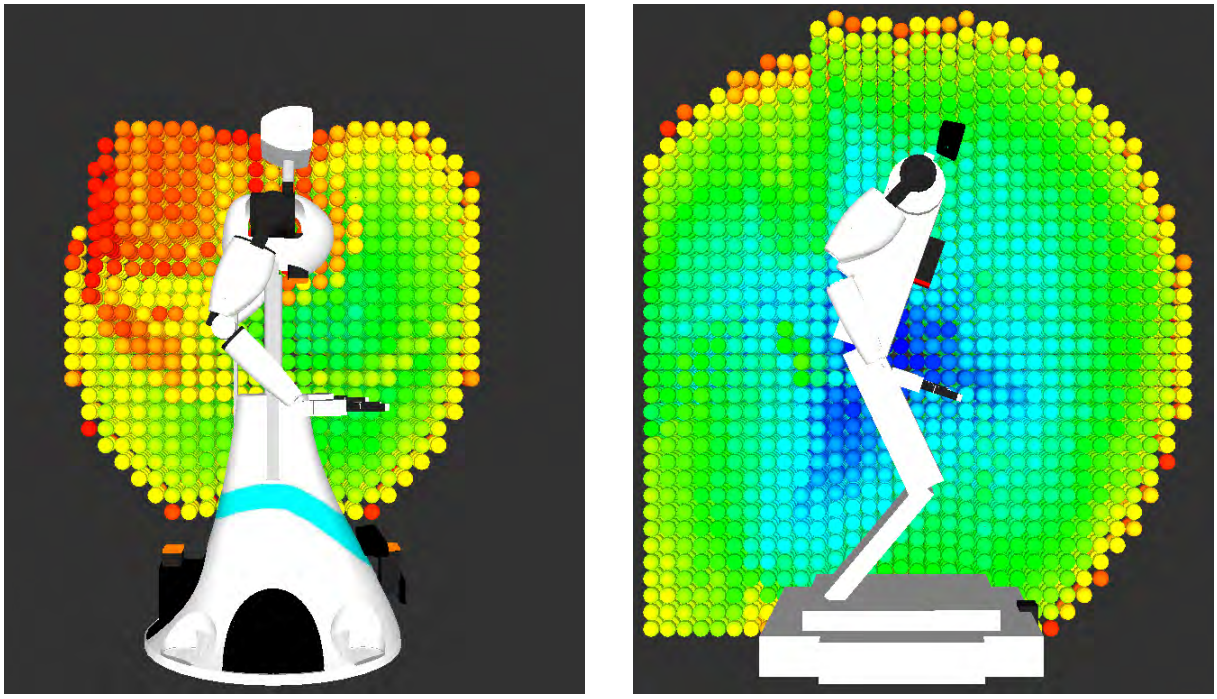
Furthermore, SERGIO is able to grasp objects that are positioned further from the edge of a table, as is confirmed by Figure 2.21a and 2.21b. Compared to AMIGO, SERGIO can grasp objects that are positioned up to 20 cm further from a table edge.

2.8 Conclusion & Recommendations

We now have a state-of-the-art completely *open hardware* design of a service robot. As was shown in Section 2.7.1, a Mecanum wheel platform with independent wheel suspension is robust enough for a service robot. SERGIO is able to drive accurately in a straight line, even on rigid, uneven surfaces. For example, at a forward velocity of $v_x = 0.75$ m/s, the velocity error in y-direction of AMIGO is $e_{vy}^{\max} = 0.28$ m/s, $e_{vy}^{\mathrm{RMS}} = 0.077$ m/s while for SERGIO this is only $e_{vy}^{\max} = 0.025$ m/s, $e_{vy}^{\mathrm{RMS}} = 0.009$ m/s. This also holds for other velocities and when driving sideways. Small thresholds such as doorsteps are also not a problem, even when approached not exactly perpendicular. The ground clearance of 17 mm is sufficient in practice, as these robots are supposed to operate in wheelchair-accessible areas.

The torso combining a nearly vertical DoF with a rotation allows the robot to pick up objects located further from the edge of a table and from the ground, as was shown in Section 2.7.2.

Building the first of two arms as introduced in Section 2.4.2 has just been completed. Once



(a) AMIGO (b) SERGIO

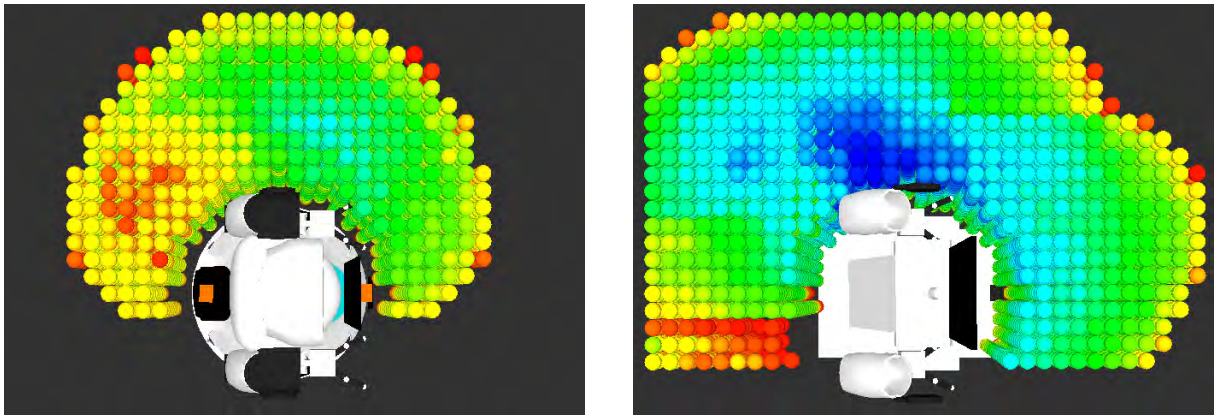
Figure 2.20: Side view of the reachability of the left arm including torso movements. The TCP in the wrist of SERGIO can reach much lower and further forward than AMIGO.

the motion controllers of this arm are implemented, its performance can be thoroughly evaluated. Nevertheless, due to the improvements the performance with respect to the current manipulators is expected to improve significantly.

The modularity of the robot has greatly improved serviceability. Assembling and separating base and upper body can be performed in a minute and a similar mounting time is expected for the arms.

As future work, our existing software stack needs to be updated in order to get SERGIO to perform useful tasks. Due to the differences between AMIGO and SERGIO, this is challenge because it puts additional demands on the generality and configurability of the software. Among the challenges that are expected are navigation, due to the large and rectangular footprint, whole-body control due to the ball screw drives and four-bar mechanism and camera localization, due to the rocking motion of the robot that is introduced by the compliant wheel suspension. When SERGIO is able to perform navigation and pick-and-place tasks, it is possible to thoroughly assess its performance in practice.

Most structural parts of the robot are inexpensive box structures. Nevertheless, the base, torso and manipulators still contain a number of expensive machined parts, which need further engineering to lower the manufacturing bill of materials. Finally, the design of SERGIO is now ready to be released, *i.e.*, to make all CAD drawings, electrical schemes, part lists etc. available on the ROP wiki.



(a) AMIGO (b) SERGIO

Figure 2.21: Top view of the reachability of the left arm including torso movements. Here, the increased reach in forward direction of SERGIO with respect to AMIGO is also clearly visible.

Chapter 3

A recipe to select a motion planner for navigation tasks of mobile robots

A motion planner for mobile robots is commonly built out of a number of different algorithms that solve the two steps of motion planning: representing the robot and its environment and searching a path through the represented environment. However, the available literature on motion planning lacks a generic methodology to arrive at a combination of representations and search algorithm classes for a practical application. This chapter[1] *presents a recipe to select appropriate algorithm classes that solve both steps of motion planning and to select a suitable approach to combine those algorithm classes. The recipe is verified by comparing its outcome to three different motion planners that have been successfully applied on robots in practice.*

3.1 Introduction

Motion planning considers the planning of a path for a robot in an environment that can contain dynamic and static obstacles and is generally characterized by uncertainty. A motion planner can serve three different goals: mapping, covering and navigation. The focus of this chapter is on motion planning for mobile robots moving on the ground plane with navigation as a goal. Motion planning typically consists of two steps (Hwang and Ahuja, 1992), namely:

1. representing the robot and its environment
2. searching a path in the represented environment

These representations and search algorithms are abundant in literature (Latombe, 1990; Hwang and Ahuja, 1992; Choset, 2005; LaValle, 2006). Furthermore, as a single algorithm generally does not suffice for a practical application, planners consist of combinations of algorithms in order to satisfy the designer's requirements (Kant and Zucker, 1986; Latombe, 1990; Brock and Khatib, 1999; Goerzen et al, 2010). Because combinations of possibly approximate algorithms are deemed necessary in practice, various approaches have been presented in literature (Latombe, 1990; LaValle, 2006; Goerzen et al, 2010).

However, the available literature lacks a generic methodology to arrive at a combination of representations and search algorithms that can handle all requirements. The selection of

[1] This chapter is conditionally accepted for publication as an article in IEEE Robotics & Automation Magazine: Lunenburg et al (2015c)

an appropriate combination of representations and search algorithms depends on i) the desired behavior as specified by the designer, ii) the dynamics and the kinematics of the robot at hand, iii) the dynamics of the environment at hand, and iv) the presence of uncertainty. Typically, these conditions are translated into seven requirements, serving as a basis for the selection of these algorithms:

- *Completeness*: the guarantee that either a solution is returned if one exists or failure is returned if no solution exists. Completeness commonly only refers to search algorithms but in fact also concerns the representations as poor design choices for the representation may prohibit any search algorithm from returning a solution even though one exists.
- *Optimality*: a solution can be optimal according to criteria such as distance, time, energy use, etc.
- *Correctness*: the guarantee that if a solution is returned, this will lead the robot to its goal.
- *Dealing with kinodynamic constraints*: a robot has dynamic constraints, *e.g.*, a limited acceleration, and can also be kinematically constrained, *e.g.*, a car cannot move sideways.
- *Robustness against a dynamic environment*: the environment changes if obstacles can move.
- *Robustness against uncertainty*: the representation of the robot and obstacles generally is uncertain and the environment can be (partially) unknown.
- *Computational complexity*: a measure of memory usage and time to acquire a solution. Keeping the computational complexity tractable usually comes at the expense of one or more of the previously mentioned criteria.

In practice, a motion planner is typically desired to be *complete* and to return an *optimal* and *correct* solution in *real-time*. However, there exists no motion planner able to achieve all these requirements while satisfying the *robot constraints* in presence of *moving obstacles* and *uncertainty* (Canny, 1988). Therefore, the motion planning problem at hand is often approximated by a simplified but still realistic version (Latombe, 1990), which introduces a trade-off between the various requirements.

The small amount of literature that discusses combinations of representations and search algorithms is not generic. Either only a single step of the motion planning problem, *i.e.*, either the representation or the search algorithm, is discussed or, in more cases, not all requirements are discussed as some are considered not relevant for a specific robot or goal. In Ferguson et al (2005) some combinations of algorithms are discussed that, given a specific representation of the environment, find a path. The work in reference Giesbrecht (2004) gives an overview of algorithms that represent the environment and algorithms that find a path and discusses combinations of both. How the combined algorithms can deal with robot constraints is not discussed. There is also an overview of combinations of algorithms that find a path for robots with certain kinematic constraints (Laumond et al, 1998). It does however not consider how to deal with moving obstacles or a partially known environment. Also, motion planners are compared that combine multiple algorithms but with a focus on unmanned aerial vehicles (UAVs) (Goerzen et al, 2010; Dadkhah and Mettler, 2012) instead of mobile robots moving only on the ground plane. Moreover, this does not consider how combined algorithms provide robustness against a dynamic environment. Next, there is an overview that discusses planners that have 'covering' as a specific goal (Choset, 2001). These planners assume a mostly static environment and thereby exclude robustness against a dynamic environment. Also, robot constraints are not

taken into account. There are examples of combinations of multiple algorithms that make a well-performing motion planner in practice for a specific mobile robot (Thrun et al, 1999; Brock and Khatib, 1999; Stachniss and Burgard, 2002; Hsu et al, 2002; van den Berg et al, 2006; Dolgov et al, 2008; Marder-Eppstein et al, 2010). Such motion planners give insight in the choice for a combination of algorithms with respect to predefined requirements, but they do not give generic directions for possible combinations.

This chapter contributes by presenting a recipe for the selection of a combination of motion planning algorithms for navigation of mobile robots. This recipe consists of three parts: In the first part (Section 3.2), an appropriate approach to combine representations and search algorithm classes is selected. In the following two parts a representation class (Section 3.3) and search algorithm class (Section 3.4) to accomplish the two steps of motion planning are selected. This recipe is intended for either novices in robotics or for experts on one of the many other topics in robotics who require navigation on their mobile robot for a practical application. Carefully considering this recipe and the discussed design criteria can be a first step in designing a navigation system for a mobile robot.

3.2 Selecting an approach to combine algorithm classes

In this section the available approaches to combine motion planning algorithms are discussed, namely: a coupled, a decoupled, a hierarchical, and a reactive approach. Selecting an appropriate approach to combine motion planning algorithms is of paramount importance as this 'system design' serves as a guideline for the subsequent design choices in Sections 3.3 and 3.4. Many of the design decisions made in this stage concern the available environment models, *e.g.*, the availability and uncertainty in these models and the presence and predictability of moving obstacles are already discussed in this section. Depending on the requirements of the application these approaches have to be combined with additional features, such as re-planning or a bounded uncertainty region. A recipe, visualized in Figure 3.1, is presented to select an approach based on the requirements that are introduced in Section 3.1. This section and the forthcoming sections primarily adopt the terminology as used in Latombe (1990); Hwang and Ahuja (1992).

3.2.1 Approaches to combine algorithm classes

First of all, in *trajectory planning* or a *coupled approach* the robot's kinodynamic constraints are directly satisfied in the search for a solution (Donald et al, 1993). These methods search for a solution in the state space $\mathcal{S}$ (Donald et al, 1993). A state lattice is an example of a representation for planning in the state space satisfying the robot's kinodynamic constraints (LaValle, 2006; Pivtoraiko et al, 2009). Methods that search directly in the state-time space $\mathcal{ST}$ are called state$\times$time search methods (Fraichard, 1998). The notion of time allows the planner to plan motions that are optimal in the sense of time in the presence of moving obstacles. A coupled approach is generally computationally expensive (Canny, 1988). To keep computations tractable, such an approach requires algorithms that exploit some form of approximation to arrive at a solution. The completeness of these approximation algorithms is difficult and impractical to prove (Goerzen et al, 2010).

A second approach is to decouple the search for a solution by first searching for a partial solution in the configuration space $\mathcal{C}$. This *decoupled approach*, also called *path planning*, generally consists of the following steps (LaValle, 2006):

- Path planning: obtain a collision-free path in $\mathcal{C}$.

- Path transformation: ensure that a path satisfies any kinematic constraints in $\mathcal{S}$ (Laumond et al, 1998; Donald et al, 1993).

- Trajectory planning: use a timing function to transform a path into a trajectory that satisfies dynamic constraints.

If this sequence is required to deal with moving obstacles, decoupling into a path planning part and a motion timing part is also possible (Kant and Zucker, 1986). This approach follows the first two steps of the decoupled trajectory planning approach, but the trajectory that is formed in the third step also accounts for moving obstacles. Recent approaches to create trajectories from a path are CHOMP (Ratliff et al, 2009) and STOMP (Kalakrishnan et al, 2011), which use optimization techniques to minimize some cost function while satisfying the kinodynamic constraints of the robot. Planning in the configuration space $\mathcal{C}$ is computationally less expensive than in the state space $\mathcal{S}$ due to the lower dimension. The inherent consequence is a negative influence on the completeness and optimality of the approach (Latombe, 1990). For example, a path obtained in $\mathcal{C}$ with sharp turns might be unfeasible for a non-holonomically constrained car. Also, a single path, obtained by a search in $\mathcal{C}$, can be transformed into a time-optimal trajectory but this trajectory is not guaranteed to be globally time-optimal.

Third, a *hierarchical approach* acquires a solution in two or more consecutive plans (Giesbrecht, 2004; Goerzen et al, 2010). This approach uses a planner to find a solution to the goal, using the configuration space or the full-dimensional solution space, followed up by a planner that searches for a solution in the vicinity of the robot using the full-dimensional solution space. This is also known as global motion or path planning and local motion planning. Similar to the decoupled approach this is a decoupling that sacrifices optimality, but it is not necessarily incomplete (Zhang et al, 2012). A hierarchical approach is also possible in $\mathcal{C}$. One could first plan a global path that only considers position and subsequently use a local planner that also takes the orientation into account and hence plans in a higher dimension.

The fourth approach is a *reactive approach* or sensor-based approach. This approach is based on a class of algorithms that are also known as obstacle avoidance algorithms. Rather than searching for a path or trajectory on a certain representation, these compute an instantaneous motion that avoids collision based on the latest sensor information. This approach does not need a representation in the form of a metric map as opposed to the previously discussed approaches. They can be divided into four classes: potential field (Khatib, 1986), first order methods or velocity obstacle methods (Fiorini and Shiller, 1998), receding horizon control (Fox et al, 1997; Ogren and Leonard, 2005) and the (global) nearness diagram (Minguez and Montano, 2000; Minguez et al, 2001). Typically, these methods only compute a desired direction of motion and hence the amount of planning is reduced to a minimum. However, as such an approach does not consider the planning problem in the entire world space, there is no guarantee that the resulting trajectory will lead to the goal. These local minima are a well-known drawback of reactive methods. Furthermore, the resulting trajectory is hardly ever optimal. Therefore, a reactive planner is typically used in combination with a global planner and hence it can be considered as a special form of a hierarchical approach. Besides the selection of a global planner and a reactive planner, the integration of these modules is important. One should, *e.g.*, consider how the system should react if either of the planners cannot compute a solution, whether a global plan is computed every sample, etc. Examples of approaches combining global planners with reactive methods can be found in, *e.g.*, Thrun et al (1999); Brock and Khatib (1999); Maravall et al (2000); Stachniss and Burgard (2002); Philippsen and Siegwart (2003); Minguez et al (2004); Marder-Eppstein et al (2010).

3.2.2 Recipe to select an approach to combine algorithm classes

The following design criteria, that are numbered A1 through A7, form a recipe to select a suitable approach to combine motion planning algorithms. The criteria are based on the requirements that are introduced in Section 3.1. The recipe is visualized in Figure 3.1.

A1) Is the environment known? In an environment that is not completely known, a plan can become unfeasible as sensor data is acquired during execution and thus a new plan is required. This requires *re-planning.* A re-plan can be acquired by recomputing the entire representation of the solution space, discussed in Section 3.3.2, or by updating the representation and performing a new search as discussed in Section 3.4.2.

A2) Is there uncertainty in the obstacle representation? Uncertainty in the representation of obstacles, *e.g.*, uncertainty due to imperfect sensors, can assumed to be contained in a *bounded uncertainty region* (Latombe, 1990). In practice, a motion planner gains robustness by inflating the obstacle representation with a distance that is equal to the present position uncertainty of obstacles. This robustness, however, comes at the expense of completeness: if obstacles are inflated too much the planner might not be able to find a solution anymore.

A3) Does the environment contain moving obstacles? To be complete and to get an optimal solution in the presence of moving obstacles it is necessary to represent their trajectories with a time dimension in the solution space. Nevertheless, this significantly increases computational complexity. Re-planning is also possible to deal with a changing environment but this will result in suboptimal plans and the planner might not be complete. A further alternative to re-planning is to update or modify the existing plan. An example of this approach are elastic bands (Quinlan and Khatib, 1993).

A4) Can the motion of obstacles be predicted? If obstacles are predictable, *i.e.*, have a known trajectory, it is necessary to add a time dimension to the solution space, yielding a state-time space $\mathcal{ST}$, in order to be complete and to ensure a time-optimal solution. A trajectory can be modeled using a prediction model if obstacles have no exactly known trajectory. For example, an obstacle position can be extrapolated in time using a maximum acceleration model. In general, one can say that the effectiveness of including a prediction model depends on the amount of uncertainty in the prediction. As the inflation of the obstacle representation increases, the number of possible solutions decreases and the remaining ones will be further from optimality. A time dimension will unnecessarily increase the dimension of the solution space if obstacles are not predictable. Re-planning and using a reactive approach are alternatives to deal with moving obstacles. The inherent consequence is that the solution will not be optimal and the planner is possibly incomplete.

A5) Is a solution in the form of a plan necessary in the full dimension and size of the solution space? A coupled approach searches for a solution in the full dimension of the solution space and therefore it can be complete and return optimal solutions. However, the complexity of a representation is exponential with the number of dimensions of the solution space. A coupled approach can thus be computationally too complex to achieve a solution within a designer's desired time constraint. The complexity can be negotiated using a hierarchical, reactive or decoupled approach but this is generally at the cost of completeness and optimality.

A6) Is a solution in the form of a plan necessary and possible within a certain time constraint? Achieving a solution in the form of a plan that avoids collision can be too time-consuming. An environment can change so rapidly due to moving obstacles and uncertainty that a plan can become unfeasible before the robot is able to even execute it. In such a case planning becomes less relevant and a reactive approach is necessary. As a reactive approach typically only considers the latest sensor information, it is not complete and its solution is not optimal. One must also question whether an explicit plan is actually necessary. It can be sufficient to apply a reactive approach that uses the structure of the environment to control the robot to its desired goal pose

instead of precomputing the entire motion.

A7) Local planning possible? A hierarchical approach is generally preferred as a local planner in this approach still obtains a solution in the full solution space. Therefore, there is a better chance of finding a solution and this solution is generally closer to optimality than in case of a decoupled approach. Despite a lower computational complexity a hierarchical approach might still not be able to return a solution in time. A decoupled approach, that is generally of a lower computational complexity, could then be used.

3.3 Selecting a representation class

Given an approach to combine motion planning algorithms (see Section 3.2) this section provides a recipe to select a suitable class of representations. This section discusses the robot representation and introduces the available classes to represent the environment, namely: an exact representation, an approximate representation, and a topological representation. The recipe is visualized in Figure 3.2.

3.3.1 Representation classes

In order to do motion planning, the robot, the workspace and the obstacles therein need to be represented. The *robot representation* is determined by the configuration parameters that define the shape of the robot. The mobile robots considered in this chapter are rigid bodies moving on a plane. Hence, the workspace $\mathcal{W}$ of the robot is a two-dimensional Euclidian space $\mathbb{R}^2$. The pose of the robot in this workspace can be uniquely defined by two position coordinates and one orientation coordinate. As a result, the configuration space $\mathcal{C}$ in which planning is performed typically has three dimensions. The configuration space can be extended to describe the rate of change of configurations, thereby forming a state space $\mathcal{S}$. To solve a motion planning problem a robot configuration is typically represented as a point in the configuration space. Obstacles are represented as configurations for which the robot is in collision.

The concept of the configuration space transforms the problem of planning the motion of an arbitrarily shaped robot into the problem of planning a sequence of configurations or states that move the robot towards the goal. To be able to search for a sequence of configurations or states in the solution space, the solution space must be represented in such a way that it connects configurations or states, *i.e.*, the connectivity of the free space must be captured.

An *explicit representation* computes the obstacle region $\mathcal{C}_{\text{obs}}$ for all obstacles. *Implicit representations* avoid this explicit construction and instead rely on a collision checking model to verify whether a configuration is free or not (Hsu et al, 2002; LaValle, 2006). In LaValle (2006), this is called sampling-based motion planning. Note that in this terminology, 'sampling' does *not* necessarily imply stochastic sampling but refers to the configurations for which a collision check is performed. Two sampling methods exist: deterministic and stochastic or random (LaValle et al, 2004).

Besides being explicit or implicit, a representation can be either exact or approximate. An *exact representation* requires obstacles to be described by a geometric model, *e.g.*, a two-dimensional polygon or a three-dimensional polyhedron. An exact representation generally captures the connectivity of the free space in a network of curves, called a *roadmap* (Latombe, 1990). An overview of algorithms that compute an exact representation is given in Latombe (1990); Goodman and O'Rourke (2004); LaValle (2006). Most exact representations are limited to the configuration space $\mathcal{C}$ as in the presence of kinodynamic constraints they lose completeness (LaValle, 2006). On the other hand, approximate representations do allow integration of kinodynamic constraints and can thus be used to represent a state space $\mathcal{S}$.

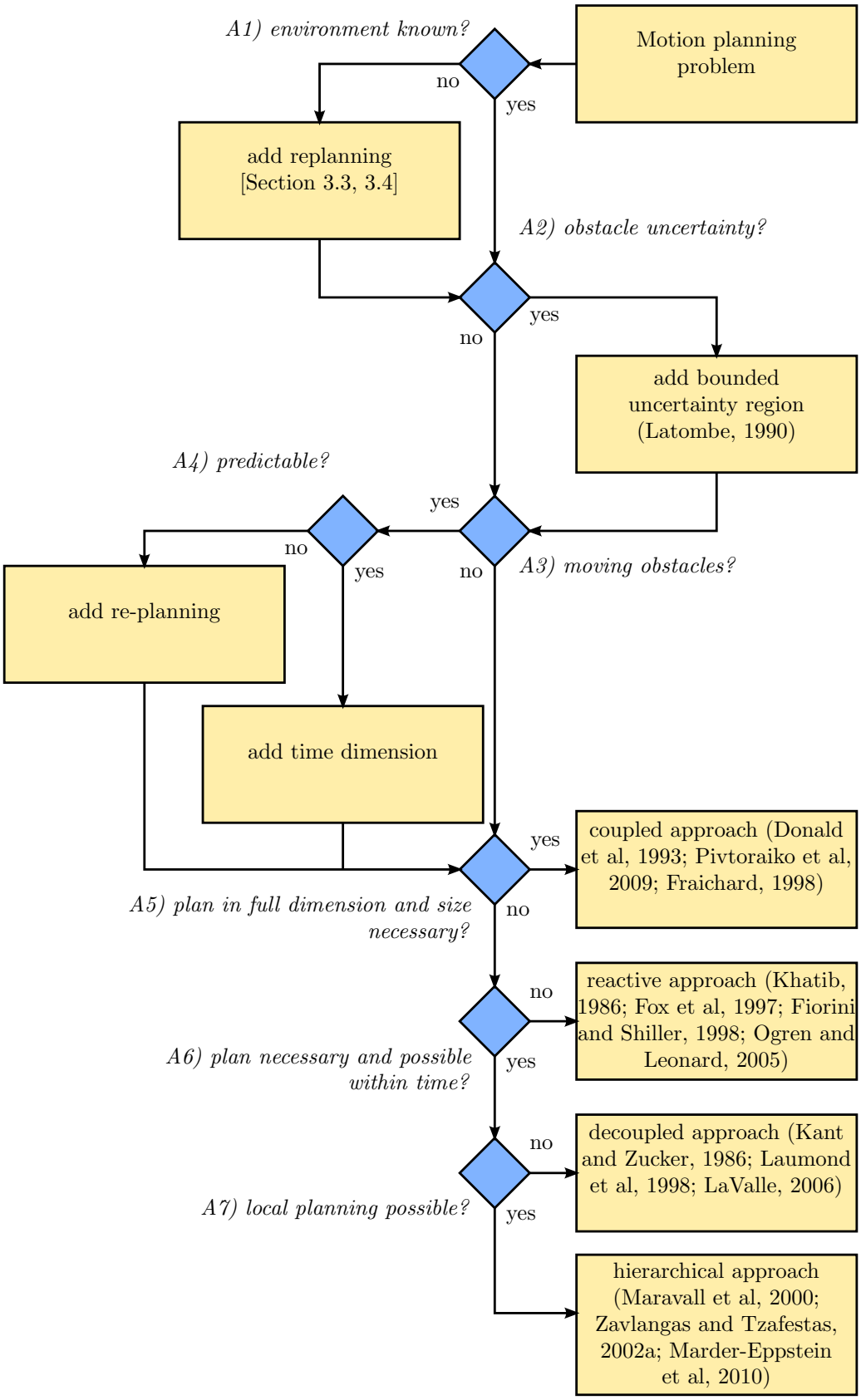


Figure 3.1: A recipe resulting in an appropriate approach to combine motion planning algorithms. The numbers A1 - A7 correspond to the design criteria posed in Section 3.2.2.

An *approximate representation* represents the free space and occupied space as collections of cells which all have the same simple, pre-specified shape, *e.g.*, a hyperrectangle. Due to the approximate nature only a weaker notion of completeness and optimality can be achieved, *i.e.*, resolution completeness and optimality (LaValle, 2006), but it is able to deal with arbitrary obstacle shapes.

A *deterministic sampling method* uses a grid that is placed over the robot's solution space. It is resolution complete and optimal, *i.e.*, the completeness and optimality depend on the resolution of the grid. The size of the grid grows exponentially with the dimension of the solution space (Lindemann and LaValle, 2005), hence deterministic sampling is typically used for low-dimensional solution spaces. If desired, this low-dimensionality enables the possibility to make the representation explicit, *e.g.*, to inflate $\mathcal{C}_{\text{obs}}$ with the robot radius to obtain an explicit costmap. A *stochastic sampling method* relies on random sampling. A configuration is evaluated for collisions as it is sampled, and hence these methods are always implicit. An example of a representation based on random sampling is a probabilistic roadmap (PRM) (Kavraki et al, 1996). There exist various variants of PRMs that are probabilistically complete, *i.e.*, the probability that the planner returns a solutions, if one exists, increases to one as the number of samples approaches infinity. More recent developments such as PRM* also approach asymptotic optimality (Karaman and Frazzoli, 2011).

A *single-query planner* or *incremental planner* such as a rapidly-exploring random tree (RRT) (LaValle and Kuffner Jr, 2001), an expansive-space tree (EST) (Hsu et al, 2002) and derivatives thereof also relies on random sampling. However, instead of computing a connectivity graph and subsequently searching this, it interweaves representation and search. Similar to PRM*, recent developments such as RRG, RRT* (Karaman and Frazzoli, 2011) and RRT$^{\#}$ (Arslan and Tsiotras, 2013) also approach probabilistic completeness and asymptotic optimality.

A *topological representation* is a more abstract representation that describes relationships among features of the environment without an absolute reference system (Zavlangas and Tzafestas, 2002a; Kuipers et al, 2004). Such environmental features can be landmarks or identifiable locations such as a room of a house or an intersection of roads. A topological map, that results from a topological representation, has the advantage of being more compact and more stable with respect to sensor noise and uncertainty in a priori information on the world space than a metric map that results from an exact or approximate representation. The topological map does not provide a framework to control the robot, since it does not take into account the dynamics of the robot and cannot capture moving obstacles. Thereto, another planner is required in a hierarchical approach, as mentioned in Section 3.2.1.

In case of the reactive planners in Section 3.2.1 and the single-query planners in this section, representation and search are tightly coupled and it can therefore be difficult or impossible to make a clear distinction between these two steps, as is done in reference Hwang and Ahuja (1992). However, for many other motion planners this distinction can be made and since this is beneficial to the clarity of the motion planning recipe, it will be maintained throughout this chapter. Although one might argue that single-query planners are search algorithms rather than representations, they are already introduced in this section to make sure single-query planners are not discarded in the recipe before proceeding to the selection of search algorithm classes.

3.3.2 Recipe to select a representation class

Given an approach to combine motion planning algorithms, the following design criteria, that are numbered R1 through R6, form a recipe to select a suitable class of representations. The recipe is visualized in Figure 3.2.

R1) What configuration variables are necessary to describe the robot? Three configuration variables are required to uniquely define a robot's position and orientation on the ground plane.

Not all variables might be necessary to describe the robot and hence the computational complexity of finding a solution in the resulting configuration space $\mathcal{C}$ can be reduced. For example, it is common practice to only consider the position of the robot when planning, thereby omitting its orientation and limiting the dimension of $\mathcal{C}$ (Latombe, 1990). This can, however, result in incorrectness, *e.g.*, a robot with an elongated shape that is described by a circular shape may not fit through a narrow passage. The greater the approximation of the shape is, the more likely the motion planner will be incorrect. This also depends on the smallest passage width, or generally, the structure of the environment.

R2) What is the solution space? The selection of an approach to combine motion planning algorithms in Section 3.2 results in a solution space that needs to be represented and searched for a solution. An exact representation is an option if the solution space is $\mathcal{C}$. An approximate method is typically used if the solution space is $\mathcal{S}$, as mentioned in Section 3.3.1. A topological representation can be used if the environment can be represented as a collection of connected locations or features. A topological representation reduces computational complexity as it is more compact than a metrical representation such as an exact or approximate representation. A planner that uses a topological representation requires a succeeding planner that uses a metrical representation as a robot is typically controlled in a metrical frame.

R3) Can obstacles be described by a geometric model? An exact representation method is complete and can provide an optimal solution if the solution space is $\mathcal{C}$ and obstacles can by modeled by a geometric model (Hwang and Ahuja, 1992; Latombe, 1990). Remark that this requires complete knowledge of obstacle shapes, thus any uncertainty, unless bounded, is prohibitive. Otherwise, an approximate representation is preferred.

R4) Can a solution be obtained in time with dense sampling? A deterministic method samples a space using a regular grid. An approach is generally close to completeness and its solutions close to optimality when dense sampling is used. The computational complexity of a deterministic sampling method can be too high to obtain a solution within a certain time requirement. Therefore, this method is typically limited to low-dimensional solution spaces as the size of the grid grows exponentially with the dimension of the solution space (Lindemann and LaValle, 2005). A multi-resolution deterministic sampling representation, such as a quadtree (2D) (Soucy and Payeur, 2004) or an octree (3D) (Hornung et al, 2013), is an option if the computational complexity is limiting. As this representation can represent volumes with a lower resolution it is typically more efficient for planning purposes than a regular, deterministic representation. A random sampling method is even of a lower complexity as it generally requires less samples to represent the world space than a deterministic sampling method. As a result it can be applied to higher dimensional solution spaces.

R5) Is the environment cluttered? In a cluttered environment care needs to be taken to prevent incompleteness when using an approximate representation method. The resolution of sampling must be dense enough to allow the robot to pass through the smallest clearance between obstacles. For random sampling methods in particular it is important that the used sampling strategy ensures dense enough sampling in small passages. This difficulty that sampling based planners face, known as the 'narrow passage' problem (Hsu et al, 1998), can be overcome using different sampling strategies (Choset, 2005; Lindemann and LaValle, 2005).

R6) Can the planner be used for multiple queries? It can be beneficial to use a roadmap representation if a motion planner must generate multiple solutions within the same world space. This can be an explicit representation, such as a Voronoi diagram or visibility graph (LaValle, 2006), but also an implicit variant, such as a PRM (Kavraki et al, 1996) or PRM* (Karaman and Frazzoli, 2011). A roadmap only needs to be computed once and can subsequently be used for multiple searches, hence the computational complexity of the representation is limited to a pre-computation step. A roadmap can be recomputed or updated (van den Berg et al, 2006) if the environment changes. On the other hand, if only one solution is required it is not necessary

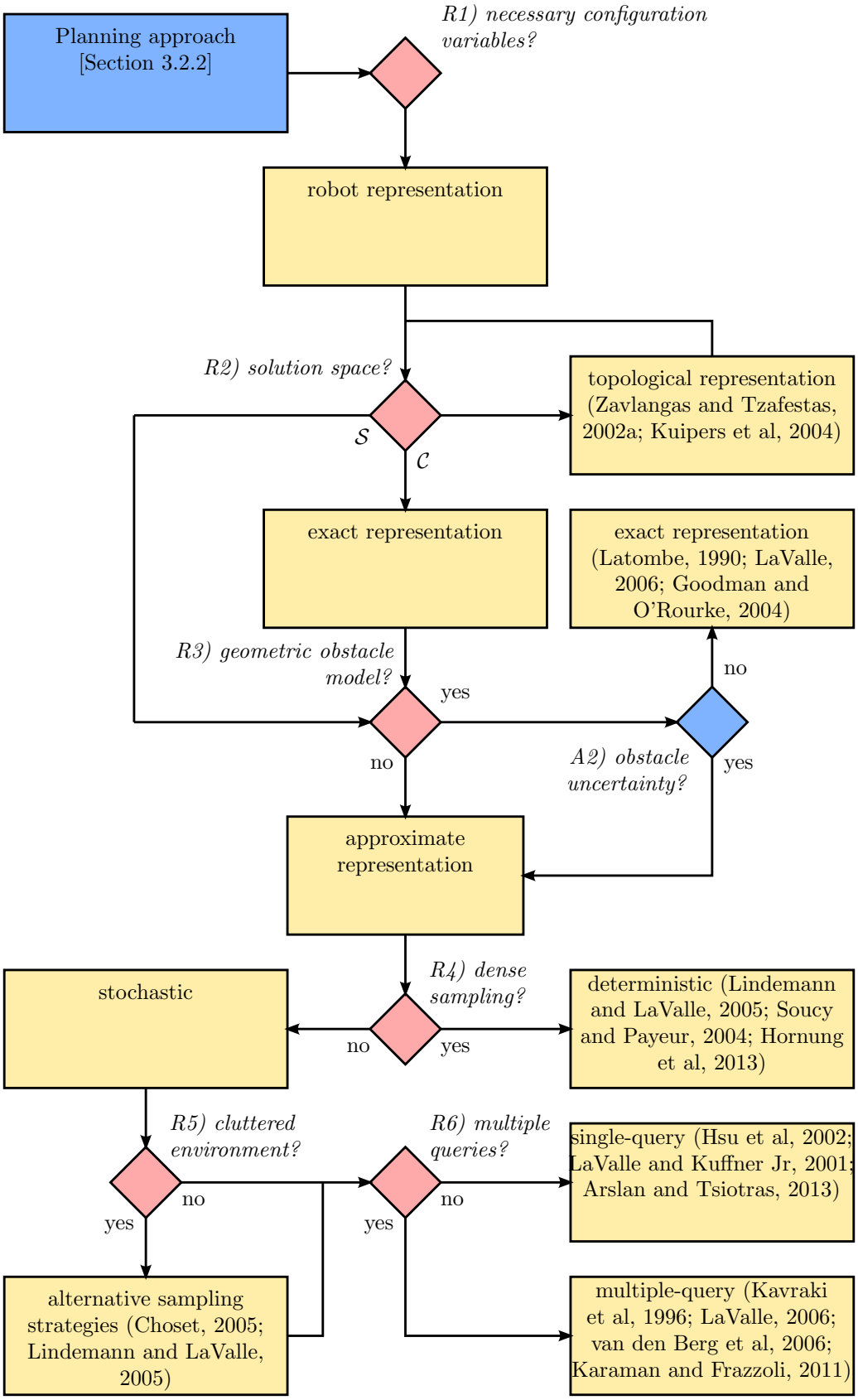


Figure 3.2: A recipe to arrive at an appropriate class of representations. The numbers R1 - R6 correspond to design criteria posed in Section 3.3.2. Number A2 corresponds to a criterium posed in Section 3.2.2.

to generate a complete representation of the world space. Representation and searching could then be interleaved using a single-query planner such as an RRT (LaValle and Kuffner Jr, 2001), EST (Hsu et al, 2002), RRT* (Karaman and Frazzoli, 2011) or RRT$^{\#}$ (Arslan and Tsiotras, 2013). Note that these algorithms do not require a search method as is introduced in Section 3.4.

3.4 Selecting a search algorithm class

The previous two sections discuss available approaches to combine motion planning algorithms and available classes of representation algorithms. In this section two available classes of search algorithms are discussed, namely: uninformed and informed. Depending on the designer's requirements these classes of algorithms can be combined with additional features, such as incremental re-planning and anytime behavior. A recipe, visualized in Figure 3.3, is provided to select a suitable class of search algorithms given an approach to combine motion planning algorithms (see Section 3.2).

3.4.1 Search algorithm classes

Most representations transform the continuous problem of finding a solution in the free space into a discrete problem of searching a graph with nodes that represent the configurations or states of the robot. Two types of search algorithms can be applied: uninformed and informed.

An *uninformed search algorithm* moves through the graph without any preference for the location of the goal. Well-known algorithms are a breadth-first search and a depth-first search. An optimal path with respect to a specified cost criterion can be achieved using Dijkstra's algorithm (Russell and Norvig, 2010).

If the direction of the goal node is known, the search can be directed, *i.e.*, can be biased toward the goal. A search algorithm that includes information about the goal is called an *informed search algorithm*. To this cause a heuristic is formulated, i.e, a function of nodes that hypothesizes a cost towards the goal node. The choice for the next node to explore is then based on this heuristic cost, *e.g.*, defined as the Euclidean distance towards the goal. The most well-known algorithm is the A* algorithm. Similar to uninformed algorithms, A* also returns the optimal path, but the search typically is of lower computational complexity, *i.e.*, it explores less space and thus returns a solution faster (Russell and Norvig, 2010). To ensure completeness and optimality, a heuristic must satisfy two properties: consistency and admissibility (Russell and Norvig, 2010). A heuristic that satisfies these two properties never overestimates the cost of reaching the goal.

3.4.2 Recipe to select a search algorithm class

Given an approach to combine motion planning algorithms, the following design criteria, that are numbered S1 through S3, form a recipe to select a suitable class of search algorithms. The recipe is visualized in Figure 3.3.

S1) Can a heuristic be defined to speed up the search? The use of a heuristic to the goal can reduce the complexity of a search (Ferguson et al, 2005). An informed search is preferred over an uninformed search if a consistent and admissible heuristic (Russell and Norvig, 2010) can be defined. Especially for a search in a high-dimensional solution space a heuristic can greatly reduce the complexity. A heuristic is then typically informed with more than just the distance to the goal by using a search in a lower dimension as a heuristic to inform a search in the full dimension of the solution space (Likhachev et al, 2008; Phillips and Likhachev, 2011). For example, a planner that does a search in the state space $\mathcal{S}$ can be informed with a heuristic that

Table 3.1: Design requirements and the corresponding selection criteria of the recipe.

Requirement	Design criterion
Environment	
Known/unknown	A1, A6
Uncertainty in obstacle representation	A2, A4, A6
Moving obstacles	A3, A4, A6, S2
Robot model	
Correctness	R1
Kinodynamic constraints	A4, A5, A6, A7, R2
Application	
Completeness	A2, A4, A5, R1, R3, R4, R5
Optimality	A3, A4, A5, R3, R4, S3
Computational Complexity	A3, A4, A5, A6, A7, R4, R6, S1

does a search in the configuration space $\mathcal{C}$. This heuristic search ignores robot constraints and moving obstacles, but it can reduce the search space of $\mathcal{S}$ substantially and thus it can return a solution faster. The reduce in space depends on whether the heuristic solution is expected to be close to the optimal solution.

The single-query planners introduced in Section 3.3 can also benefit from adding a goal bias or other heuristic to the sampling algorithm. Examples of such approaches can be found in, *e.g.*, Qureshi et al (2013) and Akgun and Stilman (2011).

S2) Does the environment change significantly between consecutive searches? A plan can become unfeasible during execution because of appearing or moving obstacles. Thereto, a motion planner can adopt a re-planning strategy. Re-planning can be performed by computing an entirely new plan. One could also use *incremental re-planning* by using results of previous searches to generate a new plan faster (Koenig et al, 2004). The benefit of taking previous solutions into account depends on the environment. If there is no significant change, a re-plan can be acquired faster, but when results of previous solutions are less useful, it can even be disadvantageous to the task of arriving at a new solution (Ferguson et al, 2005).

S3) Is a solution required at any time? It is desirable to achieve a solution as fast as possible. However, this solution is typically not optimal. An *anytime* search algorithm can return a solution within a certain time constraint and with a certain bound on its sub-optimality (Ferguson et al, 2005). As time allows this bound is tightened and given enough time the optimal solution is returned (Likhachev et al, 2003). An optimal search algorithm typically acquires this solution faster, but an anytime algorithm allows a robot to start executing an initial sub-optimal solution and improve this solution along the way. Examples can be also be found in Karaman and Frazzoli (2011).

This concludes the motion planning recipe. An overview of the requirements and the corresponding design criteria is presented in Table 3.1. This overview shows that several design criteria involve multiple requirements. This clearly indicates the trade-offs that are commonly present when designing a navigation system, *e.g.*, one could try to design a system which returns optimal solutions in the presence of moving obstacles but this would be computationally extremely complex. More of these trade-offs can be derived from this table.

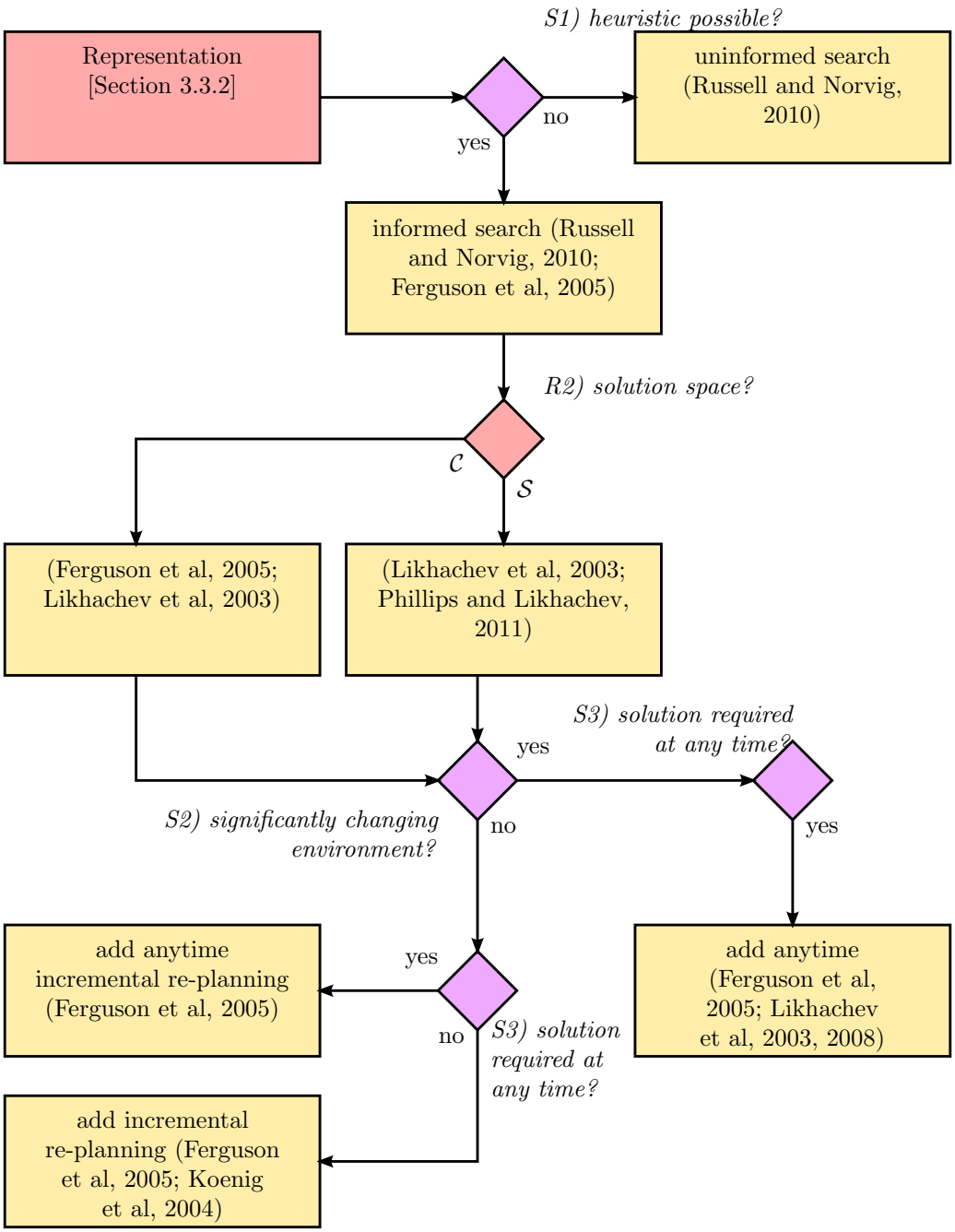


Figure 3.3: A recipe to arrive at an appropriate class of search algorithms. The numbers S1 - S3 correspond to the design criteria posed in Section 3.4.2 and R2 corresponds to a criterium posed in Section 3.3.

3.5 Recipe verification

Three use cases from literature involving the application of a motion planner in practice are evaluated to verify the proposed recipe: i) the PR2 service robot that autonomously navigated a marathon distance of 42.15 km in a real office environment (Marder-Eppstein et al, 2010), ii) the autonomous vehicle Boss that won the DARPA Urban Challenge (Urmson et al, 2008), and iii) a flexible automated guided vehicle (AGV) in partially structured warehouses (Martínez-Barberá and Herrero-Pérez, 2010).

3.5.1 Application of the recipe to use case I

The PR2 service robot has an omni-directional base platform with a top speed of 1 m/s. It navigates in an office environment that is cluttered with a variety of arbitrarily shaped obstacles (Marder-Eppstein et al, 2010).

1) Selecting an approach to combine algorithm classes: The office environment is only partly known so the robot must re-plan as it receives new information about the world space (*A1*). The depth accuracy of the sensors that are used for navigation is 1.5 cm. Hence, the resulting uncertainty must be captured in a bounded uncertainty region (*A2*). The office environment is populated by people, so the environment contains moving obstacles (*A3*). The trajectories of people are unpredictable (*A4*) and the world space is only partially known. Therefore the representation of the world space will change rapidly. As the robot is desired to navigate at a reasonable speed of about 0.5 m/s, a reactive approach is the most suitable approach to avoid collision in real-time (*A5, A6*). A reactive approach requires an obstacle avoidance algorithm and a global planner (see Section 3.2.1).

2) Selecting a representation class: With only 5 cm clearance on both sides when traversing a typical doorway a circular approximation of the PR2 will result in incompleteness. Thus, the robot is best described by a rigid body (*R1*). The solution space consists of multiple rooms and corridors and therefore lends itself for a topological representation. This representation reduces the size of the solution space of the global planner, that is equal to $\mathcal{C}$ (*R2*). An approximate representation method is preferred over an exact representation method as the obstacles are of arbitrary shape and their representation is uncertain (*R3*). The environment is cluttered with small obstacles such as office supplies and thus dense sampling is required to prevent incompleteness. A deterministic sampling method is therefore preferred over a stochastic sampling method (*R4*). A multi-resolution representation is advised to negotiate the computational complexity as an office environment has large open space but also consists of cluttered occupied spaces.

3) Selecting a search algorithm class: The PR2 is supplied with a navigation goal. The direction of this goal can be used as a heuristic to speed up the search (*S1*). In a cluttered office environment a re-plan can be significantly different from an original search and thus recomputing an entire new plan is preferred (*S2*). It is not required that the robot obtains a solution immediately upon a goal request, but it is desirable. Therefore, the search algorithm can be extended with an anytime algorithm (*S3*).

4) Comparison of the recipe's outcome and the use case: Similar to the prescription by the recipe, the PR2 motion planner (Marder-Eppstein et al, 2010) operates with a reactive approach as it uses a global planner and a local obstacle avoidance algorithm (Fox et al, 1997). The local algorithm uses a rigid body description but the robot's shape is approximated by an inscribed circle in the global representation of the world space. This can result in incompleteness as the global planner may produce a path through a narrow passage, *e.g.*, a half opened door, that the local planner can not execute. This is indeed identified as a limitation by the authors (Marder-Eppstein et al, 2010). As the recipe prescribes, this can be prevented if the global

planner would also use a rigid body description of the robot. A motion planner for the PR2 that does this is presented in Hornung et al (2012). The PR2 motion planner does not use a topological representation on top of the global configuration space representation, *e.g.*, as used in Zavlangas and Tzafestas (2002a). This is however identified as future work. Furthermore, the global planner uses an informed search algorithm (A*) like the recipe indicated, but it is not an anytime algorithm.

3.5.2 Application of the recipe to use case II

The autonomous driving vehicle Boss uses two different planners (Urmson et al, 2008): one during nominal on-road driving and one during more unstructured driving, *e.g.*, in a parking lot or during error recovery. The latter planner is used for the verification of the recipe.

1) Selecting an approach to combine algorithm classes:
Re-planning is required as Boss faces unknown spaces as it navigates (A1). Boss' sensors have a limited resolution and thus a bounded uncertainty region is necessary (A2). During navigation the vehicle encounters other vehicles (A3) whose future trajectories are predictable on short-term (A4), hence a time dimension can be added for a time-optimal solution. A solution in the form of a plan in the full-dimensional solution space is necessary to be time-optimal but as vehicles are only predictable on short-term it is only necessary to consider a time dimension on that short-term (A5). A plan is necessary to take into account the kinematic constraints of the vehicle (A6). To achieve a local optimal solution a hierarchical approach is suggested (A7).

2) Selecting a representation class: A vehicle is subject to kinematic constraints that must be taken into account while planning (R1). The velocity also needs to be considered during planning to generate feasible motions in the presence of obstacles, resulting in $\mathcal{S}$ as a solution space (R2). Given the dimension and size of the solution space a stochastic sampling method is suitable (R4). Sampling is required to be dense (R5) especially near the goal configuration as this is typically in a narrow part of the environment such as a parking lot. A single-query planner suits as previous solutions are not reusable (R6).

3) Selecting a search algorithm class: The vehicle is supplied with a navigation goal that can be used as a heuristic (S1). Incrementally updating the solution as new information is acquired is beneficial as a traffic environment in general is not very cluttered (S2). To prevent the vehicle from stopping to calculate a plan it is important that it returns a solution within a certain time constraint. Hence, an incremental, anytime re-planning algorithm is necessary (S3).

4) Comparison of the recipe's outcome and the use case: Boss uses a hierarchical approach (Likhachev et al, 2008), which is also prescribed by the recipe. The approach includes a global planner that searches for a feasible motion plan in 4D (x, y, θ, v) and a local motion planner operating in the same space, that selects a trajectory that follows the global plan. Short, feasible maneuvers are computed off-line to substantially reduce the time to acquire a solution. These maneuvers are concatenated at planning time in a single-query fashion to obtain a motion plan. In the vicinity of the vehicle and the goal, the search is dense, like the recipe indicated, while in other areas the set of possible maneuvers is more coarse. A combination of heuristics is used to further reduce the time to acquire a solution. Similar to the recipe's prescription, an incremental, anytime algorithm is used. In contrast to the recipe, a time dimension is not explicitly incorporated to deal with other moving vehicles. This is solved by marking the space that another vehicle will occupy over a short-term prediction as occupied in a static map. However, this can result in sub-optimal behavior in the presence of moving vehicles, but it returns a solution faster as the search space does not include a time dimension.

3.5.3 Application of the recipe to use case III

Contrary to many AGV's, the fork-lift truck considered (Martínez-Barberá and Herrero-Pérez, 2010) incorporates a high degree of on-board autonomy to be flexible with respect to floor layout changes and to decrease the amount of manual work when establishing the a priori knowledge of the workplace.

1) Selecting an approach to combine algorithm classes: Re-planning is required due to the presence of unknown and possibly dynamic obstacles such as other vehicles (A1). Again, a finite sensor resolution demands a bounded uncertainty region (A2). The environment does contain moving obstacles (A3) but there is no prediction of their velocity or future position (A4). It is not necessary to plan in the full state space $\mathcal{S}$ (A5) but a plan in the reduced space is necessary to get as close as possible to optimality (A6). Due to the presence of kinematic constraints, a hierarchical approach is necessary (A7). In the following, the path planner will be discussed.

2) Selecting a representation algorithm class: The robot pose is described by three coordinates. The hierarchical approach, however, permits to omit the orientation when computing the path towards the goal since the kinodynamic constraints will be accounted for by the local planner (R1). A topological representation is desirable due to the size of the environment (R2). Due to the presence of unknown obstacles an approximate representation is required (R3). Since the planner only needs to work in two dimensions and the size is limited due to the presence of a topological representation, a grid based planner can be used (R4).

3) Selecting a search algorithm class: The planner plans towards subgoals of the topological map. Hence, a goal bias can be used to speed up the search (S1). As was mentioned before, re-planning is required due to changes in the environment (S2). It is desired to plan in realtime during navigation, so an anytime incremental planner would be most suitable (S3).

4) Comparison of the recipe's outcome and the use case: Similar to the prescription of the recipe, the navigation system in Martínez-Barberá and Herrero-Pérez (2010) combines a topological representation with a 2D gridmap and a local planner. The topological representation in Martínez-Barberá and Herrero-Pérez (2010) even consists of two levels: in the first level each node represents a zone, typically a room-like space. In the second level, the nodes represent doors, wait-points, dock-points and way-points. The grid representation is searched by a D* algorithm with a goal bias since this resulted in low re-planning times. The initial search, however, did not meet the real time requirement hence it cannot be considered an anytime planner. The local planner consists of three separate 'behaviors': an obstacle avoider, path tracker and dock driver. The contributions of the behaviors, which might be executed concurrently, are fused. The obstacle avoidance relates to potential field method while the path tracking is an implementation of receding horizon control.

3.6 Conclusion and future work

This chapter presents a recipe to select a representation class, a search algorithm class and an appropriate approach to combine these for a mobile robot that moves on a ground plane. In the first part, an appropriate approach to combine algorithm classes is selected. In the following two parts a representation class and search algorithm class to accomplish the two steps of motion planning are selected. An overview of the requirements and the corresponding design criteria is presented in Table 3.1. By considering both steps of a motion planner, for all requirements as introduced in Section 3.1, the recipe can be used to find an appropriate combination of motion planning algorithms classes.

The recipe is verified with three use cases where existing motion planners are successfully applied in practice. The recipe's outcome resembles the existing motion planners and the recom-

mendations from the recipe reveal the points for improvement that are indicated by the original authors. It can therefore be concluded that the recipe is beneficial to the selection of an appropriate motion planning approach and representations and search algorithm classes. To verify the recipe more extensively, future work will apply the recipe to more robots which already feature a motion planner that functions well in practice. As a part of this, we will apply the recipe to the robots of team Tech United Eindhoven (Tech United Eindhoven, 2014) and evaluate the resulting planner designs in the RoboCup competitions (Kitano et al, 1997). Furthermore, to increase the practical usability we are also interested in complementing this work with a tool that contains a database of the selected references and queries the decision trees of the recipe for an appropriate combination of motion planning algorithms.

Chapter 4

A representation method based on the probability of collision for safe robot navigation in domestic environments

This chapter[1] *introduces a three-dimensional volumetric representation for safe navigation. It is based on the OctoMap representation framework that probabilistically fuses sensor measurements to represent the occupancy probability of volumes. To achieve safe navigation in a domestic environment this representation is extended with a model of the occupancy probability if no sensor measurements are received, and a proactive approach to deal with unpredictably moving obstacles that can arise from behind occlusions by always expecting obstacles to appear on the robot's path.*
By combining the occupancy probability of volumes with the position uncertainty of the robot, a probability of collision is obtained.
It is shown that by relating this probability to a safe velocity limit a robot in a real domestic environment can move close to a certain maximum velocity but decides to attain a slower safe velocity limit when it must, analogous to slowing down in traffic when approaching an occluded intersection.

4.1 Introduction

Robots that navigate in indoor, domestic environments face an environment that encompasses obstacles and uncertainties. Obstacles generally vary in their size and shape and can be static as well as dynamic. Uncertainty typically arises from three sources (van den Berg et al, 2011): i) sensing uncertainty due to noisy and false sensor measurements, ii) uncertainty about the environment due to (partially) unknown parts of that environment and iii) robot position uncertainty due to localization errors and external disturbances acting on the robot. In the presence of these characteristics a robot always needs to guarantee that it is safe, *i.e.*, that it can ensure to come to a timely stop when a collision is imminent. Safe navigation can be achieved by moving a robot at very low velocities, typically below 0.1 m/s. However, by incorporating knowledge on the environment a robot can move with higher velocities without becoming unsafe.

A navigation system generally consists of a representation of the environment, one or more

[1]This chapter is under review for publication as an article in Autonomous Robots: Lunenburg et al (2015a)

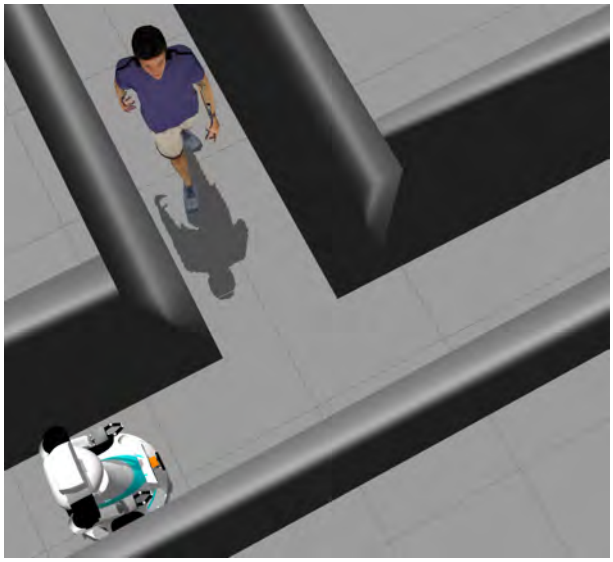


Figure 4.1: A corridor with a doorway that is occluded. In a domestic environment this increases the probability of collision for a robot as moving obstacles such as people can emerge from behind such occlusions.

algorithms that search for a path or trajectory through this environment and generate motor commands and a policy that coordinates these algorithms. A three-dimensional representation of the environment is a prerequisite to deal with the challenges that a typical domestic environment poses (Marder-Eppstein et al, 2010). An approximate cell decomposition is a common and popular approach to represent the environment (Goerzen et al, 2010; Hornung et al, 2013). Searching and executing a path in an environment with obstacles and uncertainties is typically achieved using a planner that finds a path to the goal that is executed by a reactive algorithm (Goerzen et al, 2010). Such a reactive algorithm controls the robot in real-time to avoid imminent collisions by stopping or swerving the robot when an obstacle is known to be on the robot's path.

Space that is unknown, due to occlusions and a limited sensory range, poses a threat to a robot that navigates in a domestic environment as moving obstacles might emerge from behind occlusions onto the robot's path. An exemplary situation occurs when a robot traverses a corridor as depicted in Figure 4.1. Near unknown space, such as a passage or a doorway that is occluded to the sensors, an obstacle can suddenly emerge. This requires the robot to lower its velocity such that it can guarantee that it will not collide with a possible incoming obstacle. The robot can increase its velocity when a confined part of the environment, like the part of the corridor past the doorway, is known to be free as there is no uncertainty arising from any unknown space. A reactive algorithm that has no information about this threat can approach at a velocity that does not allow the robot to detect the imminent collision and react to it to avoid collision. Hence, analogous to slowing down in hazardous situations in traffic, a representation is necessary that allows the robot to decide when it is able to move at a certain maximum velocity and when it must maintain a slower pace to ensure safety.

4.2 Related work and contribution

To achieve safe behavior some approaches modify the robot's planned path or motion using the obstacle representation. They enlarge obstacles by inflating their representation with a 'safe'

distance that encompasses the robot position uncertainty as well as the robot sensing uncertainty (Hsu et al, 2002; Chung et al, 2004). While this may ensure that a robot keeps a greater distance from obstacles, it may also prohibit a robot to traverse tight spaces. This is undesirable as tight spaces such as doorways are typically present in a domestic environment. Other approaches limit the robot's velocity based on distance to obstacles on the robot's path (Lingemann et al, 2005) or the amount of clearance on both sides of the robot (Fox et al, 1997). The former approach will fail when a moving obstacle emerges from occluded regions, *e.g.*, a doorway in a corridor as shown in Figure 4.1. The latter approach is conservative in that it will scale down its velocity in a narrow corridor, while in such a confined part of the environment the robot could safely drive at higher velocities.

Uncertainties can explicitly be taken into account to prevent collisions. A range of approaches estimate the probability of collision of the robot along its path (Missiuro and Roy, 2006; Burns and Brock, 2007; Guibas et al, 2009; van den Berg et al, 2011; Patil et al, 2012). These approaches plan a path by sampling the environment for feasible robot configurations that reduce the probability of collision during the execution of a path. These approaches result in robot motions with a lower probability of collision by explicitly considering the robot position and sensing uncertainty. However, these approaches do not take into account the uncertainty that arises due to unknown parts of the environment. In Marder-Eppstein et al (2010), a method is introduced that does track the unknown space in three dimensions. To guarantee safe behavior it is ensured that the robot never traverses this unknown space. However, it is assumed that the environment is mostly static, *i.e.*, it can not ensure safe behavior if an obstacle emerges from an occluded part of the environment. Furthermore, once any unknown space is marked as free it will remain free. This makes the representation over-confident in its assumption that space is free as this space can become occupied again in a changing environment. Hence, a time-dependent occupancy probability model is necessary, instead of merely tracking unknown space.

Finally, some approaches explicitly model the uncertainty that arises due to unpredictable moving obstacles, *e.g.*, humans (Philippsen et al, 2006, 2008; Rohrmüller et al, 2008). Moving obstacles are extracted from subsequent sensor readings and their position and velocity is estimated to obtain a probabilistic model that resembles the risk of collision. A moving obstacle can, however, be occluded up to an imminent collision.

Hence, current approaches that deal with uncertainties to allow safe navigation are not fully suited for a domestic environment. Not all present uncertainties are considered in an integrated approach and they lack an explicit model to deal with the uncertainty that arises due to occlusions in an environment with unpredictably moving obstacles.

This chapter contributes a three-dimensional representation that allows a motion planner to decide when it can move at a certain maximum velocity and when it must maintain a slower pace to ensure safety based on the probability of collision. This is achieved by explicitly representing the multiple uncertainties that are present in a domestic environment and using the probability of collision to determine a safe velocity limit, analogous to slowing down in hazardous situations in traffic.

4.3 Environment representation

In this section the proposed environment representation for safe navigation is elaborated. First, it will be described how this representation, based on the *OctoMap* framework (Hornung et al, 2013) and first introduced in Coenen et al (2014), uses probabilistic fusion of sensor measurements to be robust against uncertainty in sensing. Secondly, it is described how uncertainty due to unknown space is represented if no measurements are received and how the probability of obstacles appearing on the robot's path from behind occlusions is represented using a proactive

approach (Alami et al, 2002). These first two parts results in the representation of the probability of volumes being occupied by an obstacle. Next, the probability of the robot occupying a volume is represented using a model of the robot position uncertainty. Finally, the probability occupancy of obstacles and the probability occupancy of volumes by the robot are combined to represent a probability of collision. This probability is then related to a safe velocity limit that the robot must attain in order to guarantee safety.

4.3.1 Robot sensing uncertainty representation

The *OctoMap* framework provides a volumetric octree-based representation of the environment (Hornung et al, 2013). It models the environment as free, occupied and unknown cubic volumes or so-called voxels. Sensor measurements are integrated probabilistically using occupancy grid mapping (Moravec, 1988). This technique allows a probabilistic fusion of multiple sensor measurements making it robust to noisy and false sensor measurements. Also, it allows the integration of measurements from multiple, different sensors.

The occupancy of a voxel is updated as measurements are received. Each voxel n, with a resolution r, has a probability $P(n)$ of being occupied. The occupancy probability of all volumes is typically initialized to unknown, *i.e.*, the uniform prior $P(n) = 0.5$. Then, $P(n)$ is updated based on a sensor specific model as measurements $z_{1:k}$ up to time step k are received. The update rule for the estimated voxel occupancy, using the log-odds (L) notation (Moravec, 1988), is

$$L(n \mid z_{1:k}) = L(n \mid z_{1:k-1}) + L(n \mid z_k), \tag{4.1}$$

with

$$L(n) = \log \left[\frac{P(n)}{1 - P(n)} \right], \tag{4.2}$$

where $L(n \mid z_{1:k})$ is the estimated log-odds probability of a voxel given measurements $z_{1:k}$, $L(n \mid z_{1:k-1})$ is the previously estimated log-odds probability and $L(n \mid z_k)$ denotes the log-odds probability of voxel n being occupied given the measurement z_k. The update of occupancy probability is typically performed in log-odds as using additions is faster than the multiplications that are necessary when (4.1) is expressed in $P(n)$.

$L(n \mid z_k)$ relies on a sensor model that relates the sensor measurements to the occupancy probability of a voxel:

$$L(n \mid z_k) = \begin{cases} l_{\text{free}}, & \text{if } n \text{ is marked as free} \\ l_{\text{occ}}, & \text{if } n \text{ is marked as occupied} \end{cases} \tag{4.3}$$

Given equally likely measurements ($l_{\text{free}} = l_{\text{occ}}$), a voxel that is marked as free k times needs to be marked as occupied equally many k times before its occupancy probability is equal again. This makes the representation unable to change as quickly as the environment. As discussed in Hornung et al (2013), the representation can be made adaptable to a changing environment by limiting the probability in the update rule in (4.1) to a lower and upper bound on the log-odds value, respectively $l_{\min}$ and $l_{\max}$ or $p_{\min}$ and $p_{\max}$ on the probability. For more details on the update formula and its background the reader is referred to Moravec (1988) and Hornung et al (2013).

4.3.2 Environment uncertainty representation

The update rule introduced in (4.1) models the occupancy probability of voxels under the assumption that they receive measurements. However, large parts of the environment typically yield no measurements as they are occluded to the robot's sensors. It is important that the

update rule in (4.1) also describes the occupancy probability of voxels if no measurements are received. For example, consider a robot that moves in an environment such that all voxels n in the representation are receiving measurements, either marking voxels as free or occupied. Then, given a static environment, the occupancy of the voxels will approach the threshold, *i.e.*, either $P(n) = p_{\min}$ or $P(n) = p_{\max}$. However, it is incorrect to assume that the occupancy of those voxels that do not receive measurements does not change if the environment is dynamic. In other words, such a representation is over-confident in its assumption that space is either free or occupied. Hence, the occupancy of a voxel is more realistically modeled to become unknown again as no measurements are received for some time.

A time-dependent occupancy probability model is added to the representation to deal with this environment characteristic. Instead of only updating a voxel n if a measurement is received, it is updated at every time step k that the environment is updated. Thereto, the sensor model, introduced in (4.3), can be extended with the occupancy probability update rule

$$l_{\text{dec}} = (k - k^n_{z,\text{last}})\Delta_{\text{dec}} \quad \text{if } n \text{ is } \textbf{not} \text{ marked}, \tag{4.4}$$

where $k^n_{z,\text{last}}$ is the time step at which a voxel n received its last measurement update and Δ_{dec} indicates the rate of decay of the probability of a voxel n in l_{dec}/k. The value of the log-odds value l_{dec} depends on the occupancy of a voxel according to:

$$l_{\text{dec}} = \begin{cases} +l_{\text{dec}}, & \text{if } l < 0, \textit{ i.e.}, P(n) < 0.5 \\ 0, & \text{if } l > 0, \textit{ i.e.}, P(n) > 0.5 \end{cases} \tag{4.5}$$

Hence, if a voxel receives no measurements and $P(n) > 0.5$ it gradually turns to unknown again, *i.e.*, $P(n) = 0.5$. This is visualized for a voxel that is at the lower probability threshold $p_{\min}$ in Figure 4.2. The update rule in (4.5) does not let the occupancy probability of an occupied voxel decrease from $P(n) > 0.5$ to unknown as no measurements are received, since this would require more knowledge of obstacles to discriminate between them. For example, a voxel that belongs to a static obstacle, *e.g.*, a wall, is more likely to remain occupied than a voxel that belongs to a moving obstacle, *e.g.*, a human. The update rule in (4.5) is based on a model that is linear in log-odds, similar to the sensor measurement update model in (4.1). Therefore, the rate of decay Δ_{dec} can intuitively be chosen as a rate at which measurements are discarded again, such that the probability returns to unknown. For example, a measurement that decreases the probability of occupancy is discarded at the next step the environment is updated and no measurement is received by choosing $\Delta_{\text{dec}} = l_{\text{free}}$. Hence, it takes an equal amount of time steps to increase from $P(n) = p_{\min}$ to $P(n) = 0.5$ if no measurements are received as it took to get from $P(n) = 0.5$ to $P(n) = p_{\min}$ when this voxel was marked as free.

The representation of occupancy probability is three-dimensional. However, to keep the computational complexity of the representation tractable the three-dimensional grid of voxels is projected down to a two-dimensional representation. Therefore, each column of voxels in the occupancy map is projected down to a grid cell with occupancy probability p according to

$$p_{\mathbf{c}} = \max_i P(n_i), \tag{4.6}$$

where $\mathbf{c}$ is a grid cell with resolution r. Taking the maximum occupancy probability is a conservative strategy that is necessary for safe navigation as it ensures that the robot never underestimates the possibility that a voxel is occupied at any height in a column.

As mentioned, in a domestic environment an obstacle can move on the robot's path from behind an occlusion, as illustrated in Figure 4.1. The occupancy probability model that has been introduced so far must be extended with a velocity model to deal with this. This model is based on a proactive approach that is introduced in Alami et al (2002). By proactive it is meant

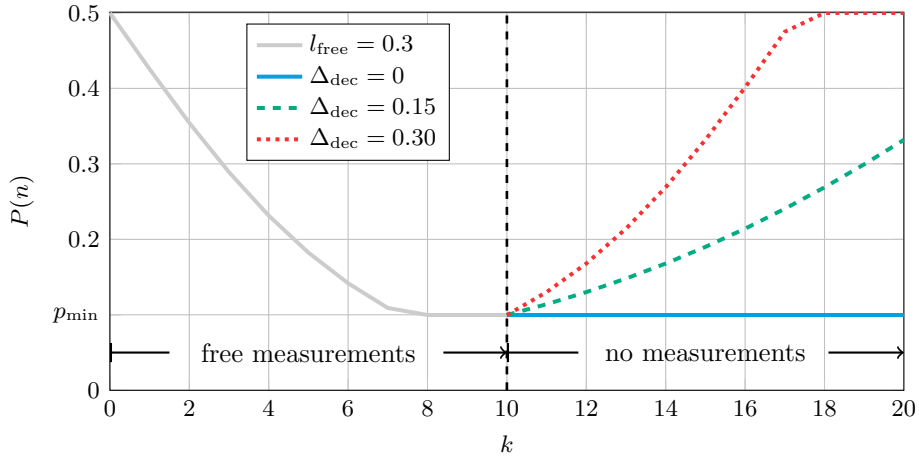


Figure 4.2: The occupancy probability of a voxel $P(n)$ decreasing with $l_{\text{free}} = -0.3$ according to (4.3) as it is marked as free at the first ten time steps and increasing with l_{dec} according to (4.5) for various Δ_{dec} as it is not marked for ten following time steps.

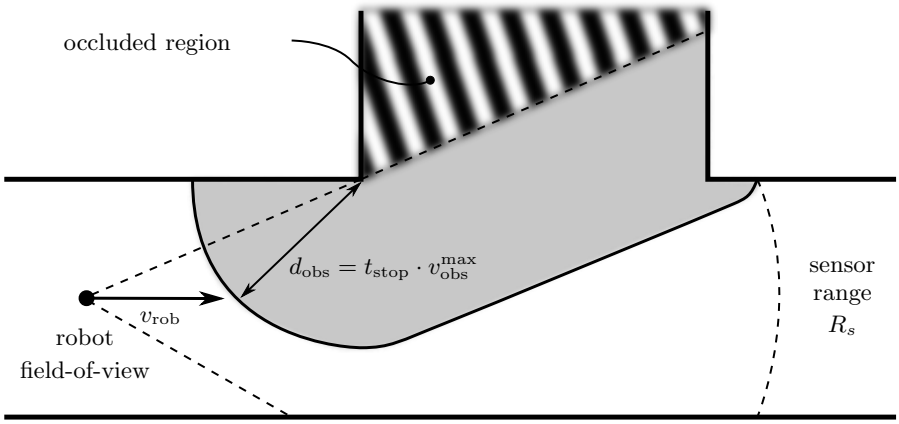


Figure 4.3: An obstacle can appear on the robot's path out of a region that is occluded to the robot. To represent this possibility the uncertain region is inflated with the maximum distance the obstacle can travel within the time the robot needs to come to a stop.

that the robot is always expecting that a moving obstacle can appear on the robot's path from an occluded region. This is achieved by inflating the occupancy probability of cells in the two-dimensional projected map that are not marked as an obstacle, as depicted in Figure 4.3. The inflation distance d_{obs} depends on the distance that an obstacle can travel within the time the robot needs to come to a stop, $t_{\text{stop}} = v_{\text{rob}}/a_m + t_d$, where v_{rob} is the robot's velocity, $-a_m$ is the maximum deceleration and t_d is the maximum update delay that is present before an incoming obstacle is actually detected. The distance an obstacle can travel with a maximum velocity $v_{\text{obs}}^{\max}$ is now $d_{\text{obs}} = t_{\text{stop}} v_{\text{obs}}^{\max}$, which is conservative as it assumes that an obstacle will maintain its maximum velocity. However, this assumption is necessary as any obstacle trajectory is assumed to be unknown.

The cells of the grid map with an occupancy probability are inflated with the obstacle inflation distance d_{obs}. A grid map cell can be affected by the inflation of multiple cells. Hence, these probabilities must be merged and this is done by taking the maximum probability of all inflated probabilities. In practice this means that all inflated cells are regarded as one potential source of collision risk. By guaranteeing that the robot is safe in this situation it can also be guaranteed that it is so in the case of multiple sources of collision risk.

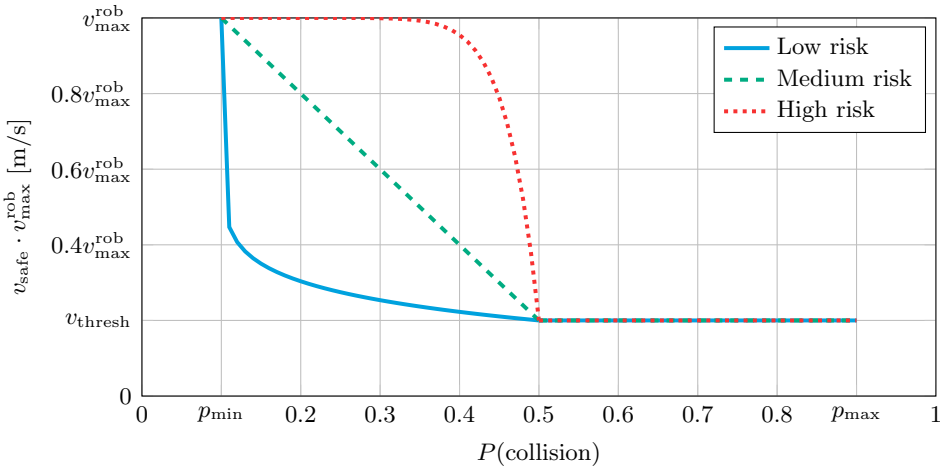


Figure 4.4: The safe velocity limit as a n-degree polynomial function of the robot's maximum velocity with $n = 0.1$ (low risk), $n = 1.0$ (medium risk) and $n = 10.0$ (high risk). A threshold velocity, $v_{\text{thresh}} = 0.2v_{\max}^{\text{rob}}$, allows the robot to move at a limited velocity in the presence of any risk of collision. The robot can be allowed to move faster by taking more risk.

4.3.3 Robot position uncertainty representation

The uncertainty that is present in the robot's position is due to errors in the estimation of the robot's position relative to a map of its environment. This uncertainty can be modeled as a bivariate normal (Gaussian) distribution

$$\mathbf{x} \sim \mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{4.7}$$

where the mean $\boldsymbol{\mu}$ is the robot position at $\mathbf{x} = (x; y)$ and $\boldsymbol{\Sigma}$ is the two-dimensional covariance matrix. For simplicity, it is hereby assumed that robot has a circular shaped platform. The occupancy probability of a cell $\mathbf{c}$ by the robot can be determined according to the normal distribution $\mathcal{N}(\mathbf{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma})r^2$, where r^2 is the surface of $\mathbf{c}$. The distribution of the robot's position is considered within a prediction interval to limit computation. Given a probability of $1 - \alpha$ the robot's position is guaranteed to be in a certain region $\mathcal{A}$ centered around the mean $\boldsymbol{\mu}$. The region $\mathcal{A}$ is an ellipsoid with a 'radius' k, given by the relation $(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) = k^2$ that can be deduced from (4.7). Given α, the ellipsoid shaped region $\mathcal{A}$ follows from $k = \sqrt{\chi_2^2(\alpha)}$, where $\chi_2^2\alpha$ is the upper (100α) percentile from the two-dimensional chi-squared distribution. For example, for $\boldsymbol{\Sigma} = [0.1\,0; 0\,0.1]$ and $\alpha = 0.05$, the robot's position is located with a probability of $p = 0.95$ in a circle with a radius $k = \sqrt{0.1 \cdot 5.99}$.

4.3.4 Combined representation

The environment uncertainty model in Section 4.3.2 and the robot position model in Section 4.3.3 are combined to obtain a probability of collision. The probability of a cell being occupied by an obstacle and the probability of that same cell being occupied by the robot are assumed to be independent. Hence, the probability of collision, *i.e.*, a cell is occupied by both the robot and an obstacle, is equal to the product of both probabilities. Now, the probability of collision at a specific cell can be determined according to

$$P(\text{collision}) = \sum_{\mathbf{c} \in \mathcal{A}} P(\mathbf{c} = \text{occ}) \cdot \mathcal{N}(\mathbf{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma})r^2, \tag{4.8}$$

This probability of collision must be related to a velocity limit v_{safe} that ensures safe navigation. Thereto, the maximum velocity must be lowered as the probability of collision increases,

analogous to driving in traffic where the road becomes dangerous. The maximum velocity can be derived given that $d_{\text{rob}} + d_{\text{obs}} < R_s$ must always hold to avoid collision. Here, R_s is the limited sensory range and $d_{\text{rob}} = \frac{1}{2}a_m(v_{\text{rob}}/a_m)^2 + v_{\text{rob}}t_d$ is the maximum distance that the robot travels to come to a stop. Given $d_{\text{obs}} = t_{\text{stop}}v_{\text{obs}}^{\max}$, the maximum robot velocity is

$$v_{\text{rob}}^{\max} = -v_{\text{obs}}^{\max} - a_m t_d + \sqrt{a_m^2 t_d^2 + (v_{\text{obs}}^{\max})^2 + 2a_m R_s}. \tag{4.9}$$

One could argue that to be strictly safe the robot must have a zero velocity in the presence of a nonzero probability of collision. However, the robot must accept some risk of collision during navigation as it is generally not absolutely certain that space is free. This is an inherent consequence of the update rule in (4.1) and the clamping threshold $p_{\min}$ on the occupancy probability. Hence, a threshold velocity v_{thresh} is used, that allows to robot to drive at a velocity close zero in the presence of any uncertainty. The choice for v_{safe} is a trade-off between moving safely at v_{thresh} or moving faster with more risk of collision. This trade-off is visualized in Figure 4.4 for a polynomial function with different degrees. Of course, the exact function to relate the probability of collision to a safe velocity limit is a design choice that depends on the consequences of collision in a particular scenario.

4.4 Implementation

4.4.1 Global and local planner

The environment representation, introduced in Section 4.3, is implemented in our existing motion planning approach that has been successfully used in the RoboCup @Home league over the past years (Lunenburg et al, 2013a, 2014). This approach consists of a global planner that searches for a plan to the goal and a local planner that computes a velocity such that this global path is followed.

The global planner uses an A* algorithm that searches for a cost-optimal path. The costs are encoded in a costmap that represents the distance to obstacles. A plan is computed at a fixed frequency of 2 Hz. If the current plan is free, a re-plan is executed if the estimated time for the new plan is significantly less. This way the robot does not hold on to an old plan when a shorter path has become available but switching between two paths of approximately similar costs is avoided. If the current plan is blocked and an alternative plan is significantly longer, the robot will wait before executing this. This ensures that if a path is blocked for only a short time (*e.g.*, by a moving obstacle), an unnecessarily long re-plan is not executed. If the new plan is of approximately equal length to the original plan it will be executed immediately to avoid unnecessarily long waiting.

The local planner computes an omni-directional velocity (v_x, v_y, v_θ) in the direction of the next pose of the global path that maximizes velocity while obeying acceleration limits and the velocity limit that the representation imposes, *i.e.*, v_{safe} as introduced in Section 4.3.4. If the error e_θ between the path and the robot orientation exceeds a certain threshold, an in-place rotation is performed to keep the robot facing the driving direction. To avoid discrepancies between the local and global planner, both use the same global representation.

4.4.2 Task integration

With the representation, the global and the local planner the robot can move from a start pose to a target pose. However, navigation systems typically also address issues such as recovery behaviors and replanning to make the system more robust. Nevertheless, navigation is part of

a larger task. Hence, it should not be considered in itself and integration with other modules is essential to optimize performance.

Therefore, in order to enhance this integration, the navigation system is coordinated by a task executer, *i.e.*, the software component that activates the various subsystems of the robot to achieve a certain task. This executer has additional knowledge of both the environment and the task so that it can make more informed decisions about the desired behavior. This manifests itself in the definition of goals, actively directing sensors and in recovery behaviors:

Goals: Goals can be defined as a pose x, y, θ in the global coordinate frame or as a semantic query to the robots knowledge base. This query will return a list of possible poses that meet the requirement. The best target pose is selected based on distance to travel and proximity to obstacles. Not only do these queries provide a more intuitive interface (you can ask the robot to drive to, *e.g.*, 'the table' or 'a shelf that has not yet been visited'), but they also make navigation more robust by trying the next pose in the list if a target pose turns out to be unreachable.

Directing sensors: Actively directing sensors, *e.g.*, gaze direction, greatly enhances the environment representation. Since the task of the robot may put additional requirements on the gaze direction, the gaze direction itself is also controlled by the task executer. Under normal operation, the robot looks to the current path at a fixed distance in front of him. If an obstacle is encountered on the path, the robot will turn its attention to this obstacle.

Recovery behaviors: As is also mentioned in, *e.g.*, Marder-Eppstein et al (2010), even with a good navigation system the robot can still get stuck in some situations, exposing the need for recovery behaviors. Especially in cluttered, dynamic environments, the robot may not be able to clear all obstacles that are no longer there. By explicitly looking at obstacles, the probability of moving obstacles and sensor noise being cleared increases significantly. The currently used recovery behaviors include clearing the space around the robot if the local planner is stuck and resetting the representation to its default state, consisting only of obstacles and unknown space, if no valid global plan is possible. The task executer decides when these behaviors are executed, easing integration of possible future behaviors such as asking people to move out of the way or removing obstacles by itself.

4.5 Experimental results

The navigation approach in this chapter is verified using the AMIGO robot, a domestic service robot developed by Eindhoven University of Technology. AMIGO competes in the RoboCup@Home League. This annual competition, where domestic service robots compete in performing household tasks, is part of the international RoboCup project (Kitano et al, 1997). AMIGO has a four-wheeled omni-directional base that is capable of navigating through wheelchair-accessible areas. Its torso is equipped with two anthropomorphic arms to perform manipulation tasks. To extend the reach of the arms the torso is connected to the circular base with a lifting mechanism. In this study the arms are not used. A Hokuyo UTM-30LX Laser Scanner, positioned at the front-side of its base, provides a 220° view at 30 cm above the ground. AMIGO uses adaptive Monte Carlo localization (AMCL) (Fox, 2001) to localize itself on a static, a priori map. A Microsoft Kinect mounted with a pan-tilt unit on top is used to provide three-dimensional pointcloud data.

Simulations have been performed in a scenario based on the 2013 RoboCup@Home League set-up and experiments have been performed in a domestic environment in the Robotics Lab of the Eindhoven University of Technology as well as in the university library. The parameters are set according to Table 4.1. The maximum robot velocity robot is determined according to (4.9), with $t_d = \frac{2}{3}$ s. This maximum delay is based on the minimum update rate of the representation which is 3 Hz. As it can take up to two measurements before an obstacle is detected the maximum

Table 4.1: Different Model Parameters Used in Experiments

sensors		environment		robot	
l_{free}	-1.10 $(p = 0.25)$	$v_{\text{obs}}^{\max}$	1.0 m/s	$v_{\text{rob}}^{\max}$	0.7 m/s
l_{occ}	$+0.85$ $(p = 0.70)$	R_s	3.2 m	v_{thresh}	$0.2 v_{\text{rob}}^{\max}$
$l_{\min}$	-1.40 $(p = 0.20)$	t_d	$\frac{2}{3}$ s	a_m	0.5 m/s^2
$l_{\max}$	$+2.20$ $(p = 0.90)$	r	0.05 m	σ_x^2, σ_y^2	0.1 m
				α	0.05

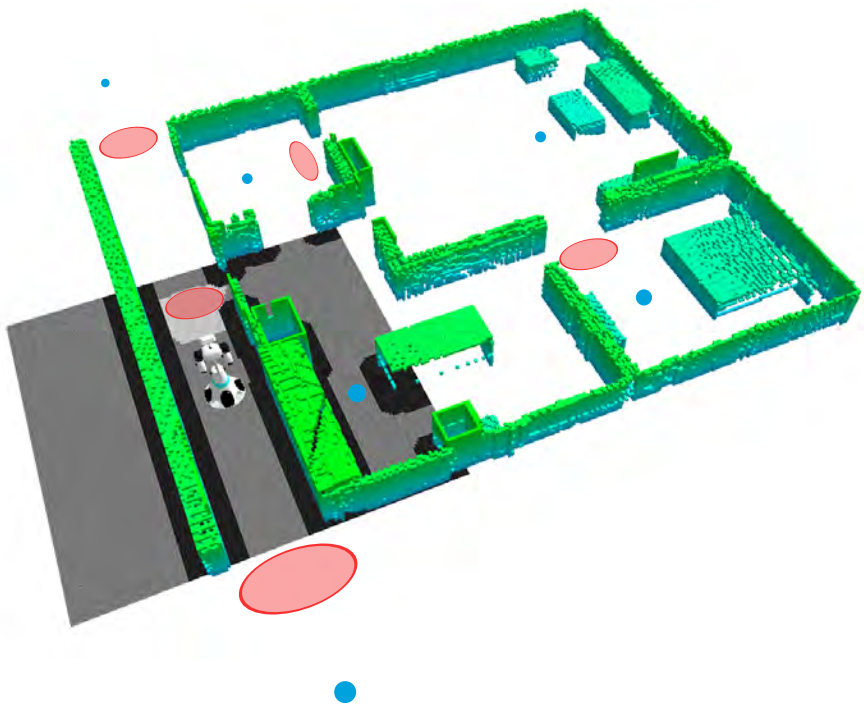


Figure 4.5: A model of the 2013 RoboCup@Home League set-up used for the simulation experiments. The blue dots indicate the predefined goal locations that are randomly visited by AMIGO. The areas that are marked red indicate regions where an obstacle can cross with AMIGO.

delay is equal to $\frac{2}{3}$ s. The voxel resolution of the representation is $r = 0.05$ m, such that AMIGO can still fit through the narrowest doorway with a width of 80 cm. The covariance terms of the position uncertainty model are based on the covariance matrix provided by the AMCL module. For simplicity, the maximum variance obtained from previous tests with AMIGO is used.

4.5.1 Results of simulation experiment

Simulations are performed with AMIGO in a model of the RoboCup@Home League 2013 set-up, as shown in Figure 4.5. The representation is tested for the velocity risk profiles $n = \{0.1, 1, 10\}$, probability decay rates $\Delta_{\text{dec}} = \{0.0, 0.15, 0.30\}$ and with and without taking into account moving obstacles, $v_{\text{obs}}^{\max} = \{0.0, 1.0\}$ m/s. The position uncertainty is modeled according to the parameters in Table 4.1 for all tests.

In a first test the robot drives through the corridor, as shown in Figure 4.5, while an obstacle appears from the occluded doorway on its right, thereby blocking the robot's path. This simulation demonstrates the effect of taking obstacle velocities into account as well as the difference

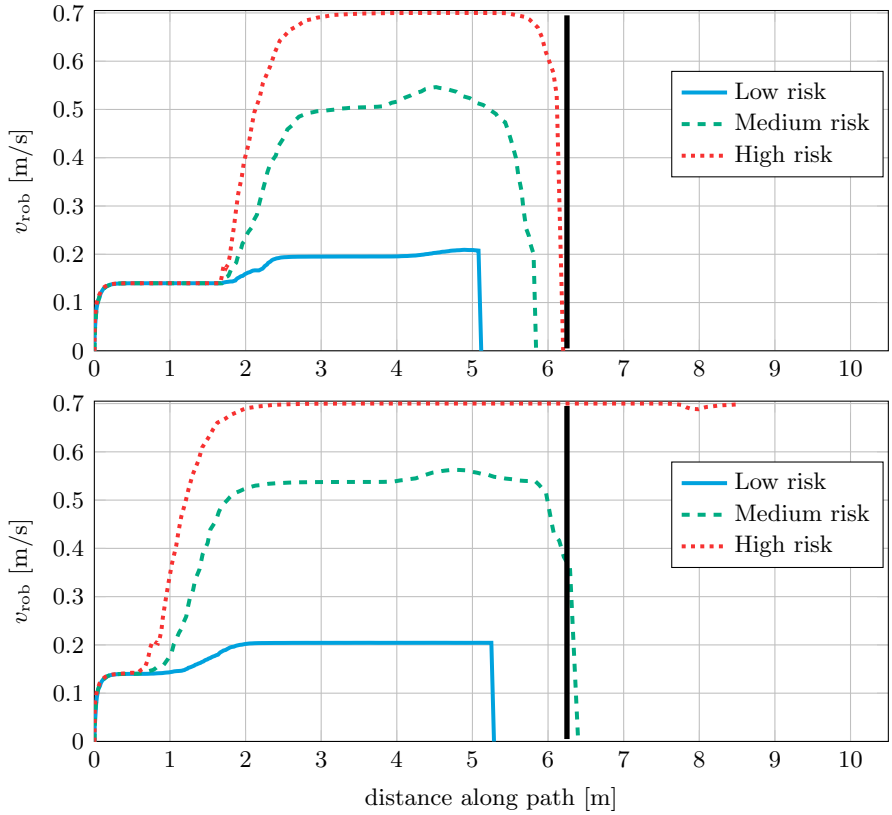


Figure 4.6: The velocity of AMIGO with (upper figure) and without (lower figure) taking the obstacle velocity into account, respectively $v_{\text{obs}}^{\text{max}} = 1.0$ m/s and $v_{\text{obs}}^{\text{max}} = 0.0$ m/s, along the same path through the corridor of the simulation environment. Halfway the corridor an obstacle entered on AMIGO's path from an occluded doorway, indicated by the black line at 6.25 m. In the upper plot, the robot stops before colliding with the obstacle, regardless of the velocity function. In the lower plot, the robot collides with the obstacle if a medium or high risk velocity profile is used.

between the low, medium and high risk velocity functions. To focus on these two effects, the rate of probability decay is not considered in this test. The velocity profile along the path is presented in Figure 4.6. At the beginning of the test the part in front of the robot is unknown and hence it moves at the speed defined by the velocity threshold. When the obstacle velocity is taken into account, *i.e.*, $v_{\text{obs}}^{\text{max}} = 1.0$ m/s, the unknown area is inflated. Since this unknown area behind the robot is inflated past the robot footprint, the threshold velocity for $v_{\text{obs}}^{\text{max}} = 1.0$ m/s must be attained longer than for $v_{\text{obs}}^{\text{max}} = 0.0$ m/s. Near the doorway the test with $v_{\text{obs}}^{\text{max}} = 0.0$ m/s disregards the increased probability of collision due to moving obstacles resulting in a collision for a medium ($n = 1$) and high ($n = 10$) risk velocity profile. For $n = 10$ the robot even failed to recognize the obstacle and hit it with maximum velocity. Due to the position uncertainty the robot typically moves at a velocity near its threshold for a low risk velocity profile, thereby giving it enough time to detect the obstacle and to stop in time. The test with $v_{\text{obs}}^{\text{max}} = 1.0$ m/s shows that a velocity near the maximum velocity, *i.e.*, $v_{\text{rob}}^{\text{max}} = 0.7$ m/s, is possible without collision.

In a second simulation AMIGO drives to random locations, within a set of predefined locations as depicted in Figure 4.5, for 30 minutes. Obstacles are modeled to move at random time intervals over predefined paths that will intersect with AMIGO, resulting in areas with an increased risk of collision similar to the test in the corridor. By navigating over a longer timespan the influence of the rate of probability decay (Δ_{dec}) can be determined on the performance. As a measurement of performance the number of collisions and the average velocity during the test

Table 4.2: Simulation Test Results

	rate of change	v_{safe} profile: low risk ($n = 0.1$)	medium risk ($n = 1$)	high risk ($n = 10$)
$v_{\text{obs}}^{\max} = 1.0\ \frac{m}{s}$	$\Delta_{\text{dec}} = 0.3$	$n_{\text{coll}} = 0$ $\bar{v} = 0.11$	$n_{\text{coll}} = 0$ $\bar{v} = 0.24$	$n_{\text{coll}} = 1$ $\bar{v} = 0.30$
	$\Delta_{\text{dec}} = 0.15$	$n_{\text{coll}} = 0$ $\bar{v} = 0.14$	$\mathbf{n}_{\text{coll}} = \mathbf{0}$ $\bar{\mathbf{v}} = \mathbf{0.27}$	$n_{\text{coll}} = 3$ $\bar{v} = 0.32$
	$\Delta_{\text{dec}} = 0.0$	$n_{\text{coll}} = 1$ $\bar{v} = 0.19$	$n_{\text{coll}} = 5$ $\bar{v} = 0.30$	$n_{\text{coll}} = 9$ $\bar{v} = 0.35$
$v_{\text{obs}}^{\max} = 0.0\ \frac{m}{s}$	$\Delta_{\text{dec}} = 0.3$	$n_{\text{coll}} = 0$ $\bar{v} = 0.13$	$n_{\text{coll}} = 3$ $\bar{v} = 0.26$	$n_{\text{coll}} = 7$ $\bar{v} = 0.31$
	$\Delta_{\text{dec}} = 0.15$	$n_{\text{coll}} = 0$ $\bar{v} = 0.14$	$n_{\text{coll}} = 3$ $\bar{v} = 0.29$	$n_{\text{coll}} = 7$ $\bar{v} = 0.35$
	$\Delta_{\text{dec}} = 0.0$	$n_{\text{coll}} = 4$ $\bar{v} = 0.20$	$n_{\text{coll}} = 12$ $\bar{v} = 0.33$	$n_{\text{coll}} = 14$ $\bar{v} = 0.43$

is reported in Table 4.2.

The results show that the rate of probability decay has a noticeable influence on the number of collisions. For $\Delta_{\text{dec}} = 0.15$ the number of collisions drops significantly. The difference with tests with $\Delta_{\text{dec}} = 0.3$ is not significant. Furthermore, the tests with a low risk velocity function, *i.e.*, $n = 0.1$, resulted in a robot that appeared too conservative as it was not able to achieve its goal position in most situations. Hence, accepting some risk results in a robot that navigates with a higher average velocity without necessarily increasing the number of collisions. In a few occasions, a moving obstacle was not completely cleared before it was out of the sensor view and therefore cluttered the environment, preventing the robot from reaching its goal. A model for the decrease in occupancy probability for occupied volumes, as discussed in Section 4.3.2, would alleviate this problem.

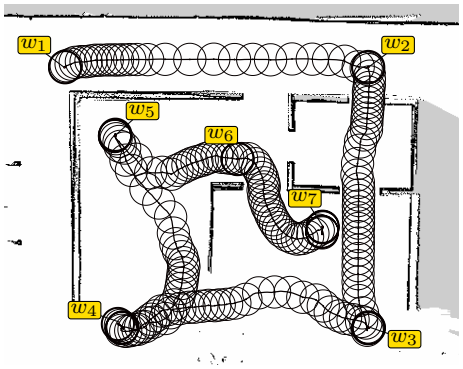
It can be concluded that the increased probability of collision due to moving obstacles must be taken into account and the probability of occupancy must be time-dependent in order to navigate without collision. Furthermore, the trade-off between velocity and the probability of collision is clear. Accepting some risk of collision during navigation, *i.e.*, choosing a medium risk velocity limit (see Figure 4.4), results in a robot that moves at its maximum velocity if it can and at its velocity threshold when it must to ensure safety.

4.5.2 Validation in a laboratory experiment

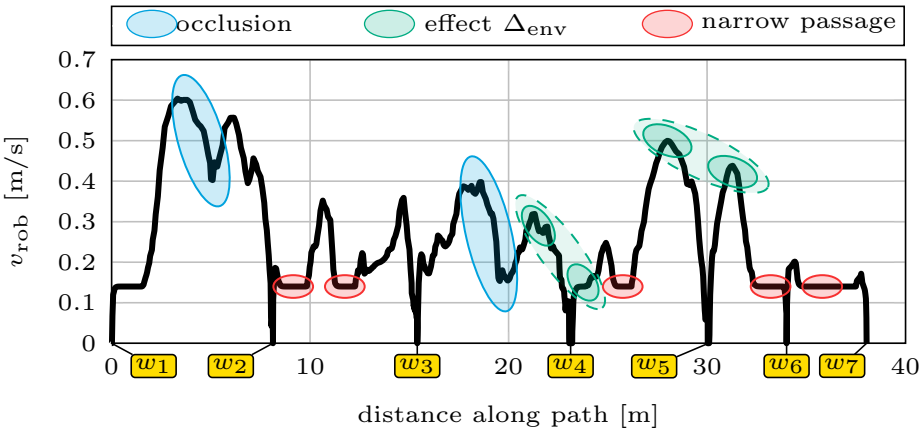
The proposed environment representation is validated by an experiment on the robot in a domestic environment. This environment, shown in Figure 4.7a, is a partial replica of the 2013 RoboCup@Home League set-up on a slightly smaller scale (0.9:1). In the experiment the robot visits a set of waypoints numbered w_1 to w_7. The robot is initialized with a three-dimensional map, as depicted in Figure 4.7b, and a static map for localization, as depicted in Figure 4.7c, that both are generated off-line. A variety of challenging obstacles are present, that are not in the a priori map, such as a pair of mini wooden shoes between w_3 and w_4, the IV pole and overhanging sidetable near the hospital bed at w_4 and the desk chairs near w_5 and w_6. The parameter set as used in the simulation experiments is used for the experiment and, based on the simulation test results, the variable parameters are set to $v_{\text{obs}}^{\max} = 1.0$ m/s, $\Delta_{\text{dec}} = 0.15$ and $n = 1$.

(a) Test setup at the Robotics Lab of Eindhoven University of Technology.

(b) Robot visualization.



(c) Footprints on the localization map.



(d) The robot velocity along the executed path during the experiment.

Figure 4.7: An experiment with AMIGO in a real domestic environment. The robot drives to a number of predefined waypoints (see Figure 4.7c) in the environment depicted in Figures 4.7a and 4.7b. The resulting velocity profile can be seen in Figure 4.7d where a decrease in velocity due to occlusions is marked cyan, a velocity difference due to a changing environment is marked green and a decreasing velocity due to a narrow passage is marked red.

Figure 4.8: AMIGO driving through the university library. (Photo courtesy of FM-Fotografie.)

AMIGO navigated without collision in the domestic environment. In Figure 4.7c the robot's footprint on the localization map and its executed path are shown at every second. The velocity along the executed path is displayed in Figure 4.7d. The three-dimensional representation correctly represented the obstacles encountered during the run. At parts with an increased probability of collision due to occlusions, *e.g.*, near the doorway in the corridor, the robot lowered its velocity, allowing it to stop in time if an obstacle suddenly appears on its path. In narrow passages the robot lowered its velocity to its threshold to be robust against position uncertainty. The effect of the probability decay rate Δ_{dec} is noticeable near w_4 and w_5. The space near these waypoints is marked as free as the robot approaches. However, as the robot continues to its next waypoint the space becomes unknown again as it is out of sensor range. Hence, a lower velocity limit is present. In open and known space, such as at the beginning of the corridor and near w_5, the robot achieved velocities near its maximum velocity. The sudden decreases in velocity between waypoints are the results of in-place rotations to keep the robot facing its driving direction.

4.5.3 Experiences in a real world experiment

Although the experiment in the previous section clearly demonstrates the caution that the robot takes in case of risk of collisions due to occlusions, to a dynamic environment and narrow passages, the environment itself actually is static without any people moving around. Since the robot is supposed to navigate through a domestic environment, a second experiment has been conducted. This has taken place in the library of the Eindhoven University of Technology (see Figure 4.8). The library is much larger (approximately 70 m $\times$ 40 m) than the environment in Section 4.5.2 and much more challenging because of the amount op people walking around and paths being blocked by, *e.g.*, chairs and bags located at random locations.

The first step to obtain a clean default OctoMap of the environment was to make a localization map (see Figure 4.9). Subsequently, this map was post-processed to remove people, chairs, bags and other non-stationary objects. Furthermore, overhanging desks were included. The result can be seen in Figure 4.10a. Finally, this was extruded to obtain a default OctoMap, see Figure 4.10b.

In this environment, a number of interesting locations was defined as if the robot was showing

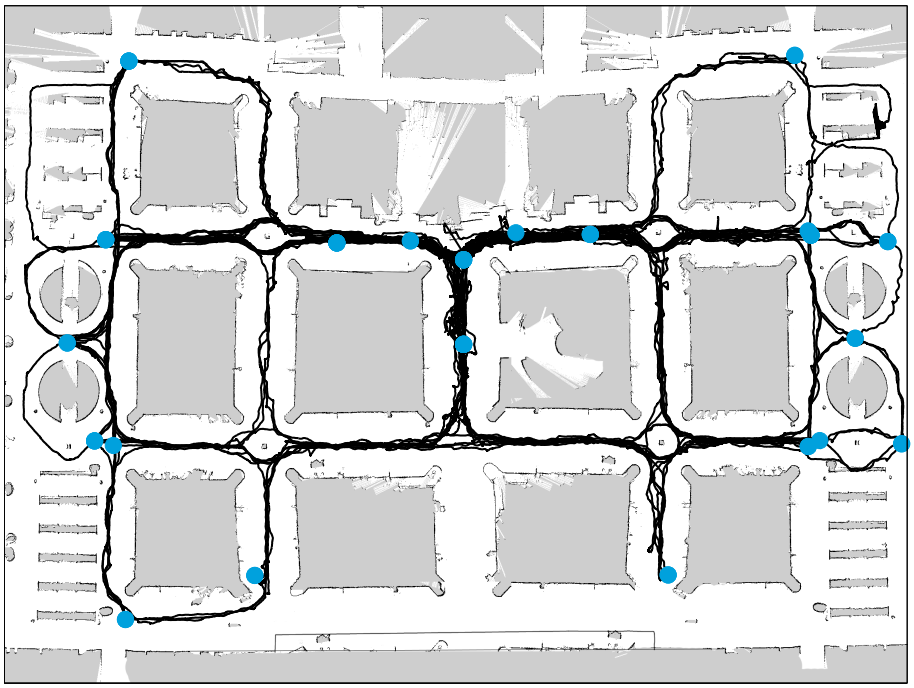


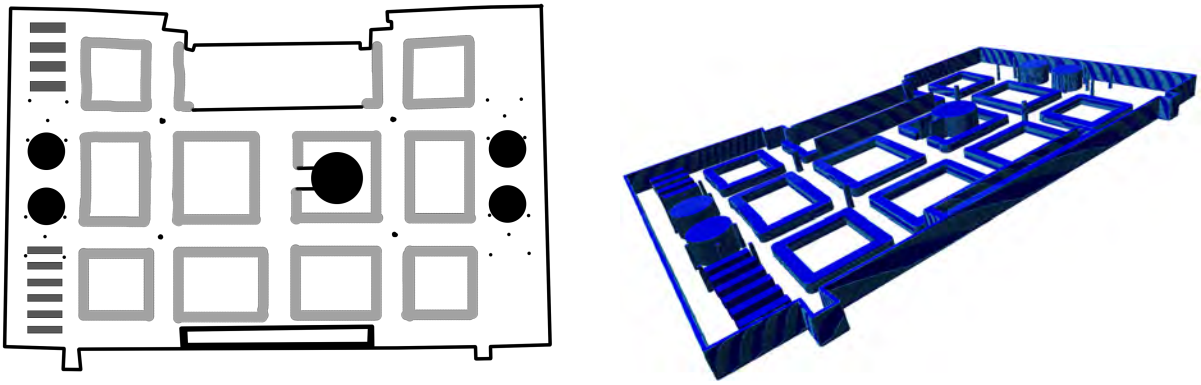
Figure 4.9: The localization map of the university library and the paths AMIGO took during the experiments. The goal locations are denoted in blue.

people the way to, *e.g.*, the elevator, the copier or the location of certain books. Using a smartphone, a user could select a location and the robot would show him the way. In case there was no pending goal request, the robot would select one at random. This way, the robot covered a total of 2.9 km in this environment. Although this is by far not as much as in, *e.g.*, Marder-Eppstein et al (2010), a number of conclusions can be drawn from this experiment.

The representation approach presented in Section 4.3 works. Even difficult objects such as the legs of the deskchairs were generally perceived well, as can be seen in Figure 4.11a. On a few occasions, however, this was not the case (see Figure 4.11b). This could occur if there was a slight variation in the localization so that the multiple voxels in the vicinity of the legs received an 'occupied' measurement but not enough to reach the upper threshold. Ususally the robot would still drive around the chair but on a couple of occasions when the available space was very tight, nevertheless, the robot would slightly touch the chairs. The bags of students that were on the floor were detected correctly every time.

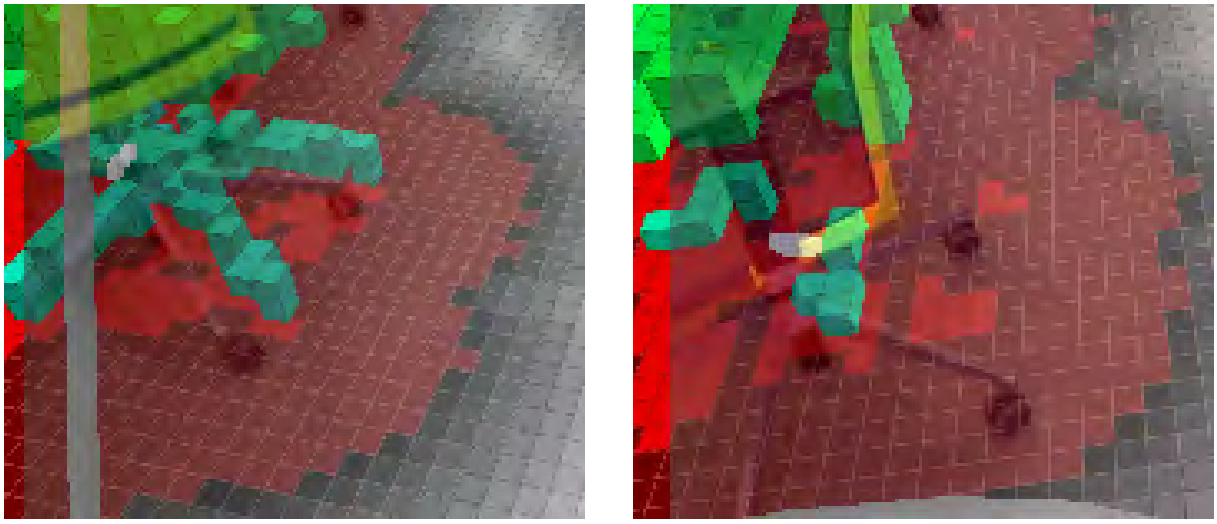
During the experiments, many people were walking around in the library. Especially around lunch time, there were many people taking a close look at the robot and thereby obstructing its path. In many situations, the robot was still able to continue after a replan. On a number of occasions, however, the robot was stuck and a recovery behavior, either clearing the OctoMap in the vicinity of the robot or resetting the entire costmap had to be invoked. Moving people demonstrated a shortcoming of this representation: since the voxels are all independent, the representation might get cluttered if not all voxels of a moving obstacle are cleared. An example of this can be seen in Figure 4.12: a couple of floating occupied voxels just outside the sensor range R_s that are the remainder of a person passing by prevent the robot from planning a path forward (Figure 4.12b), although there is clearly enough space to pass through the chairs on the left and the people on the right, as can be seen in the robot's view (Figure 4.12a).

The behavior demonstrated in the lab experiment was also visible in this experiment, particularly the lower velocities due to narrow passages and occluded areas. The former proved to be effective to avoid collisions, however, the latter proved to be too conservative: due to the



(a) The processed map. **(b)** The default OctoMap.

Figure 4.10: The localization map was post-processed as an intermediate step. People, chairs and bags were removed and overhanging desks were included. Subsequently, this map was extruded to produce the default OctoMap.

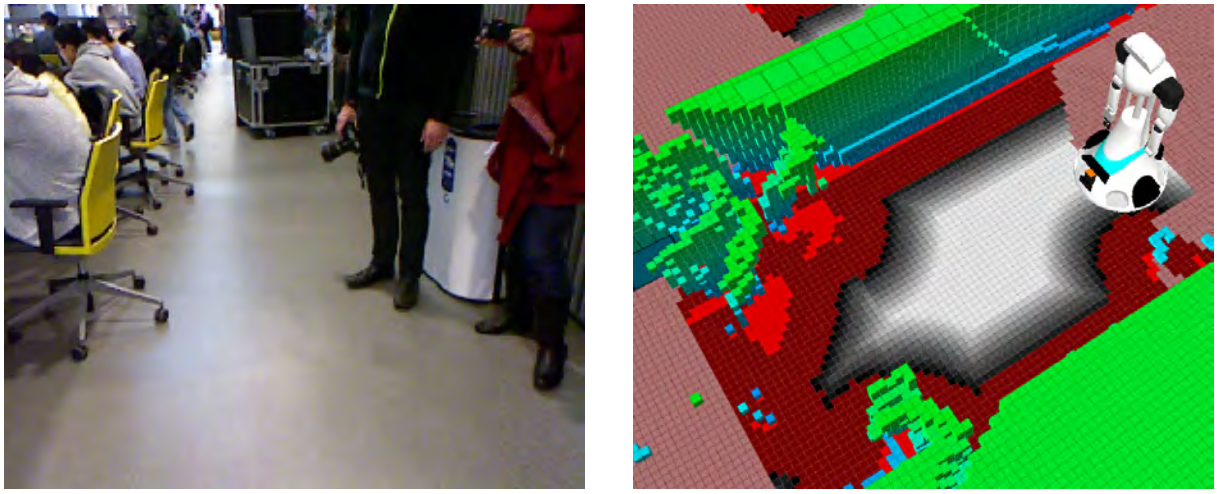


(a) The chair has been mapped correctly. **(b)** The chair has not been mapped correctly.

Figure 4.11: Overlay of the camera image and the OctoMap and costmap. Usually, chairs and their legs were mapped correctly but occasionally the legs did not appear in the OctoMap.

cluttered environment, there were always (small) unknown areas in the vicinity of the robot because the sensors could not see beyond bags or the legs of people or chairs. After inflation, these caused the robot to drive at the minimum velocity v_{thresh} most of the time. To illustrate this effect, the experiment was conducted with three different values for $v_{\text{obs}}^{\max}$. In Figure 4.13, the robot passes through the same part of the environment with $v_{\text{obs}}^{\max} = 1.0$ m/s, $v_{\text{obs}}^{\max} = 0.5$ m/s and $v_{\text{obs}}^{\max} = 0.0$ m/s. In case $v_{\text{obs}}^{\max} = 1.0$ m/s, the robot velocity hardly exceeds the lower bound $v_{\text{thresh}} = 0.14$ m/s, which is even in this environment not sufficient. This is confirmed by the average velocities for the different $v_{\text{obs}}^{\max}$, as can be seen in Table 4.3.

A major difference with the lab experiment was the library floor, which provided less grip than the carpet in the lab experiment. The local planner proved not sufficiently robust for the resulting slip and therefore caused oscillations. This manifested itself especially in the orientation with an amplitude around 0.2 rad/s. However, a larger problem that was caused by both slip and some inaccuracies in the localization map was a large localization error. This error occasionally exceeded 0.8 m and 0.16 rad, which is much more than originally modeled with $\sigma_x^2,\ \sigma_y^2 = 0.1$ m.



(a) View from the robot: it can easily pass through between the people and the chairs.

(b) An example where a moving person has left some 'clutter' in the OctoMap.

Figure 4.12: An example where a moving person has left some 'clutter' in the OctoMap.

Since this method only has a global representation method and does not rely on a local collision map, this error can cause the robot either to get stuck (see Figure 4.14) or to cause collisions. In this experiment, the localization error resulted in imminent collisions on two occasions which is not acceptable in the application. This illustrates the need for a local representation as well, as was already argued in Moore et al (2009).

4.6 Discussion

4.6.1 Parameters

Inflating the obstacle representations often implies that a more or less arbitrary (exponentional) decay function is used. One of the motivations of this research was to eliminate these arbitrary functions and tuning parameters by modeling the various sources of uncertainty of an environment separately. Although many parameters such as the maximum obstacle velocity, the maximum sensor range, maximum update delay and robot velocity and acceleration are well-defined, there are still parameters that are not directly related to a measurable physical quantity. Most notably, the probability decay rate Δ_{dec} and the function relating the probability of collisions to a safe velocity (see Figure 4.4) are selected empirically by doing the extensive simulations presented in Section 4.5.1. Future research should provide the necessary insights to relate these model parameters to measurable quantities as well.

Table 4.3: The average robot velocity $\bar{v}$ for varying $v_{\text{obs}}^{\text{max}}$. If $v_{\text{obs}}^{\text{max}} = 1.0$ m/s, $\bar{v} \approx v_{\text{thresh}}$ which is considered too conservative.

$v_{\text{obs}}^{\text{max}}$ [m/s]	$\bar{v}$ [m/s]
1.0	0.13
0.5	0.17
0.0	0.28

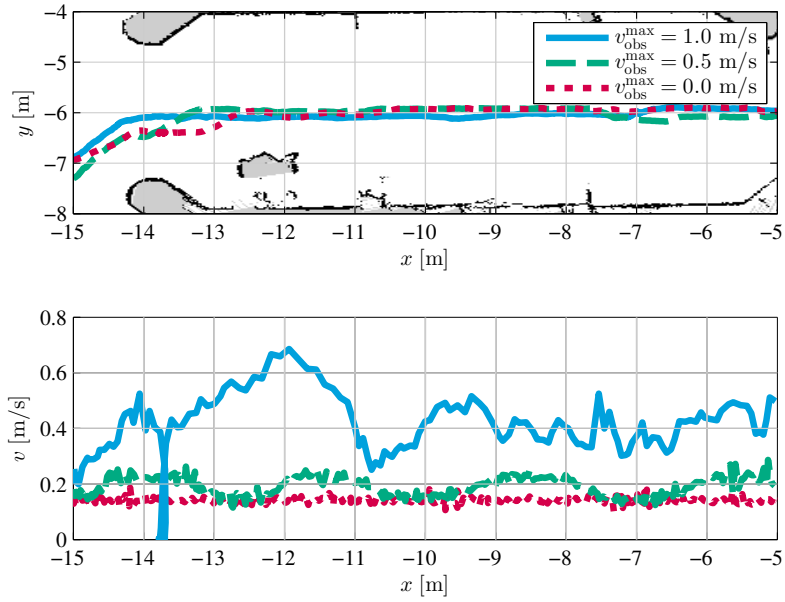


Figure 4.13: Three approximately similar paths (upper plot) result in significantly varying velocities (lower plot) for $v_{\text{obs}}^{\max} = 1.0$ m/s, $v_{\text{obs}}^{\max} = 0.5$ m/s and $v_{\text{obs}}^{\max} = 0.0$ m/s.

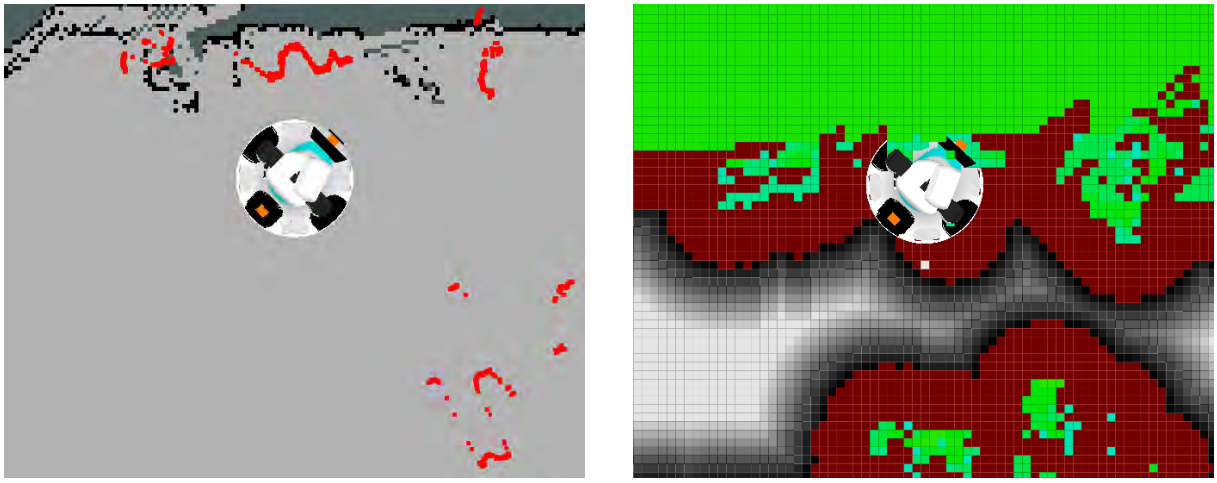
4.6.2 Independency of measurements

As a result of the time dependency of the environment representation in this work, the free space becomes unknown over time but occupied space remains occupied. Hence, if certain voxels of dynamic obstacles are not cleared correctly, these will impede motion planner and might cause the robot to take unnecessary detours or even prevent the robot from reaching its goal. Simply forgetting obstacles over time, on the other hand, might also lead to unsafe situations. This problem is inherent to the independency of the voxels. Therefore, a representation significantly benefits if measurements are associated with the objects in the environment: if the robot detects that an obstacle has moved away from its path, it can immediately clear all associated voxels. Furthermore, this enables to explicitly account for additional properties such as movement of obstacles.

4.7 Conclusions and future work

The proposed volumetric representation allows a robot to safely navigate in a domestic environment. It probabilistically models the occupancy of volumes as sensor measurements are received and, as opposed to typical representations, also if no measurements are received. Furthermore, the probability of moving obstacles appearing on the robot's path from behind occlusions is taken into account. These probabilities are combined with a model of the robot position uncertainty to form a probability of collision. Based on this probability a safe velocity limit is defined.

Extensive simulations have demonstrated that this approach results in the desired behaviour, *i.e.*, the robot moves with velocities up to the maximum of 0.7 m/s if this is safe but slows down in case of narrow passages or uncertain areas of the environment. This has been confirmed by laboratory experiments, where the same behavior was demonstrated and challenging obstacles such as small objects on the floor or overhanging tables were successfully avoided. This approach



(a) A localization error occurs. The amount of laser measurements (red) is limited and these points do not coincide with the map (black).

(b) As a result, the robot 'thinks' it is located in an obstacle.

Figure 4.14: An example where the robot gets stuck due to a localization error.

also worked in a real-world experiment, performed in the university library. However, it was found that the inflation of the uncertain areas was still too conservative. Furthermore, the approach was not sufficiently robust against localization errors.

Several directions can be indicated for improvement of the current approach in future work. A number of conservative assumptions were necessary to ensure safe navigation, in particular i) obstacles may maintain their maximum velocity if they occur on the robot's path, ii) obstacles *never* disappear, hence the probability of occupied obstacles $P > 0.5$ does not decrease over time and iii) dynamic obstacles can emerge at any height from every voxel, thus the maximum occupancy probability of a column is inflated . By including more information, *e.g.*, i) obstacles are not adversary and will try to avoid collisions, ii) certain obstacles such as humans may move away over time and iii) there are no flying obstacles, these assumptions can be relaxed to result in more efficient robot navigation. The representation can be improved by adding explicit obstacle models of, *e.g.*, humans as proposed in Philippsen et al (2006, 2008); Rohrmüller et al (2008), but also of static parts of the environment, *e.g.*, walls. By discriminating in obstacle representations, the probability of their presence can be modeled separately and thus more accurately. Furthermore, an actuated sensor can be controlled to actively reduce uncertainty in the vicinity of the robot instead of always looking forward on the robot's path. This will decrease the collision probability and thereby increase the safe velocity limit. For example, at start-up the robot is then able to directly face the uncertain space in front of its base, while during navigation it can look further ahead. The position uncertainty can also be modeled more accurately. It is now represented with a normal distribution based on an a priori determined maximum variance, while directly using the covariance matrix from the AMCL module is more accurate because this is updated based on the sensor measurements. Additional robustness against localization errors can be added by a local representation, *i.e.*, directly reacting to measurements. Better performance of the total system can be achieved by an improved local planner. Finally, deeper insight in the choice of model parameters is desirable. It would be particularly useful to investigate how to measure the probability decay rate Δ_{dec} of a certain environment and how to relate the safe velocity v_{safe} to the probability of collision $P(n)$. Although those parameters depend on the environment as well as the specific application at hand, a theoretical and extended simulative analysis in different environment set-ups can reduce heuristic tuning of parameters and make the presented method better generalizable.

Chapter 5

Robot navigation using an object-oriented world model

Recently developed robotic software architectures allow for the execution of semantically represented action recipes that can deal with the presence of variation in both environment and robot, see, e.g.*, Di Marco et al (2013); Janssen et al (2013). In the process of grounding the used semantic concepts, knowledge of the environment is needed, which is often represented and integrated in an ad-hoc way, where each of the software components,* e.g.*, for localization, navigation and manipulation, uses and maintains its own specific world model.*
The solution proposed in Blumenthal et al (2013) is to have a common and shared world model where each component has access to relevant parts of the environment. In this reference, however, it is not yet shown how this world model can be integrated with other software modules such as motion planning.
The main contribution of this chapter is to demonstrate how a single, centralized, object-oriented world model can be used for multiple modules in the software stack of a service robot instead of having separate world models for each module. For the specific use-case of application, it is shown how this approach increases robustness and safety: i) while using this object-oriented world model as a basis for the planning representation, ii) while using task context to determine the goal regions with respect to world model objects and iii) while using symbolic knowledge of the task and goal to coordinate the motion planners.
Using this object-oriented world model leads to a planning representation that is less prone to become cluttered with small obstacles that are the result of objects that have not been entirely cleared. Furthermore, the robot can behave differently around different obstacles, improving the trade-off between being quick and being safe. The object representations together with task context allow for the use of goal regions, making the navigation system more robust against obstacles at the navigation goal and more flexible due to the absence of hardcoded waypoints.

5.1 Introduction

Recently developed robotic software architectures allow for the execution of semantically represented action recipes that can deal with the presence of variation in both environment and robot, see, *e.g.*, Di Marco et al (2013); Janssen et al (2013). In the process of grounding the used semantic concepts, knowledge of the environment is needed, which is often represented and integrated in an ad-hoc way, where each of the software components, *e.g.*, for localization,

navigation and manipulation, uses and maintains its own specific world model.

Although this approach is common in the development of robotic applications, it is sub-optimal in the sense that the world models for each module share a number of common issues. Most importantly, all world models need to deal with the data association problem, *i.e.*, associate measurements with false detections, newly appeared objects or existing objects in the world model. This is used to filter out sensor noise and inaccuracies in, *e.g.*, localization, object locations and the kinematic and geometric robot model.

The world models used for task planning and execution are usually *object-oriented* and building such a model is also known as semantic mapping. In this case, data association is performed at the object level, *i.e.*, the outcome of perception algorithms is compared with the objects that are present in the world model. Typically, these object models contain symbolic knowledge about the objects. Furthermore, the objects may have associated motion models depending on the object type, allowing object tracking. These semantic maps are usually not suitable for tasks such as motion planning and navigation because i) typically only objects that have been recognized by a perception module are mapped and ii) geometric properties of objects are often not modeled.

The models for localization and navigation are often purely grid-based. The information these models contain is purely volumetric, *i.e.*, whether voxels are occupied or free. Furthermore, data association is performed at sensor data level. Building such a model is referred to as SLAM (Simultaneous Localization and Mapping). After a mapping phase where a SLAM algorithm is used to build a map, the world is typically assumed to be static, as is mentioned in Milford and Wyeth (2010). As simply relying on recent sensor data is a naive approach due to the presence of dynamic obstacles, it is desired to do *persistent navigation*, *i.e.*, to continuously update the navigation representation. However, as is shown in Chapter 4, a purely geometric approach can then lead to a cluttered world model as measurements are not associated with objects, thereby hindering performance.

Similar issues occur with manipulation. In this case, data association typically implies padding the geometric model of the manipulator and removing all sensor data that is inside this volume. If this padding is insufficient, minor inaccuracies in the kinematic model might cause the robot to falsely conclude it is in collision. On the other hand, adding too much padding will make the robot overconfident. Therefore, manipulation will also benefit from more advanced data association methods.

The geometric world models used for localization, navigation and manipulation could therefore benefit from the symbolic knowledge that is present in the object-oriented world models. Therefore, it is proposed to replace the world models for all software modules by one central object-oriented world model containing both the symbolic information required for the task planning and execution models as well as the geometric information that is required other modules such as localization, navigation and manipulation. In this chapter, we will show how this object-oriented world model can be used for navigation and what the benefits are with respect to purely geometric world models.

5.2 Related work & contribution

5.2.1 World representations

A well-known example of a navigation system for a domestic environment can be found in Marder-Eppstein et al (2010). This uses a voxel grid as a solely geometric world model. The alternative geometric representation introduced in Hornung et al (2013) is more robust against sensor noise due to the probabilistic integration of sensor measurements and more efficient due to the use of

octree data structures. In Coenen et al (2014), this approach is extended with a time-dependent occupancy probability model to model the dynamic characteristics of the environment. Furthermore, this work inflates the uncertain areas to avoid collisions with objects appearing from behind occlusions. Although this approach has been successfully used in various challenges in the 2014 RoboCup@Home competitions (Wisspeintner et al, 2009), it suffers from the independence of the occupied grid cells as discussed in the introduction.

If a representation is purely geometric, it is assumed that all (leaf) nodes are independent and all nodes have the same update models. However, this assumption is not valid in practice and there is no update model that is suitable for all types of objects. This limits the performance of motion planners, which manifests itself in various situations:

- Moving obstacles result in a cluttered environment representation if obstacles are not cleared completely after they have moved.
- Possible movement of objects is not taken into account. The probability of collision when driving closely past a person or a door is taken equal to the probability of collision when driving next to a static wall. As is mentioned in Lu et al (2014), this is not only important for safety but also to obtain socially acceptable behaviors.
- The robot cannot reason about the obstacles it encounters.

The approaches in Philippsen et al (2006); Rohrmüller et al (2008) are examples where dynamic obstacles are explicitly modeled. Both methods work in 2D, with a probabilistic model to estimate future obstacle positions. In Philippsen et al (2006), probabilistic worst-case reasoning is applied to estimate the probability of the robot and obstacle being at the same place at the same moment. In Rohrmüller et al (2008) a risk map is computed. Both approaches only consider obstacles that are in the field of view of the robot, but do not take obstacles occurring from behind occlusions into account.

Examples of semantically annotated world models can be found in Pronobis and Jensfelt (2012); Elfring et al (2013). In the work of Pronobis and Jensfelt (2012), a probabilistic framework for semantic mapping has been presented which abstracts sensor information and integrates it into a semantic map in a probabilistic way. It is shown how this can be used as a topological map for navigation but does not present a complete geometric model that can be used down to the level of local motion planning. The world model in Elfring et al (2013) contains the position of known objects, but these objects have no associated volumetric information and unknown objects are not modeled.

Geometric and semantic information are combined in, *e.g.*, Zender et al (2008); Wurm et al (2011); Chiesa (2013). In Zender et al (2008), a multi-layered spatial representation is introduced. This system has one central world model consisting of different abstraction layers. However, the layered approach hides the semantic knowledge from the low-level components such as motion planning. In Wurm et al (2011), the environment is modeled as hierarchies of octrees. Here, it is mentioned that semantic annotations can be used to facilitate data association or to adapt certain map properties and an example of a representation for tabletop manipulation is shown. It is assumed that that there exist methods to segment a point cloud, whereas in our approach the segmentation is actually aided by using the models that are already present in the world model. The method introduced in Chiesa (2013) also adds semantic information to a 3D occupancy grid. However, this approach does not scale beyond small indoor rooms.

The solution proposed in Blumenthal et al (2013) is to have a common and shared world model where each component has access to relevant parts of the environment. In this reference, however, it is not yet shown how this world model can be integrated with other software modules such as motion planning.

5.2.2 Contribution

The main contribution of this chapter is to demonstrate how a single, centralized, object-oriented world model can be used for multiple modules in the software stack of a service robot instead of having separate world models for each module. For the specific use-case of application, it is shown how this approach increases robustness and safety:

- while using this object-oriented world model as a basis for the planning representation
- while using task context to determine the goal regions with respect to world model objects
- while using symbolic knowledge of the task and goal to coordinate the motion planners

An essential property of this approach is that improvements of the world model directly lead to improvements in other modules such as navigation.

In the following section, a system overview will be presented, followed by a description of the world model in Section 5.4, the motion planners in Section 5.5 and the coordination of these planners in Section 5.6. The results of this approach are presented in Section 5.7 and finally, conclusions and recommendations are given in Section 5.9.

5.3 System overview

An overview of the entire system is given in Figure 5.1. The object-oriented world model is a collection of entities that form the believe state of the robot. These entities can either be prior knowledge, such as the floor, the walls and pieces of furniture that typically do not move or objects that have been mapped by the robot. Volumetric information can be associated with each entity and hence this world model can also be used for navigation. The object-oriented world model is discussed in more detail in Section 5.4.

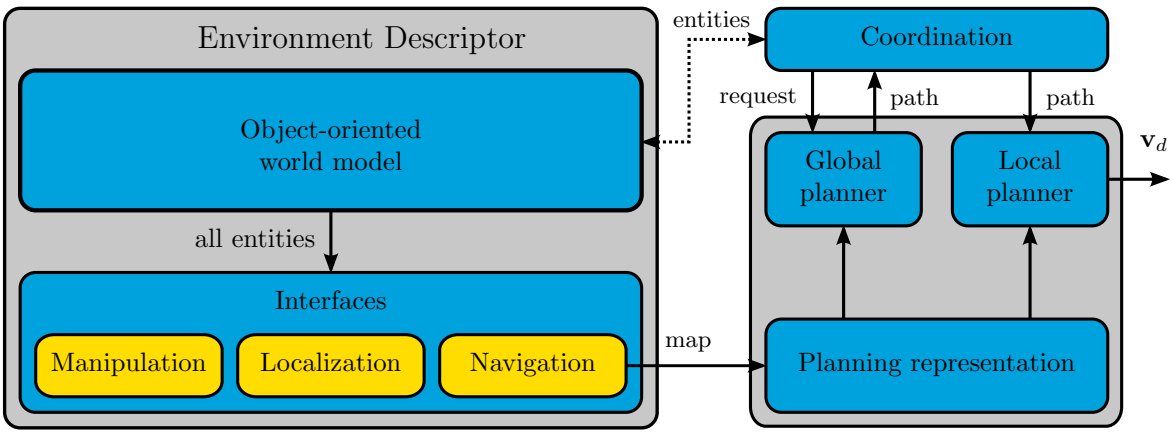


Figure 5.1: System overview with the world model (Section 5.4) on the left with *application-specific* interfaces. The example depicted in this figure is navigation, where both the global and the local planner use the planning representation based on the world model (Section 5.5). The coordination block has a query interface (the dashed arrow) to obtain world model entities that are used to determine the goal region, to decide on recovery actions and possibly to decide when navigation has succeeded (Section 5.6)
.

Since the object-oriented world model contains the walls and static furniture, it replaces the localization map and static navigation map that are typically present. The 3D object models are top-down projected and recent sensor data is added to obtain a planning representation that is subsequently used by a global and a local planner. These will be discussed in Section 5.5. Finally, the coordination of the planners is discussed in Section 5.6.

5.4 Representation

The object-oriented world model that is used here is briefly introduced in Lunenburg et al (2015b). As can be seen in Figure 5.2, the object-oriented world model consists of four main modules. In this figure, the 'world model' block contains the actual data that is used to describe the world.

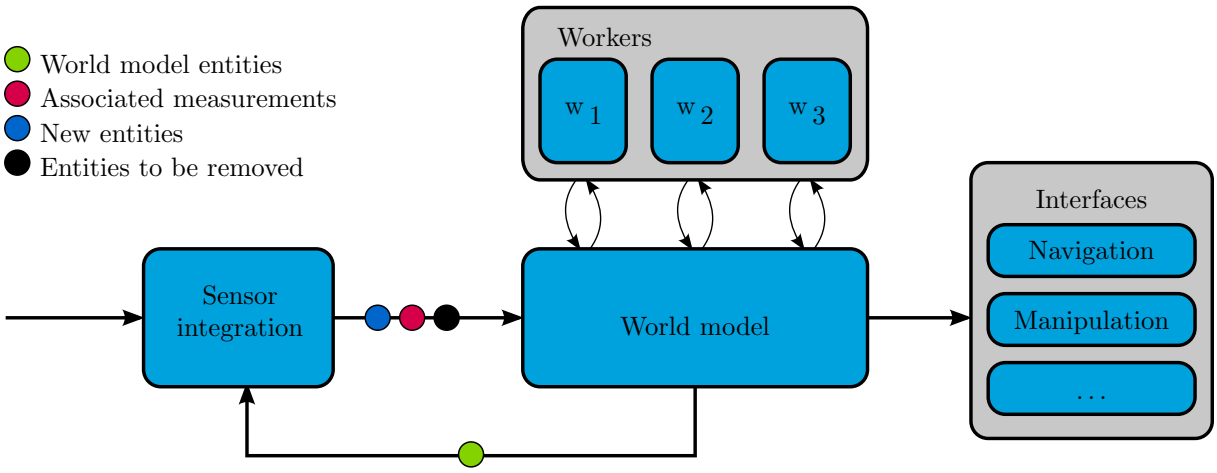


Figure 5.2: Overview of the object-oriented world model. The 'world model' block contains the actual data resulting from the the 'sensor integration' block. 'Workers' such as perception algorithms can add additional (semantic) information to the entities. Finally, the interfaces provide other modules such as navigation, localization etc. with *relevant* information derived from the world model entities.

The sensor integration component processes the data from the robot's sensors such that it can provide the world model with updated shape and position information about i) entities that were already present in the world model, ii) new entities and iii) entities that are no longer present and hence can be removed. This process consists of the following steps:

1. Calculate the sensor pose of the 3D sensor. Currently, only an RGB-D sensor is used but other sensors such as LRFs can be integrated as well
2. Render the world model: a 'virtual' depth image is generated based on the current world model state and the sensor pose (as if the world model was observed by the sensor)
3. Extract features:
   - Calculate normals of the *sensor point cloud*
   - Calculate normals of the *rendered (world model) point cloud*
4. Try to associate each *sensor point* to a *world model point* using the Euclidean point and normal distance. If a *sensor point* cannot be explained (based on a threshold), add it to a *residual point cloud*.
5. Cluster the *residual point cloud* into *cloud segments*
6. Try to associate the *segments* with *current entities* by calculating the overlap of their shapes
   - If a segment *can* be associated with a world model entity, *i.e.*, the overlap between this segment and a world model entity exceeds a certain threshold → *update* the entity shape

- If a segment *cannot* be associated → *add* it to the world model. New entities are represented as extruded polygons.

7. Clear world model entities that are in view but could not be associated in step 4) or 6).

The workers in Figure 5.2 are components that add additional information to the world model entities. At the moment, a number of perception routines has been implemented to perform size matching, color matching, SIFT object recognition, people detection, face detection and face recognition. An aggregation algorithm subsequently determines what is the most probable object type. These perception routines act on the measurements that are associated with world model entities. Hence, the robot does not have to switch perception methods on and off but objects are recognized on the fly. Besides the automatic perception routines, workers can also add information by human robot interaction. Currently, a graphical user interface can be used to label world model entities manually.

The final component in Figure 5.2 is called 'Interface'. For each module that requires information from the world model, a plugin can be implemented to extract the *relevant* information that is required for this module and to convert this into a format that is suitable for this module.

In case of navigation, the 3D world model entities are top-down projected onto a 2D grid map (see Figure 5.3), which serves as the basis for the navigation representation. Here, a cell value not only encodes whether a cell is free, occupied or unknown but also encodes the *type* of obstacle that is occupying this cell, *e.g.*, whether this is a human, a table or a wall. Subsequently, the navigation system can decide how to deal with this obstacle, as will be discussed in the next section.

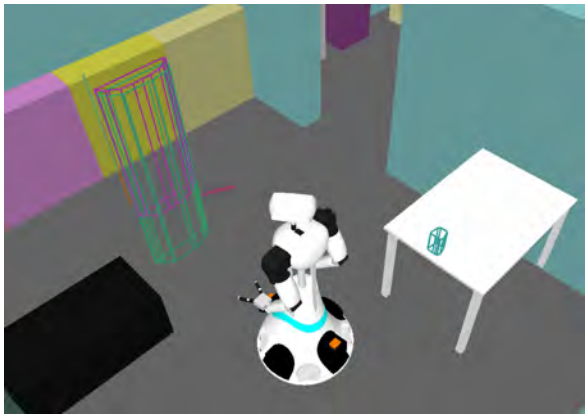
In its current status, the world model does not yet contain the features of previous semantic world models as described in Elfring et al (2014a,b). A future research direction would be to include motion models to allow for object tracking as well as integrating an algorithm to do re-association. The former can be used by navigation to decide, *e.g.*, on which side to pass a moving obstacle while the latter will improve the clearing. This demonstrates one of the advantages of a single, centralized world model that is used for multiple tasks: the navigation module will directly benefit from improvements in the overall world model.

5.5 Motion planning

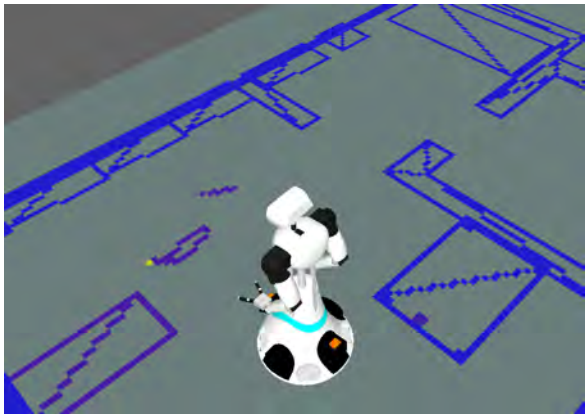
As an example, motion planning is used here as an example to show the benefits of the centralized world model. The various elements of motion planning and their implementations will use the world model in different ways. The 2D occupancy grid resulting from the top-down projection of world model entities is used as the basis for the planning representation. Nevertheless, in further processing steps, it might still be necessary to add additional sensor data, either because it is not processed by the world model or to provide additional safety. The planning representation is used by a global planner that computes a path from start to goal pose and a local planner that computes the velocity commands to track this path. These planners will be discussed in this section. For determining the goal region, recovery options and termination criteria, a query interface is used to directly access the world model entities. This will be discussed in Section 5.6.

5.5.1 Global planner

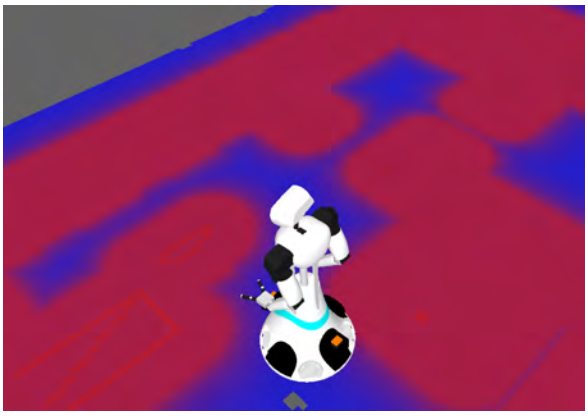
The global planner in this architecture has a fairly common setup. The main difference with the purely geometric approach is that obstacle types are encoded in the occupancy grid resulting from the top-down projection of world model entities, as discussed in Section 5.4. The incoming occupancy grid is used as the first layer in a layered costmap as introduced in Lu et al (2014). In



(a) Visualization of the world model of the robotics lab, with walls in green, cabinets in purple and yellow, a white table, black box and unknown object represented by an extruded polygon.



(b) Map after downprojecting the world model entities. The colors indicate that the walls, cabinets and table (blue) are considered to be stationary, while the box and the extruded polygon are considered unknown (purple).



(c) Costmap resulting from inflating the map in Figure 5.3b.

Figure 5.3: An example of the downprojection of world model entities.

the second layer, data from additional sensors such as LRFs is added to the costmap. These have a larger range and viewing angle than the 3D sensors that are typically used to create the world model as described in Section 5.4. In future implementations, however, it might be beneficial to incorporate this data in the world model as well. Subsequently, the robot footprint is cleared from the costmap and all obstacles are inflated with the inscribed radius of the robot footprint r_{rob}, reducing the search problem to a 2D configuration space where the robot orientation is not taken into account. Further inflation with an exponential decay function

$$c(d) = \begin{cases} c_{\max} & \text{if } d \leq r_{\text{rob}} \\ c_{\max} e^{-s(d - r_{\text{rob}})} & \text{if } r_{\text{rob}} < d \leq r_{\text{infl}} \\ 0 & \text{if } d > r_{\text{infl}} \end{cases} \tag{5.1}$$

allows the global planner to keep some distance to obstacles, whenever possible. The maximum inflation radius r_{infl} as well as the decay rate represented by s depends on the type of obstacle: it is usually safe to move closely past a wall, whereas, *e.g.*, unknown objects might move (see the top plot of Figure 5.4). This is not only beneficial for safety but driving closely past humans is also considered to be socially undesirable, as is discussed in Lu et al (2014). After inflation, a path is searched through the resulting planning representation using an A* planner.

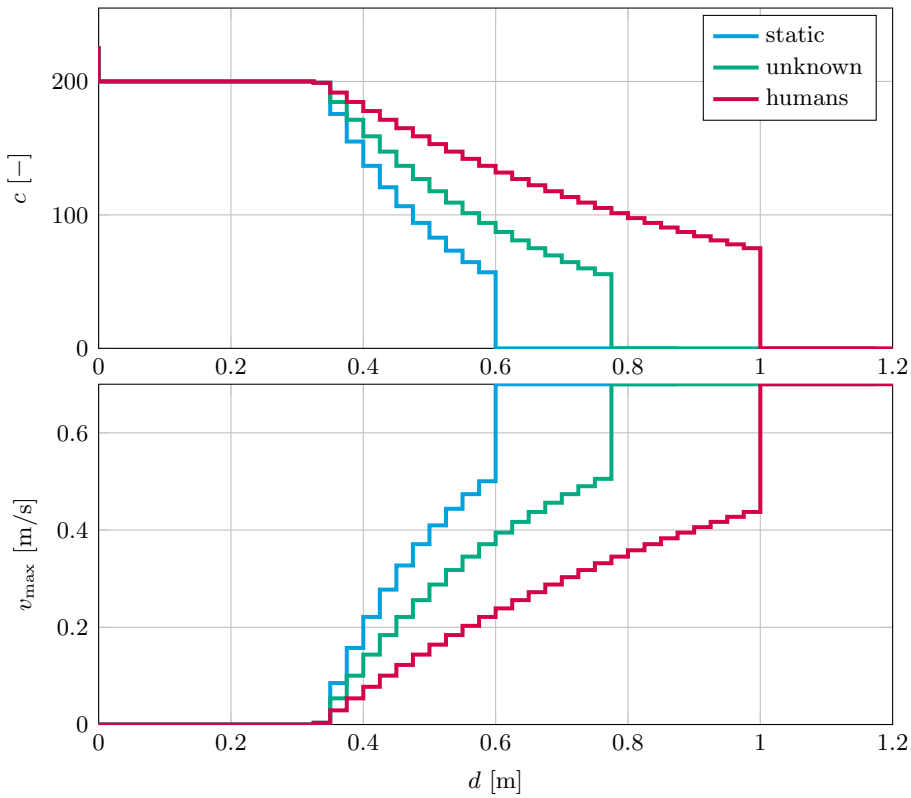


Figure 5.4: Costs c (upper plot) and maximum velocities $v_{\max}$ (lower plot) as a function of the distance to the robot center d for static obstacles (cyan), unknown obstacles (green) and humans (red).

5.5.2 Local planner

For local planning, data of additional sensors is also added to the costmap. As was mentioned in Section 4.7, this is necessary to account for localization inaccuracies and to allow instant reaction to changes in the environment such that the delay time t_d is independent of the world model update rate. The costs are inflated similar to the global planner, *i.e.*, based on the obstacle types. However, the two main ideas to account for a dynamic environment that were presented in Coenen et al (2014) are also incorporated here:

- time-dependency: areas of the environment that have been observed to be free might become occupied over time
- moving obstacles such as people and other robots might emerge from occluded parts of the environment.

To include the time-dependency in the costmap of the local planner, it is tracked at which time a certain cell has last been observed to be free. If it has not been observed for a certain duration t_u, the cell is declared unknown again. The value for t_u depends on the dynamics of the environment. The use of a 2D representation is allowed by assuming that there are no flying obstacles, *i.e.*, a moving obstacle can be detected by a planar sensor. This is computationally less expensive than the 3D representation used in Chapter 4 but more importantly, a 2D laser range finder with a larger range and wider view angle than a typical RGB-D sensor such as a Kinect camera can be used, making this approach less conservative.

A second assumption is that moving obstacles always have a certain minimum size $d_{\min}$. To account for moving obstacles emerging from behind occlusions, the unknown areas are dilated with radius $d_{\min}$ and subsequently inflated with a radius d_{obs}. Similar to the approach in

Table 5.1: The parameters for the various phases of navigation.

	α	β	γ	δ
Default	1	0	1	1
Arriving	1	0	1	0
Aligning	0	1	1	0

Chapter 4, d_{obs} should be an estimation of the distance that an obstacle can travel in the time it takes the robot to come to a stop.

The carrot planner that acted as a local planner in Chapter 4 proved to be not sufficiently robust in the real-world experiments (Section 4.5.3). Therefore, an alternative local planner is used in this chapter, based on the Dynamic Window Approach which was first introduced in Fox et al (1997). Similar to Chapter 4, the maximum velocity v_{max} is limited based on the obstacle cost c. Together with the inflation that is dependent on the obstacle types, this results in a behavior that the robot is allowed to drive at relatively high velocities near a wall or a static obstacle but has to slow down around humans. An example is shown in Figure 5.4: in the upper plot the cost decay function is shown, while the lower plot shows the corresponding maximum velocity. A candidate trajectory will be discarded if it exceeds this maximum velocity.

Compared to the standard DWA, state behavior has been introduced because testing the DWA planner showed that it was difficult to find a set of parameters that is suitable in each situation. For example, the desired behavior when moving along the global path is different from the desired behavior when precisely positioning the robot at the goal location or when pulling out of a tight area. In general, the objective function J of a candidate trajectory with velocity $\mathbf{v}$ can be defined as:

$$J(\mathbf{v}) = \alpha J_{\text{g}}(\mathbf{v}) + \beta J_{\text{p}}(\mathbf{v}) + \gamma J_{\text{a}}(\mathbf{v}) + \delta J_{\text{v}}(\mathbf{v}) \tag{5.2}$$

where J_{g} favors trajectories that progress further along the global path, J_{p} favors trajectories that stay close to the global path, J_{a} penalizes error in the heading direction or alignment with the global path and J_{v} penalizes backward and sideways motions. The parameters α, β, γ and δ can be used as scaling parameters. The three states that were therefore introduced are:

- Default: proceed along the global path as much as possible while minimizing unnatural sideways motions. By driving forward as much as possible, the view of the camera and LRF is maximized, thereby optimizing the input to the world model.
- Arriving: when the robot is close to its goal, the distance to the global path is less important and sideways and backwards motions are penalized less severe.
- Aligning: When the difference between the heading and the direction of the global path exceeds a certain threshold, the robot first safely aligns while maximizing the distance to obstacles before proceeding to its default behavior.

The resulting parameters can be found in Table 5.1.

5.6 Coordination

Navigation typically is not a task on itself but rather part of a greater mission or action recipe, see, *e.g.*, Tenorth et al (2013); Janssen et al (2013). The robot, *e.g.*, has to explore the environment, search for a specific object, go to a person or position itself to manipulate an object.

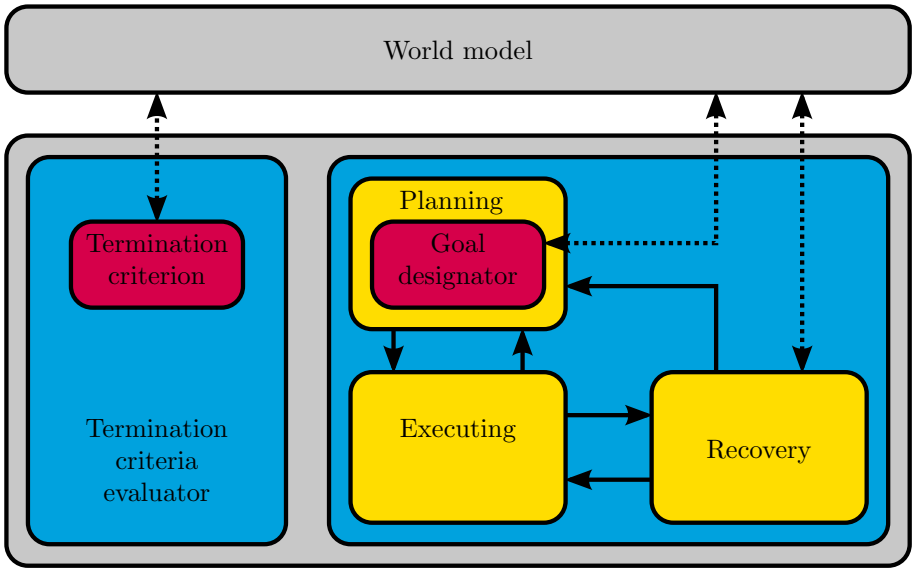


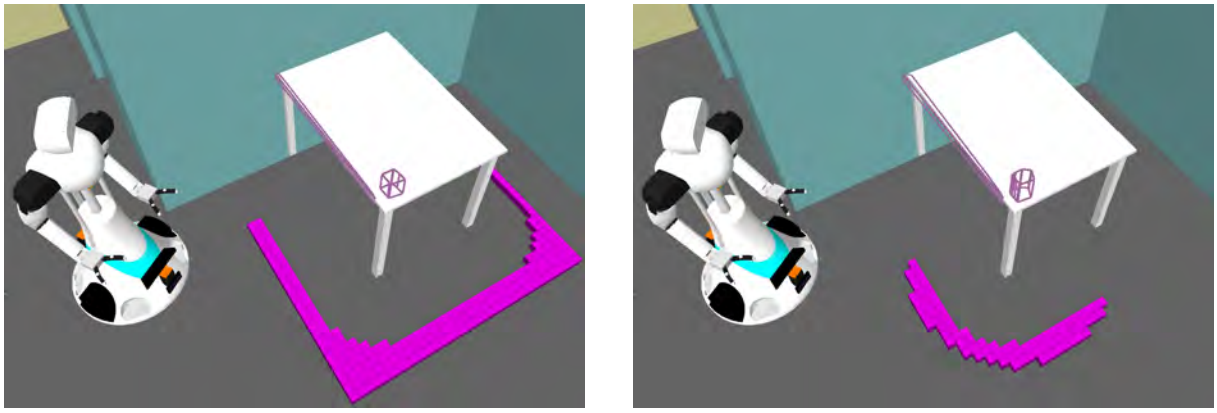
Figure 5.5: Overview of the coordination state machine. The termination criterion and goal designators are task-dependent arguments. The dashed lines represent a query-interface to the world model to obtain specific information, *e.g.*, about the shape or the affordance of an object. Both blue processes run concurrently: as soon as the termination criterion is met, the navigation task has succeeded.

These tasks can all be seen as subclasses of 'navigation'. For these tasks, the 2D occupancy grid is not sufficient. More specifically, it affects the 'Planning' (what is the goal constraint?) and the 'Termination criteria evaluator' (when is the task finished?) and the 'Recovery' (what should the robot do in case its path is obstructed?) blocks in Figure 5.5.

5.6.1 Searching a global plan

The implementation in Coenen et al (2014) already had a semantic query interface to the world model, but adding additional decision parameters often led to complicated and error-prone constructions. A common example appeared when the robot was searching for an object: the executive had to query the world model for search locations that had not been visited and assert which locations had been visited, storing *task-specific* knowledge (the locations that have been visited) in the world model. A more robust solution is the use of designators as is also proposed in, *e.g.*, Beetz et al (2010). These designators contain the task-specific knowledge and are only resolved at the moment that the goal location is required. During resolution, it is possible to query the world model for the up-to-date pose and shape of an entity in the world model.

Using this pose and shape, a goal for the motion planners has to be derived. Although the concept of a *goal region* is long-known in motion planning, see, *e.g.*, Latombe (1990), navigation goals are still typically defined as waypoints in the fixed world frame, possibly with a 'goal area radius' and an independent orientation constraint, *i.e.*, the desired orientation with respect to the world frame is independent of the position within the goal region. When taking the task context into account, nevertheless, the shape and size of the goal region $\mathcal{G}$ as well as the desired orientation within that region depend on the task at hand. In our approach, navigation goals are therefore defined as constraints with respect to *world model entities*. The constraint depends on both the pose and the shape of a world model entity. As a result, navigation is more robust because the probability that the entire goal region is occupied with obstacles is smaller than the probability that a single waypoint is occupied. Two examples can be seen in Figure 5.6: in the left figure, the goal region is defined such that the distance from the robot center to the table edge is approximately 70 cm, *e.g.*, when the robot is searching for objects. Although not visible



(a) Visualization of the goal region $\mathcal{G}$ to search the table for objects.

(b) Visualization of the goal region $\mathcal{G}$ to grasp the object depicted with the extruded polygon on the corner of the table.

Figure 5.6: Examples of goal regions $\mathcal{G}$ depicted in purple. Note that the these regions can never be approximated with a single waypoint and a radius.

in this figure, the orientation constraint is defined such that the robot faces the middle of the table, regardless of its position in the goal region. The right figure shows the goal region which is a circle with radius $0.45 < r < 0.60$ around the center of the object, which is a convenient distance for this robot to grasp it. Note that in this case, a suitable orientation constraint is also defined.

The corresponding position and orientation constraints $\mathcal{G}$ are subsequently computed based on the purpose of navigation and are sent to the global planner. The A* planner searches a path to $\mathcal{G}_{\text{lc}} = \{\mathbf{c} \in \mathcal{G} | c(\mathbf{c}) < c_{\text{min}}\}$, *i.e.*, the subset of the goal region of which the costs $c(\mathbf{c})$ are below threshold c_{min}. If no path can found, the search is repeated towards $\mathcal{G}_{\text{hc}} = \mathcal{G} \setminus \mathcal{G}_{\text{lc}}$. If a path is returned, it will be send to the local planner that will start moving the robot along this path.

5.6.2 Executing a plan

If a robot is navigating for a purpose, it is not necessarily done when reaching a waypoint, *e.g.*, when following a person or when searching for a specific object. Therefore, the termination criteria evaluator runs concurrently with the other blocks. Depending on the task at hand, information of the world model is required to evaluate the termination criterion. Whenever this task-dependent termination criterion is met, the robot has completed the navigation subtask. This is similar to the use of fluents, as is described in, *e.g.*, Janssen et al (2013), in that it can stop the navigation thread.

5.6.3 Recovery

Despite the effort to keep the environment representation up-to-date as much as possible, a global plan can always become invalid, causing the robot to get stuck (Marder-Eppstein et al, 2010). Therefore, recovery behaviors are required. A straightforward solution is replanning, but while taking an alternative path is a good solution in some cases it might also lead to unnecessary detours. Therefore, a flexible architecture has been implemented in which the robot considers its options based on the *type* and hence the *affordances* of the obstacle it encounters. Here, 'affordance' means what something or someone can offer or afford to do (Pandey and Alami, 2013). This approach consists of three steps:

1. Query the world model for information about the object that is blocking and determine

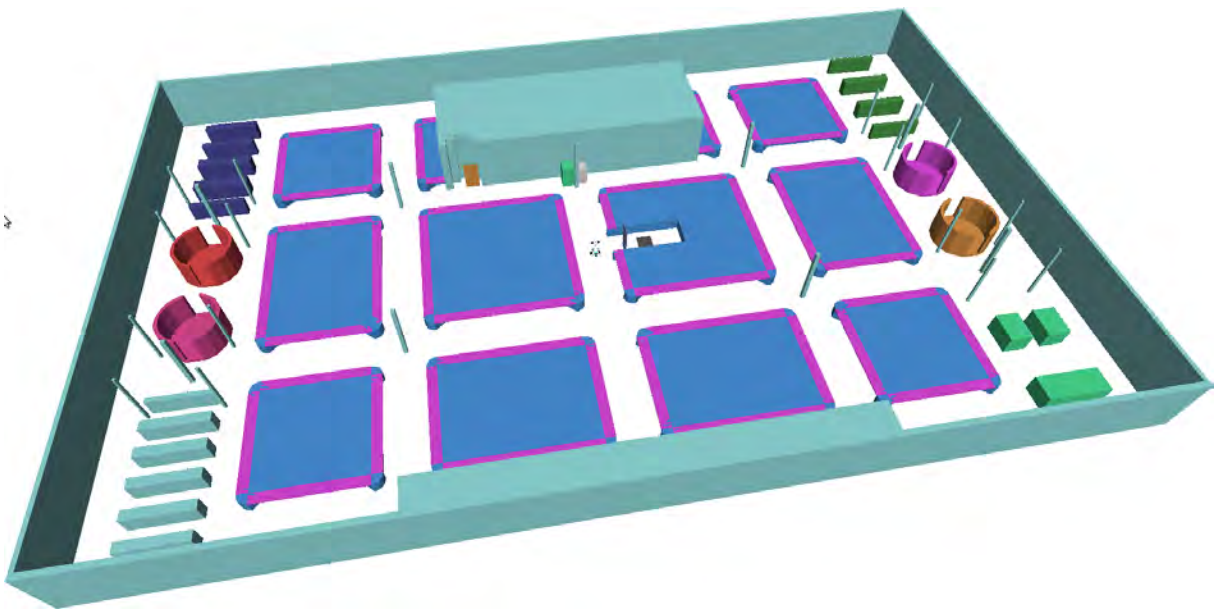


Figure 5.7: The default model of the university library.

the resulting recovery options.

2. Compute an estimation of the costs c_i^{rec} associated with each option i.

3. Execute options based on the estimated costs until the robot is able to continue its path.

Currently, three recovery options have been implemented: i) waiting for a pre-specified duration, ii) re-planning and iii) in case of a human blocking its way, the robot may ask this person to step aside. Waiting and asking a human to step aside have fixed costs, while the costs of re-planning depend on the additional time that is estimated for the alternative path. In the future, more affordances and corresponding recovery options will be added to the world model objects such that obstacles can also be removed by grasping or pushing them, similar to Levihn et al (2014).

5.7 Experiments

The navigation system in this chapter is verified using the AMIGO robot, a domestic service robot developed by Eindhoven University of Technology. AMIGO (see Figure 5.8) has a four-wheeled omni-directional base that is capable of navigating through wheelchair-accessible areas. Its torso with one translational degree of freedom is equipped with two anthropomorphic arms to perform manipulation tasks. A Hokuyo UTM-30LX Laser Range Finder, positioned at the front-side of its base, provides a 220° view at 30 cm above the ground. A second LRF is placed on the torso. AMIGO localizes itself on the static entities in the world model using Monte Carlo localization. A Microsoft Kinect mounted with a pan-tilt unit on top is used to provide three-dimensional point cloud data.

Similar to Chapter 4, a large scale experiment in the university library was performed. Compared to a testlab, the university library is a challenging environment due to its size and the presence of obstacles such as desk chairs as well as the presence of people other than the operators of the robot. A model of the 58 m $\times$ 38 m environment was made (see Figure 5.7), consisting of the walls and columns in green, the desk areas in purple, four circular relax areas

Figure 5.8: The AMIGO robot navigating through the university library. (Photo courtesy of FM-Fotografie.)

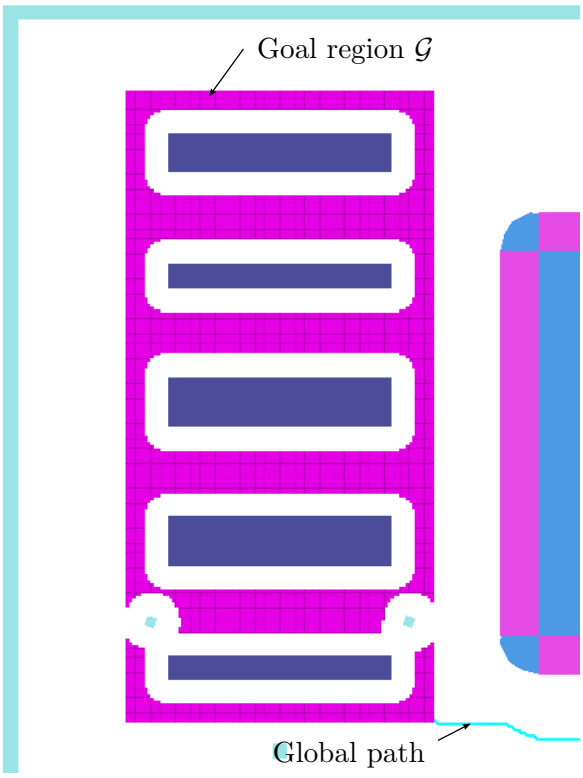
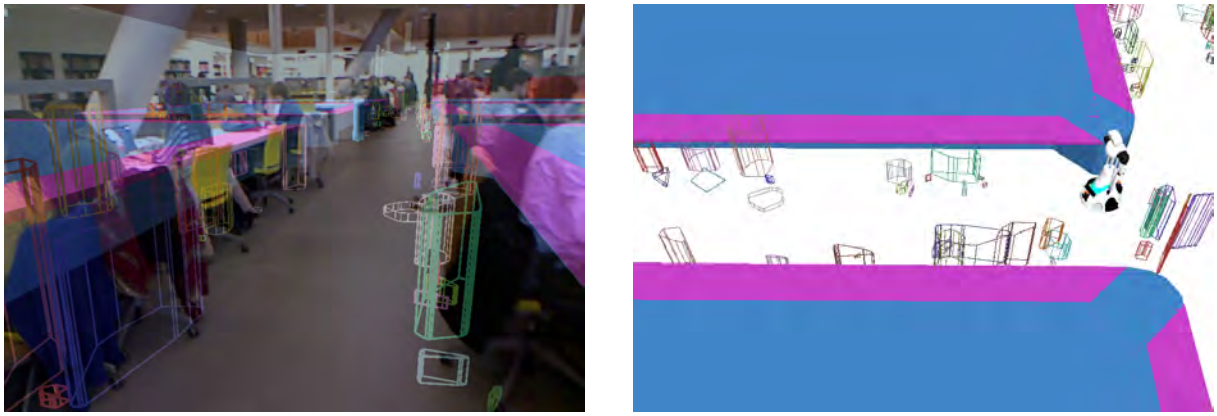


Figure 5.9: Top-view visualization of the goal region $\mathcal{G}$ for five book shelves (see the upper left corner of Figure 5.7). Since the lower right corner is far from any obstacle but close to the robot, the global path is planned towards here.

and a number of bookshelves in the corners. These objects are all static, whereas non-stationary (furniture) objects such as chairs were not modeled. Hence, these have to be mapped by the robot while navigating. Concerning the 'workers' as discussed in Section 5.4, only the people detection modules are relevant for this experiment. These rely on detecting the contour of the shoulders and the head as well as OpenCV face detection.

A query to the world model was defined that returned all entities that are known a priori. These could be used directly for navigation, without defining additional waypoints. The goal regions were defined as if the robot was showing visitors around. An example is shown in Figure 5.9, where the large pink area depicts the goal region for the book shelves shown in purple. Note that the robot will typically drive to the part of the goal region to which the shortest global path is found. In this experiment, the robot either showed the user a location that was selected using a browser-based graphical user interface or selected a goal at random. This way, the robot covered a total of 3.2 km.

In Chapter 4, an important issue was the OctoMap world representation getting cluttered because obstacles were not cleared entirely. The object-oriented approach described in Section 5.4 proved to be much more robust in this respect. Despite the absence of aggressive recovery behaviors such as clearing the space around the robot or resetting the entire world model to the default state, the robot was able to drive around in the presence of people walking around. The remaining obstacles were mapped as can be seen in Figure 5.10. Labeling of the world model entities can be improved: during navigation, the robot looked to the ground to make sure it does not miss any small obstacles. As a result, however, the robot did not see the shoulder and



(a) Overlay of the camera image with the world model.

(b) World model view.

Figure 5.10: Visualization of one of the corridors, with static objects solid and mapped objects as extruded polygonal wireframes.

head contour or the face of a human and hence humans were mostly considered as unknown obstacles.

The effect of the type-dependent obstacle costs as introduced in Section 5.5 can be seen in Figure 5.11. The robot maintains a larger distance to human obstacles and lowers its velocity, compared to static obstacles. This proved particularly useful when driving close to the desk chairs: even if the legs were not mapped perfectly accurate, the robot would keep sufficient distance to avoid collisions. In general, the DWA planner performed much better than the carrot planner in Chapter 4, not showing the oscillatory behavior mentioned in Section 4.5.3. The behaviors introduced in Section 5.5.2 were clearly visible with the robot moving forward as much as possible, while utilizing its holonomic capabilities whenever needed.

Due to the poor performance of the people detection, the robot usually only deliberated between waiting and executing a re-plan. In Figure 5.12 a situation is shown where the global path was blocked by a human. Since the alternative path was much longer, the robot first asked the person blocking its path to move away before executing the re-plan. In the entire library experiment, it happened only once that no global path to the goal could be found because there were too many obstacles present. In future work, more elaborate motion models of obstacles (see, *e.g.*, Elfring et al (2014a,b)) can be used to overcome this issue.

5.8 Discussion

The world model introduced in Section 5.4 is still a basic implementation and many of the ideas introduced in, *e.g.*, Elfring et al (2014a,b) concerning tracking and re-association of objects has not been integrated. These additions can greatly improve the performance: if, for example, the robot recognizes a person, *i.e.*, re-associates this person with a previous entity, it can update the position if this entity instead of creating a new one. A similar result can also be obtained when other sensors such as LRFs are incorporated, since these allow object tracking at a larger distance. Furthermore, the perception routines that act as 'workers' are only able to recognize people and small objects. As a result, many of the entities remain 'unknown' and hence lack semantic knowledge.

Similar to Chapter 4, obstacles are not cleared over time. As a result, it happened once that the robot could not reach its goal because no feasible global path could be found, as was

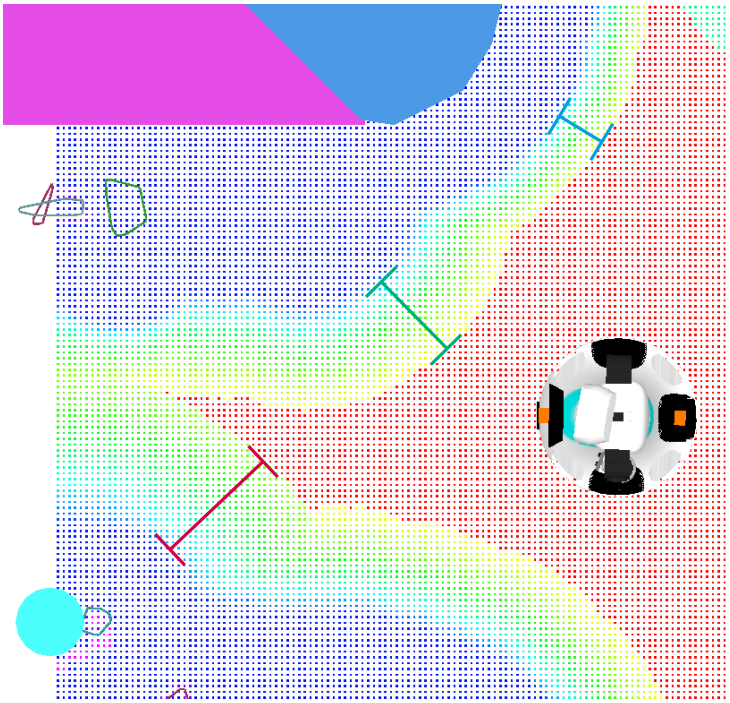


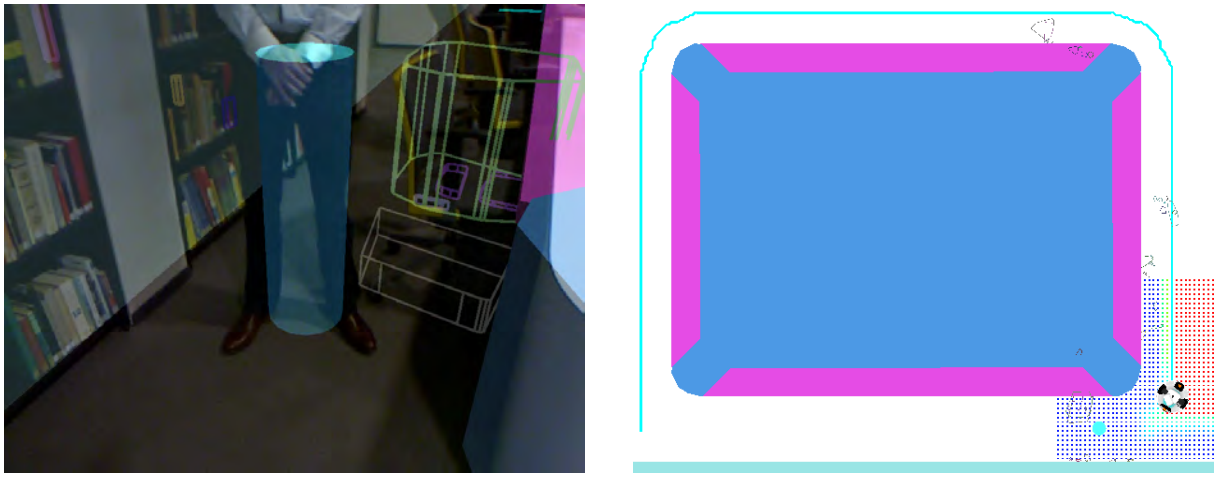
Figure 5.11: The obstacle costs in the vicinity of the robot. One can see that the desk (blue ruler) is inflated with a smaller radius than the unknown objects (green ruler, resulting from the unknown entities at the upper left corner and recent sensor data) and the human denoted by the cyan cylinder (red ruler).

mentioned in Section 5.7. To address this problem, a 'persistance' property could be added to objects, which would be an estimation of the duration D that an object of a certain type remains in place. If this object is still present in the world model D after it has been seen by the robot, the global planner would be allowed to plan a path through this area (at higher cost). Upon arrival, the robot would either confirm that the object is no longer there or execute a recovery behavior.

Besides the possible improvements in the world model, much is expected from re-using this approach for other tasks such as manipulation. The integration of this world with a whole-body planner and controller is currently being investigated. As was briefly mentioned in Section 5.7, localization already used this world model instead of a separate localization map. This worked well, but further experiments are required to compare the performance of both approaches.

5.9 Conclusions & future work

This chapter shows how a volumetric, object-oriented world model can be used as a central source of information. As an example application, the centralized world model served as a basis for the navigation representation. Using the object-oriented world model leads to a planning representation that is less prone to become cluttered with small obstacles that are the result from obstacles that have not been entirely cleared from the representation. The robot could navigate through the library with many people walking around and obstacles at unexpected positions without aggressive recovery behaviors such as clearing the space around the robot or resetting the world model to a default state. Furthermore, the robot can behave differently around different obstacles, being more cautious around people compared to static obstacles. This improves the trade-off between being quick and being safe. In general, the DWA local planner performed very well during the experiments and did not show the oscillatory behavior



(a) Overlay of the camera image with the world model, where the cyan cylinder indicates that the obstacle has been recognized as a person.

(b) World model view with an alternative path in cyan, which is much longer than the straight path towards the goal.

Figure 5.12: The obstacle blocking the global path is a human. Since the length of the alternative plan is much longer than the original plan, the robot first asks the person to step aside. As the person remains at his position, the re-plan is executed anyway.

that was seen using the carrot planner in Chapter 4.

The object representations together with task context allow for the use of goal regions, making hardcoded waypoints obsolete. This was also shown in the library experiment, where all goal regions were deduced from the world model entities. Furthermore, this makes the navigation system more robust against obstacles at the navigation goal.

The main advantage of using a central world model over a completely separated models for, *e.g.*, navigation and task execution, is that all modules such as navigation directly benefit from improvements in the world model. Here, a number of improvements are identified that directly fit in our architecture:

- Improved object motion models. During the experiments, it happened once that all possible paths were blocked by obstacles so that the robot could not plan a global path. If all objects had an associated motion model (see, *e.g.*, Elfring et al (2014b)) an estimation could be made of which obstacles might have moved. As a result, the robot could get a global plan anyway and see if this is still blocked.

- Add tracking of entities to the world model. This allows to do a prediction of the motion of moving obstacles, which can be used by the robot to decide, *e.g.*, whether it should slow down or pass a person on the left or right.

- Detection of large objects: if large objects such as furniture are correctly recognized, the robot can adapt its distance to these obstacles as well as its velocity accordingly.

- More recovery behaviors can be added if the recognized objects have more affordances. Straightforward examples are pushing objects aside and picking objects up to place them out of the way.

A further improvement is to improve the determination of the gaze direction such that people are more easily recognized. A final recommendation is to use the centralized world model for more applications, starting with manipulation and thoroughly evaluating the performance of localization.

Chapter 6

Conclusions & recommendations

In this chapter, the main conclusions of this research are summarized, followed by recommendations for further research.

6.1 Conclusions

In Chapter 2, SERGIO was introduced: a state-of-the-art, *modular open hardware* design of a service robot featuring a holonomic platform, a 2-DoF upper body and two 7-DoF arms. Due to the use of Mecanum wheels, the robot is fully holonomic with only four wheel motors. In Section 2.7.1, it was shown that the independent wheel suspension allows SERGIO to drive accurately in a straight line, even on rigid, uneven surfaces. For example, at a forward velocity of $v_x = 0.75$ m/s, the velocity error in y-direction of AMIGO is $e_{vy}^{\max} = 0.28$ m/s, $e_{vy}^{\mathrm{RMS}} = 0.077$ m/s while for SERGIO this is only $e_{vy}^{\max} = 0.025$ m/s, $e_{vy}^{\mathrm{RMS}} = 0.009$ m/s. This also holds for other velocities and when driving sideways. Small thresholds such as doorsteps are not a problem, even when approached under an angle. The ground clearance of 17 mm is sufficient in practice, as these robot are supposed to operate in wheelchair-accessible areas.

The robot's upper body has a nearly vertical DoF by using a four-bar mechanism and a hip pitch joint. The combination of a nearly vertical DoF with a rotation allows the robot to pick up objects located further from the edge of a table and from the ground, as was shown in Section 2.7.2. Building the first of two arms as introduced in Section 2.4.2 has just been completed. Due to the presented improvements the performance with respect to the current manipulators is expected to improve significantly. The arm is suspected to move more smoothly due to the use of spiral instead of straight-cut bevel gears. Furthermore, the backlash is significantly reduced and the improved placement of the absolute position sensors should make a homing procedure obsolete. Finally, the modularity of the robot has greatly improved serviceability. Assembling and separating base and torso can be performed in a minute.

In Chapter 3, a recipe was presented to select a representation class, a search algorithm class and an appropriate approach to combine these for a motion planner for a mobile robot that moves on a ground plane. This recipe considers all requirements that are relevant for a specific application.

The recipe is verified with three use cases where existing motion planners are successfully applied in practice. The recipe's outcome resembles the existing motion planners and the recommendations from the recipe reveal the points for improvement that are indicated by the original authors. It can therefore be concluded that the recipe is suitable for the selection of an appropriate motion planning approach, representation class and search algorithm class.

A volumetric environment representation based on OctoMaps was introduced in Chapter 4.

Here, a time-dependent occupancy probability model was implemented so that 'free' parts of the environment become 'unknown' again over time if they are not observed. Furthermore, uncertain parts of the environment were inflated to account for obstacles coming out of occluded parts of the environment.

This proposed volumetric representation allows a robot to safely navigate in a domestic environment. Extensive simulations have demonstrated that this approach results in the desired behavior, *i.e.*, the robot moves with velocities up to the maximum of 0.7 m/s if this is safe but slows down in case of narrow passages or uncertain areas of the environment. The simulation results have been confirmed by laboratory experiments, where the same behavior was demonstrated and challenging obstacles such as small objects on the floor or overhanging tables were successfully avoided. Furthermore, this approach to represent the environment also worked in a real-world experiment, performed in the university library. However, it was found that the inflation of the uncertain areas was still too conservative and that the approach was not sufficiently robust against localization errors. Finally, it was found that the independence of the voxels and the absence of semantic information about the environment and the obstacles therein limit the performance.

These shortcomings led to the development of a volumetric, object-oriented world model in Chapter 5. It was concluded that having one, centralized (shared) world model can improve performance with respect to a system architecture where each software module, *e.g.*, task planning and execution, localization and motion planning, has to maintain its own world model.

As an example, it was shown how a navigation system can benefit from the use of task context and a symbolically annotated object-oriented world model. Using the object-oriented world model leads to a planning representation that is less prone to become cluttered with small obstacles that are the result of objects that have not been entirely cleared from the representation. The robot could navigate through the library with many people walking around and obstacles at unexpected positions without aggressive recovery behaviors such as clearing the space around the robot or resetting the world model to a default state. Furthermore, the robot showed different behavior around different obstacles, being more cautious around people than around static obstacles. This improves the trade-off between being quick and being safe. The object representations together with task context allow for the use of goal regions that are deduced from the world model entities, making the entire system more robust against obstacles at the navigation goal and more flexible due to the absence of hardcoded waypoints.

Compared to the approach in Chapter 4, the assumptions on moving obstacles were relaxed. By assuming that obstacles emerging from behind a wall were not flying and hence could always be detected by a 2D sensor and by assuming a minimum obstacle size, the robot could reach its maximum velocity when driving through the university library. Nevertheless, the library does not have many sharp corners and a fixed inflation radius was used, so more experiments are required to draw a final conclusion.

Concerning the local planners, the DWA planner with state behavior used in Chapter 5 performed much better than the carrot planner in Chapter 4. Even though the library floor is not ideal for AMIGO, the robot could drive fairly smoothly. Most of the time, the robot shows natural behavior, *i.e.*, moving in a forward direction as much as possible, while utilizing its holonomic capabilities when starting to execute a path or when accurately positioning at its endgoal.

6.2 Recommendations for future research

Despite the advances discussed in Section 6.1, there is still a long road ahead to introduce domestic robots in our homes. Based on the work in this thesis, a number of recommendations

for future research is proposed in this section.

Before SERGIO is able to perform useful tasks, our existing software stack needs to be updated. The main issues are configuring the motion planners and refactoring the implementations of skills such as grasping and placing. Due to the differences between AMIGO and SERGIO, this latter issue is challenge because it puts additional demands on the generality and configurability of the software. Among the challenges that are expected are navigation, due to the large and rectangular footprint, whole-body control due to the ball screw drives and four-bar mechanism and camera localization, due to the rocking motion of the robot that is introduced by the compliant wheel suspension. As soon as SERGIO is performing navigation and pick-and-place tasks, it is possible to thoroughly assess its performance in practice.

Compared to AMIGO, SERGIO is a step forward in terms of modularity. Nevertheless, it is a prototype with a number of labor intensive parts, making it still expensive. Future hardware developments should lead to a dramatic reduction in the costs, similar to the developments with our soccer robots (see Appendix B). Finally, the design of SERGIO is ready to be released, *i.e.*, to make all CAD drawings, electrical schemes, part lists etc. available on the ROP wiki.

To verify the recipe introduced in Chapter 3 more extensively, it must be applied to more robots which already feature a motion planner that functions well in practice. As a part of this, the recipe can be applied to the robots of team Tech United Eindhoven (Tech United Eindhoven, 2014) and the resulting planner designs can be evaluated in the RoboCup competitions (Kitano et al, 1997). Furthermore, the practical usability of this recipe would increase if a tool is developed that contains a database of the selected references and queries the decision trees of the recipe for an appropriate combination of motion planning algorithms.

Concerning the environment representation introduced in Chapter 4, several directions of future work were identified. The most important issues were addressed in Chapter 5 but a number of recommendations can still be made. Firstly, an actuated sensor can be controlled to actively reduce uncertainty in the vicinity of the robot instead of always looking forward on the robot's path. This will decrease the collision probability and thereby increase the safe velocity limit. For example, at start-up the robot is then able to directly face the uncertain space in front of its base, while during navigation it can look further ahead. Furthermore, deeper insight in the choice of model parameters is desirable. It would be particularly useful to investigate how to measure the probability decay rate Δ_{dec} of a certain environment and how to relate the safe velocity v_{safe} to the probability of collision $P(n)$. Although those parameters depend on the environment as well as the specific application at hand, a extended simulation analysis in different environment setups can reduce heuristic tuning of parameters and make the presented method better generalizable.

The main advantage of using the central world model in Chapter 5 over completely separated models for, *e.g.*, navigation and task execution, is that navigation directly benefits from improvements in the world model. Here, a number of improvements are identified that directly fit in our architecture:

- Improved object motion models. During the experiments, it happened once that all possible paths were blocked by obstacles so that the robot could not plan a global path. If all objects had an associated motion model (see, *e.g.*, Elfring et al (2014b)) an estimation could be made of which obstacles might have moved. As a result, the robot could get a global plan anyway and see if this is still blocked.

- Add tracking of entities to the world model. This allows to do a prediction of the motion of moving obstacles, which can be used by the robot to decide, *e.g.*, whether it should slow down or pass a person on the left or right.

- Detection of large objects: if large objects such as furniture are correctly recognized, the robot can adapt its distance to these obstacles as well as its velocity accordingly.

- More recovery behaviors can be added if the recognized objects have more affordances. Straightforward examples are pushing objects aside and picking objects up to place them out of the way.

A further improvement is to improve the determination of the gaze direction such that people and objects are more easily recognized.

Now that the benefits of a volumetric, object-oriented world model have been demonstrated for the use case navigation, it can also be exploited by other software modules. During the experiments, the robot also localized itself on the world model and a sensible next step is to also use this approach for motion planning of the manipulators.

In the current approach, navigation benefits of the use of a central world model. A final recommendation is to also do the reverse: actively use the robot capabilities such as navigation and manipulation to improve the world model.

Appendix A

Domestic service robot AMIGO

This chapter discusses AMIGO, the robot that is used for most experiments throughout this thesis. AMIGO (see Figure A.1) is short for Autonomous Mate for Intelligent Operations. AMIGO has been developed in 2010 and has since been used in, *e.g.*, the RoboEarth project (Waibel et al, 2011) and the RoboCup@Home competition (Wisspeintner et al, 2009), with a second place at RoboCup 2014 in João Pessoa. The core specifications of AMIGO can be found in Table A.1 and a more elaborate description of the base, upper body, manipulators and electronics can be found in Sections A.1 to A.4.

(a) In the university library. (Photo courtesy of FM-Fotografie.)

(b) In the RoboCup@Home competition. (Photo courtesy of Bart van Overbeeke Fotografie.)

Figure A.1: The AMIGO robot.

A.1 Base platform

The design of the base platform of AMIGO (see Figure A.2) was originally intended for a Middle-Size League soccer robot (Alaerds, 2010). The use of this existing design sped up the development of the robot, although some modifications were required to make it suitable as platform for a service robot (Clephas, 2011). The main modification is that it is upscaled to 60 cm $\times$ 60 cm. This way, the footprint is increased for stability and space for peripheral equipment but the robot can still easily pass through doorways and narrow hallways.

Table A.1: Core specifications of AMIGO

	AMIGO
Name	Autonomous Mate for IntelliGent Operations
Base	Fully holonomic omni-wheel platform based on a soccer robot
Torso	Vertical ball-screw spindle (stroke: 0.32 m)
Manipulators	2 7-DoF Philips Experimental Robotic Arms
Neck	Pan-tilt unit using two Dynamixel RX-28 servo actuators
Head	Kinect for XBox 360
External devices	Wireless emergency button
Dimensions	Diameter: 0.75 m, height: ±1.5 m
Weight	±70 kg
Additional sensors	Hokuyo UTM-30LX laser range finder on base and torso
Microphone	RØDE Videomic
Batteries	4× Makita 24 V, 3.3 Ah
Computers	3× AOpen Mini PC with Core-i7 processor and 8 Gb RAM

Although having three omniwheels ensures a statically determined platform and is sufficient for propulsion, it is decided to use four instead to maximize the stability in arbitrary direction for these dimensions. The diameter of the omniwheels has been increased to 15 cm, which is sufficient to drive over small thresholds. The motors and amplifiers that are used to drive the wheels are 150 W, 24 V DC motors with 1 : 43 gearbox and Elmo violin current amplifiers. With these motors, AMIGO is able to reach a top speed of 1.80 m/s = 6.5 km/h. Corresponding to a brisk human walking velocity, this has proven to be sufficient in practice.

The main material for the base is sheet metal. Contrary to using machined parts, sheet metal is very suited since it is light and does not require expensive milling. The main construction elements are an octagonal central box and four legs attached to it. The large space between the legs can be used to place much of the heavy peripheral equipment such as batteries, hence lowering the center of gravity. This further enhances the stability of the robot. Both the central box as well as the four legs are closed boxes, leading to a design that is both light and extremely stiff (Alaerds, 2010). To ease assembly of the base, folded metal sheet components are used to reduce the number of connections. The wheels of the robot are connected rigidly to the four legs, while a base plate underneath the four legs provides mounting possibilities for the peripheral equipment.

A.2 Upper body

To increase the reach of the manipulators (Section A.3), the upper body of AMIGO contains a ball-screw spindle mechanism (Clephas, 2011). The main advantages of this mechanism are its simplicity and that is consists of only standard components. This mechanism furthermore provides inherent safety: in case of a motor failure, the robot will not collapse but only cause the robot to slowly founder. Excessive motor forces, on the other hand, are attenuated by the slip coupling which is present in the lifting mechanism.

In Figure A.4a the robot can be seen in its upper and in Figure A.4b in its lower position. The stroke of the upper body is 32 cm. With a maximum velocity of 0.14 m/s, it can traverse its stroke in 2.3 s. As can be seen in Figure A.4, the robot is able to pick up objects from the floor in its lower position. The height in its upper position is 1.60 m, hence giving AMIGO a

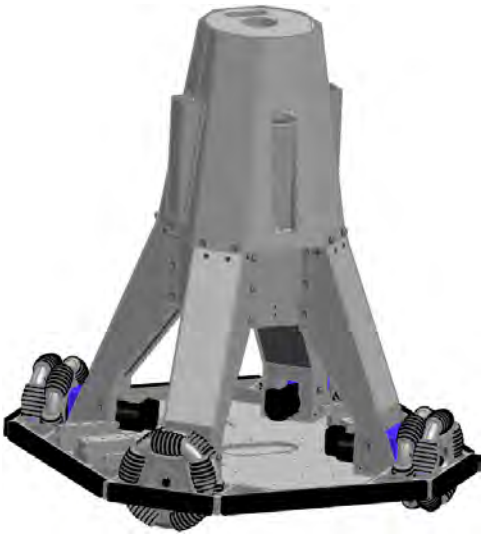


Figure A.2: The base platform with four legs connected to a central box.

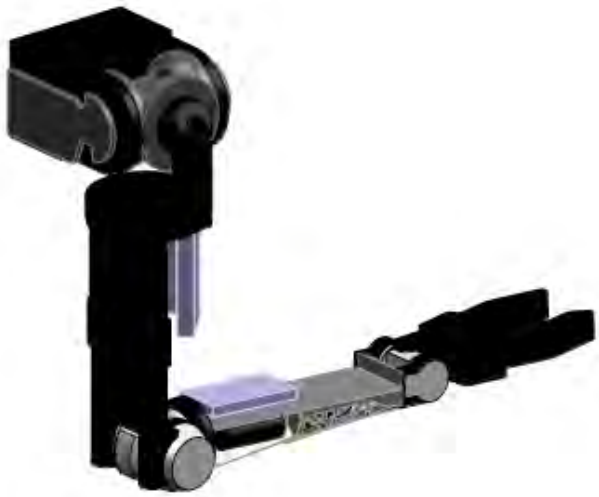


Figure A.3: A Philips Experimental Robot Arm.

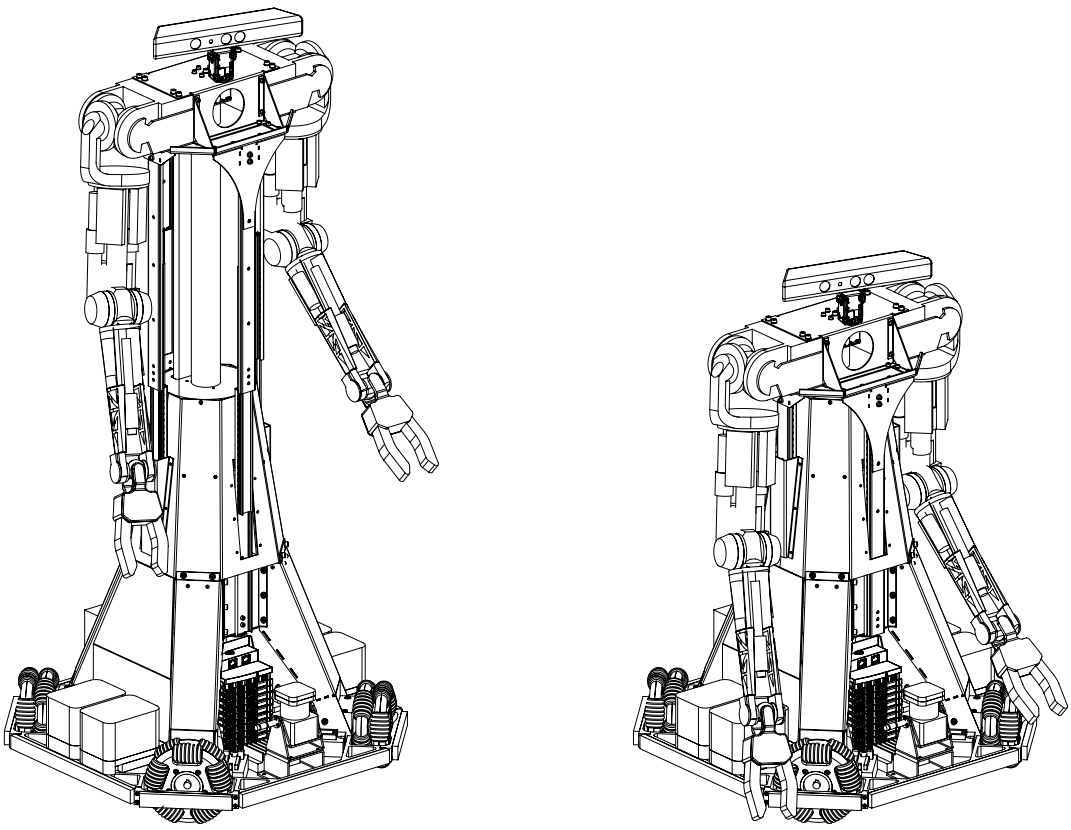
friendly appearance. Next to the arms and the lifting mechanism, a pan-tilt unit is mounted in the upper body, on which a robot head or a camera can be placed. This pan-tilt unit consists of two RX-28 Dynamixel servos including DC motor, gearbox and control circuitry. These can communicate either via RS485 or USB interfaces.

A.3 Manipulators

AMIGO is equipped with two Philips Experimental Robotic Arms (PERA, see Figure A.3). This anthropomorphic manipulator was designed as a research tool with example applications typically including household tasks (Rijs et al, 2010), which is exactly the intended purpose of this robot. To give it its anthropomorphic properties, it has human-like dimensions and seven DoFs, similar to human arms. The shoulder, elbow and wrist joints contain differential drives, giving it a further human-like appearance. Finally, it is equipped with a two finger one DoF gripper to grasp objects.

The PERA is moved by DC motors and each joint is additionally equipped with a Hall magnetic sensor and a torque sensor. The motors and gearboxes are selected such that fully stretched, each arm can lift 1.5 kg, which is sufficient for its purpose. Furthermore, the maximum velocities that can be reached are also based upon human motions. The encoder is used for the regular position control while the Hall sensor is used to determine the absolute position of the joint. In practice, only the hall sensors in the shoulder are used due to unreliable measurements of the remaining sensors. The torque sensors allow the torque control and therefore soft robotics control (see, *e.g.*, Albu-Schäffer et al (2007)) and they can be used for collision and failure detection. Nevertheless, the torque sensors are currently not used.

To prevent the possibility of harming people or damaging objects, the shoulder joint, which is the most powerful joint, is equipped with slip couplings to limit the maximal force the arm can exercise.



(a) AMIGO in its upper position. **(b)** AMIGO in its lower position.

Figure A.4: Torso motion of AMIGO.

A.4 Electronics

In the previous sections, the mechanical design of the robot has been discussed. In this section, the peripheral equipment that is used is discussed.

A.4.1 Input/Output

To control the motors and read the sensors of the base and upper body, AMIGO is equipped with a Beckhoff EtherCAT stack. EtherCAT is used to because of the high sampling rates and low jitter, enabling position and velocity control on the PC's. The EtherCAT stack contains slaves with encoder inputs and analog outputs used to connect the differential encoders and amplifiers of the base and the lifting mechanism to the PC. Furthermore, slaves with analog inputs supply information about the battery voltage and current output and slaves for the RS485 protocol are available to connect the pan-tilt unit to the PC. Finally, a number of digital inputs and outputs is present to enable the amplifiers, notify the software whether or not an emergency button is pressed and for status LEDs. In the manipulators, the original USB I/O boards have been replaced by custom EtherCAT boards, introduced in Section 2.5.1.

A.4.2 Sensors, computers and power

The head of the robot is a Kinect camera mounted on the pan-tilt unit introduced in Section A.2. The Kinect is used for 3D navigation and people- and object detection and recognition. For

human-robot interaction, a directional microphone, a speaker and an LED bar are present. The base platform houses a second LRF for localization and navigation.

The robot houses three Core i7 mini PC's with 8 GB RAM, located in the base platform. The communication between the computers is handled by a Gigabit Ethernet router with a wireless link (both 2.4 GHz and 5.0 GHz) to a base station.

To provide the necessary power, AMIGO has four Makita 24 V, 3.3 Ah power tool batteries that can be substituted within seconds. This has proven to be very convenient for a research platform since one never has to wait for the robot to charge.

For safety reasons, AMIGO is equipped with an emergency circuit. Both a wired and a wireless emergency stop are mounted on AMIGO, cutting power to all amplifiers of base, lifting mechanism and arms as well as applying the brake to the spindle motor in case of an error or emergency situation.

Appendix B

Sharing open hardware through ROP, the Robotic Open Platform

The robot open source software community, in particular ROS, drastically boosted robotics research. However, a centralized place to exchange open hardware designs does not exist.
This chapter[1] therefore introduces the Robotic Open Platform (ROP). A place to share and discuss open hardware designs. Among others it currently contains detailed descriptions of Willow Garage's TurtleBot, the NimbRo-OP created by the University of Bonn and the AMIGO robot of Tech United Eindhoven.
Eventually, ROP will contain a collection of affordable hardware components, allowing researchers to focus on cutting-edge research on a particular component instead of having to design the entire robot from scratch.
As an example of how the Robotic Open Platform is able to facilitate this knowledge transfer, we introduce TURTLE-5k: A redesign of an existing soccer robot by a consortium of our university and companies in the wider Eindhoven area. Cooperating with industrial partners resulted in a significant cost reduction.

B.1 Introduction

In recent years, increasing research efforts have been devoted to domestic service robots. Creating robotic agents able to autonomously assist humans in a highly unstructured environment is a huge step in research and engineering, too big for any university, research institute or company alone. Collaboration and knowledge exchange is therefore of utmost importance. Within robotics, the open source community has given software developments a tremendous boost, with the Robot Operating System (ROS) as its best-known exponent. Open hardware, on the other hand, is less common.

Nevertheless, some examples of open robotic hardware designs already exist. Willow Garage's PR2 Wyrobek et al (2008) has been introduced as an Open Platform: Users are encouraged to modify the system and open interfaces enable the use of different grippers, forearms, whole arms or sensors. A second open hardware design by Willow Garage is TurtleBot. Open hardware designs for humanoid robots are, e.g, NimbRo-OP Schwarz et al (2012), iCup Metta et al (2008)

[1] This chapter has been published as an article in RoboCup 2013: Robot Soccer World Cup XVII: Lunenburg et al (2013b)

and DARwIn-OP Ha et al (2013). Finally, Thymio II Magnenat et al (2012) is a low-cost open hardware educational robot.

Besides having individual robots released under an open hardware licensce, there are also community efforts to promote the exchange of hardware designs. Robotsource[2] is an online initiative on which some of the designs mentioned above are released, but it is aimed at marketing and productizing robotic solutions rather than sharing open hardware. RoboCup on the other hand, being the largest annual open source robotics event, provides a huge pool of knowledge on open hardware design.

However, unlike ROS provides for software, there is no centralized online platform for hardware designs yet. Furthermore, there is no standard format to publish open robot hardware, causing the information that is published to be insufficient to actually build the robots.

Therefore this work presents the Robotic Open Platform (ROP): An online place to share open hardware designs for robots. Inspired by ROS, embedded wiki pages provide specifications and explanations but also link to a repository containing CAD files, electric drawings and information on open source hardware licenses. The ROP 'Questions and Answers' page provides a place people can go to ask others in the ROP community for help. The Robotic Open Platform is a place where people can share, discuss and follow up on hardware designs, which eventually should be as easy as sharing ROS packages is now.

In the following section, we will discuss some examples of hardware knowledge exchange within and between RoboCup leagues. In Section B.3 we discuss ROP in more detail, and elaborate on how ROP currently facilitates hardware knowledge exchange. In the final section, we introduce TURTLE-5k, an example of ROP facilitating knowledge transfer between industry and the RoboCup community.

B.2 Sharing hardware designs within RoboCup

RoboCup, being the biggest annual open source robotics event, provides a huge pool of knowledge on hardware design. In this section we will highlight some examples of knowledge exchange within the RoboCup community.

B.2.1 RoboCup Middle-Size League

The effects and benefits of sharing hardware designs is clearly visible within the RoboCup Middle-Size League (MSL). According to the rulebook, a team is almost completely free in designing their soccer playing robots. Only restrictions on size, weight and color exist. However, the developments over the past years have led to solutions commonly used throughout the league. In this section vision units, the motion base, shooting mechanisms and ball handling mechanisms will be discussed respectively[3].

A robot needs to be able to perceive its environment and as much of the soccer field as possible. While in the past sometimes tilting camera systems were used, nowadays most teams use an omnidirectional vision module Nadarajah and Sundaraj (2013): A single camera pointed at a hyperbolic mirror on top of the robot (Figure B.1).

MSL soccer robots need to be both fast, agile and preferably holonomic. In the past, steering wheels and differential drives were common. However, since these drive concepts are non-holonomic or semi-holonomic, most MSL teams currently use omni wheels D'Andrea et al (2001) (Figure B.2).

[2] `http://www.robotsource.org/`

[3] The figures depicted in this section are taken from the team description papers or websites of the corresponding teams.

(a) BITAC **(b)** Minho **(c)** Water **(d)** Tech United Eindhoven

Figure B.1: Omnivision systems of several MSL teams.

A third example is found in the adoption of the shooting mechanism. Many systems can be used to shoot a ball, such as spring driven devices Dirkx (2004), pneumatic actuators Searock et al (2004) or electromechanical systems using a solenoid Meessen et al (2010). Of these solutions, the solenoid has proven to be superior since the shooting power can be fully controlled Meessen et al (2010) and is therefore currently used by most MSL teams.

A final example concerns ballhandling mechanisms. Solutions using only rubber bands to control the ball have been used, but more recently most teams use a system consisting of two movable levers in front of the robot with rotating wheels attached De Best et al (2011), enabling the robots to move in any direction while keeping possession of the ball.

B.2.2 RoboCup@Home League

In the previous section we discussed knowledge exchange within a RoboCup league. However, knowledge can also be shared between leagues. As an example we will discuss the design of the AMIGO robot, participating in RoboCup@Home.

The @Home league is aimed at the development of autonomous service robots for domestic applications. Compared to MSL, this league is relatively new. Therefore less solutions used throughout the league exist.

Various types of base-platforms are used, ranging from differential drives Seib et al (2013); Ziegler et al (2013) and steering wheels Dwiputra et al (2013); Stückler et al (2011) to fully holonomic platforms with omni wheels or Mecanum wheels Okada et al (2012). The concepts for the upper bodies and manipulators are also very different. Some robots have a human-like concept with a movable torso and two anthropomorphic manipulators Okada et al (2012); Stückler et al (2011), while others only have a single (industrial) manipulator without a torso

(a) Water **(b)** NuBot **(c)** Carpe Noctem **(d)** Tech United Eindhoven

Figure B.2: Omni wheels of several MSL teams.

(a) Carpe Noctem **(b)** MRL **(c)** Tech United Eindhoven

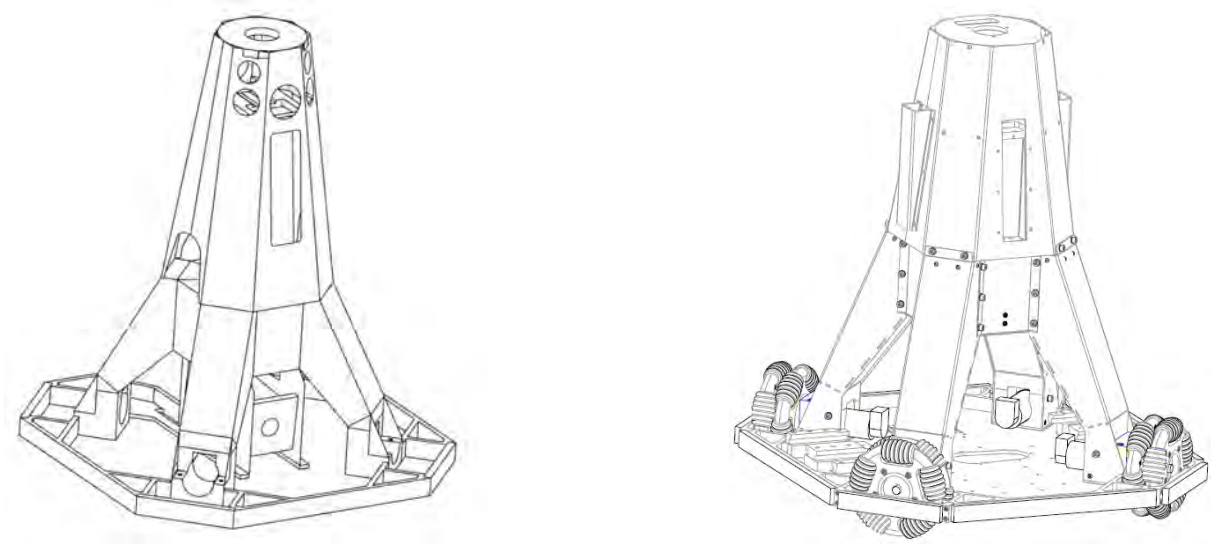
Figure B.3: Ballhandling systems of several MSL teams

joint Dwiputra et al (2013); Seib et al (2013); Ziegler et al (2013).

While designing the AMIGO robot for the RoboCup@Home league, team Tech United Eindhoven used hardware knowledge obtained in MSL as a blueprint. Although the purpose of a soccer robot and a domestic service robot are different, they do share some requirements:

- Both robots need powerful actuators and amplifiers. Either to accelerate quickly and reach a high velocity (soccer robots), or to drive over small thresholds such as doorsteps (domestic service robot).
- Various sensors and actuators need to be connected to on-board computers.
- A (semi-) holonomic base platform is convenient, either for object manipulation (service robot) or for agility (soccer robot).
- Both platforms need an internal power source.

To design a base, an omni wheel platform which was originally developed to be an MSL soccer robot is used (Figure B.4). This design has been scaled up for stability reasons and to increase space available for peripheral equipment.



(a) Concept of the TURTLE robot. **(b)** The AMIGO robot.

Figure B.4: The mechanical design of the base platform for a soccer robot has been used as a basis for a domestic service robot. From these figures, the similarity is evident.

The diameter of the omni wheels has been increased to 15 cm, which is sufficient to drive over small thresholds such as doorsteps. The wheels are actuated by the same motors and

amplifiers as used for the MSL robot. Only the gearboxes are different, since these require a larger reduction due to the larger mass, larger wheels and smaller velocities of a domestic service robot.

The PCs on AMIGO are connected to sensors and actuators using Beckhoff EtherCAT stacks. This is also directly copied from the TURTLE soccer robots. A final similarity between the two robots concerns the power supply. In both cases, power tool batteries are placed on the base platform.

Since other than handling a ball, no object manipulation is required for an MSL robot, this is where the comparison ends. The torso of AMIGO is a custom design with a ball screw actuator for vertical movement and two commercially available anthropomorphic robotic arms attached. For vision purposes a commonly used Kinect camera is mounted on a pan-tilt unit made out of Dynamixel servomotors.

B.3 Robotic Open Platform

In the sections above we have shown that sharing knowledge on hardware design decreases the development efforts required to build a robot. However, such knowledge transfer is not always as efficient as it should be, mainly due to the lack of a centralized platform to do so.

Inspired by the impact ROS had on creating and sharing open source robot software, we launched the Robotic Open Platform (ROP). This website with embedded wiki pages is designed to provide a central place for the exchange of hardware designs. Comparable to open source software, companies and research institutions can release their robot hardware design on ROP under an open hardware license. Users can post a description of their robot or component on the ROP wiki, including figures and videos. By providing a link to a repository containing detailed lists of specifications, CAD drawings, electric schemes, part-lists and any additional documentation required to build the robot, ROP grants access to the hardware-equivalent of source code. These documents are accompanied by the license under which the designs are released.

Furthermore, a Questions and Answers section has been included to allow contributors to ask for assistance.

With the information provided, all users should be able to build their own robot. Similar to open source software, possible modifications and improvements to designs also have to be released. Having ROP as a central platform to share hardware designs can enhance the collaboration within RoboCup, but also within the entire international robotics community. The first robots that have been published on ROP were discussed in previous sections: The TURTLE soccer robot and service robot AMIGO, followed by the TurtleBot, Thymio II and NimbRo-OP.

Although having a central place to share hardware designs is a nice step in what we envision to be the right direction, there is still a limitation: The hardware designs discussed in Section B.2 are fully integrated designs, making it difficult to combine different components in a new robot. Therefore modular design principles must be applied and standard electro-mechanical interfaces must be defined. When this is achieved, a collection of highly configurable and affordable hardware components will become available. Eventually, using hardware modules will be as

easy as using ROS packages is now.

B.4 TURTLE-5k, a low-cost MSL robot

In the previous section, ROP has been described as a platform to share open hardware designs. TURTLE-5k is a first example of ROP leading to better hardware: Together with industry an existing MSL soccer robot has been redesigned to significantly reduce costs.

B.4.1 Small series versus mass production

Within RoboCup, we are building a maximum of six robots of a specific design, which is much less than the large series of mass-produced service robots we all envision. This quantitative difference is of huge influence on choices made while designing a robot, since the optimal ratio between development costs and production costs depends on the size of the series. For example, using folded sheet metal instead of milling parts out of an expensive aluminum monolith can reduce costs but requires more development effort. When only one or two robots of a specific design are built, it does not pay off to put additional effort in making the design itself as low-cost as possible.

B.4.2 Cooperation with industry

While a lot of expertise on smart robot design exists at RoboCup teams, knowledge on how to turn these ideas into something that can be produced at low-cost is lacking. This knowledge, however, is abundantly present in industry. With this in mind, a consortium of three companies[4] and Eindhoven University of Technology was founded. By building on specifications and CAD drawings of the original TURTLE soccer robot, previously published on ROP, the consortium came up with a prototype design of a much more affordable MSL robot. Since this robot design will be on ROP as well, it can be improved by people all over the world. Assuming a minimum of ten teams chooses to buy these low-cost soccer robots, we think it is possible to reach a cost-per-robot of five thousand euros, which is one-fifth of the original cost. Hence, the name of this robot will be TURTLE-5k.

At the time of writing this chapter, a prototype is being built, including a ball handling and shooting mechanism. It will be tested, evaluated and redesigned if necessary, such that it can be presented at RoboCup 2013.

After completion, this newly designed robot should provide a base-platform, recapitulating hardware knowledge gained over the past fifteen years of MSL's existence as much as possible. It should be easily adjustable and extensible such that the league maintains innovative on the hardware level. Teams focussing on a specific part of the robot, *e.g.*, vision, should be able to add their own equipment or replace existing units with their own. Therefore, TURTLE-5k will not be a standard-platform for the MSL. Instead, it gives teams the opportunity to buy an affordable base, already able to play soccer but requiring extensions in order to stay competitive. It allows teams to spend a larger part of their budget on cutting-edge research by reducing start-up costs and time.

We consider the TURTLE-5k project to be a first example of ROP facilitating knowledge transfer between different areas of robotics research. In industry much of the knowledge on cost-focussed design we are looking for, is readily available. Through ROP, we are able to reach it.

[4] ACE, project management and mechanical design. VEDS, electrical design. Frencken Group, manufacturing and assembly

B.5 Conclusions

In robotics research, well-designed hardware is just as important as smart software. Open hardware releases will pave the way towards such well-designed hardware. Sharing CAD files, parts lists and electrical schemes reduces start-up costs for anybody willing to start research in the robotics field. It boosts progress towards better hardware design. Although initiatives such as RoboCup promote knowledge exchange, a central platform to share knowledge on hardware designs was still lacking.

Through the Robotic Open Platform, such knowledge on mechanical and electrical design can be centralized, which stimulates modular design principles, standardization of interfaces and facilitates knowledge exchange within the worldwide robotics community. Having a collection of highly configurable and affordable hardware components allows researchers to spend their effort and budget on cutting-edge research instead of having to 'reinvent the wheel' once again.

The TURTLE-5k project is a perfect example of ROP providing knowledge exchange between different communities. Based on specifications of an existing soccer robot, published on the ROP wiki, industry is able to improve the design. As a result, knowledge from industry is blended with knowledge accumulated in the RoboCup community.

Appendix C

Using topological maps for manipulation with domestic service robots

Whole-body control is a promising control method for domestic service robots. However, since whole-body control is a potential field-based method, it only operates at a local scope and it might therefore be unable to find a global solution in a complex environment with local minima. Therefore, to integrate a whole-body controller with a task executer, a planner is required that provides connectivity information at a global scope while constraining only the DoFs that are relevant for the (sub-task) at hand.

The contribution of this chapter is to apply a topological planner for manipulation, where the topological graph is an abstract representation of possible robot configurations that describes possible sequences of motions.

All nodes in the topological graph are grounded to specific end-effector attractors. Each attractor can constrain up to six DoFs and possesses a number of properties, such as the target pose, target offset, stiffness and goal tolerances.

The feasibility of the motion is subsequently validated by forward integration of the closed-loop whole-body controller.

C.1 Introduction

Over the past years, increasing research attention has been devoted to domestic service robots. These robots usually consist of a mobile base with one or two robotic arms. These arms have six or seven Degrees-of-Freedom (DoFs) and are commonly mounted on a torso with additional DoFs. A fundamental competence of these robots is the ability to safely manipulate objects in a domestic environment.

Many of the control methods for this purpose are based on the operational space formulation (Khatib, 1987). This formulation eventually resulted in whole-body control, of which implementations can be found in, *e.g.*, Nagasaka et al (2010); Dietrich et al (2012) and Gienger et al (2005); Sentis and Khatib (2005), where the latter references concern biped humanoid robots instead of wheeled mobile robots. There are various motives that justify the use of whole-body controllers in a domestic environment:

- Robustness against dynamic environments due to the presence of humans. This requires reactive motions, which is a key feature of whole-body controllers. A task of a whole-body controller is typically defined as an impedance in *Cartesian space*, *i.e.*, as a virtual spring between the current end-effector pose and the desired end-effector pose (also called 'attractor'). By defining this as a compliant spring rather than a stiff trajectory, tasks involving interaction with the environment are more robust against inaccuracies in the environment model and this compliance enhances safety in case of undesired interaction with the environment.

- the kinematic redundancy of domestic service robots. Both the path and goal constraints of a task are typically defined in Cartesian space, and due to this redundancy the specification of joint positions and trajectories is no longer a practical means of defining a task (Brock and Khatib, 2002). This is especially true if a robot has multiple manipulators. In whole-body control, the redundancy is utilized for online optimization of secondary motion objectives such as keeping a desired posture and avoiding joint limits.

- Since the end-effectors are controlled in Cartesian space, a point-to-point motion typically results in an end-effector trajectory that is straight in Cartesian space, which looks much more human-like than a trajectory that is straight in joint space.

However, since whole-body control is a potential field based method, it only operates at a local scope. As a result, it might be unable to find a global solution in a complex environment with local minima. This illustrates the need for a global planning method on top of the whole-body controller, as is recognized by, *e.g.*, Yang and Brock (2006); Toussaint et al (2007); Behnisch et al (2010, 2011); Dietrich et al (2012).

An important issue for this global planning method is the question which DoFs to constrain. Already in Nakanishi et al (2007), it is argued that it is desirable to control only the minimum number of DoFs of a robot such that the remaining DoFs can be utilized to optimize secondary motion objectives in the null space of the primary task. The minimum number of DoFs required to complete a task may even vary during a motion. An example is grasping an object from a table. Initially, the end-effector moves to the correct height while being close to the robot to avoid the table (the height and distance to the robot body are constrained by the task), then moves to a pre-grasp pose and a grasp pose (up to six DoFs need to be constrained). After closing the gripper, the object is lifted (the height and possibly, to keep an object such as a coffee cup level, roll and pitch are constrained) and the arm is retracted (the end-effector needs to be close to the robot and roll and pitch are possibly constrained).

To integrate a whole-body controller in a task executer, a planner is required that provides connectivity information in a global scope. Furthermore, it constrains only the DoFs that are relevant for the (sub-) task at hand to maximize the number of redundant DoFs that can subsequently be used to optimize secondary motion objectives.

C.2 Related work and contribution

In literature, examples can be found of motion planners that i) plan in joint space, ii) plan an end-effector path in Cartesian space, typically represented as a sequence of attractors or iii) call motion primitives from a task executer.

A recent example of a joint space planner can be found in Şucan and Kavraki (2012). However, this planner does not take path constraints into account. To include path constraints when planning in the joint space, a constraint manifold is computed and a path is searched on that manifold (Şucan and Chitta, 2012). Although good results can be obtained with these planners,

there is no robustness against dynamic environments: the increasing number of degrees of freedom increases computational complexity for motion planning in joint space. For a 7-DoF arm, planning times are in the order of magnitude of 0.5 s and these planning times will increase with increasing number of DoFs. This limits the ability to replan (Brock and Khatib, 2002). Furthermore, planning in joint space may result in motions that are far from human-like, which is undesired for anthropomorphic domestic robots.

For whole-body controllers, the planners do not result in an explicit trajectory but rather a sequence of attractors in Cartesian space. These attractors can constrain up to six DoFs of the end-effector, *i.e.*, both the position and the orientation.

In Brock and Khatib (2002), the concept of elastic strips is introduced as a framework to robustly execute a global motion. However, it is not discussed how this global motion is computed or how to recover from an invalidation of the global motion. To include this global connectivity information, elastic roadmaps are introduced in Yang and Brock (2006, 2010). Here, a sampling-based planning approach is used to create a roadmap in workspace. The elasticity implies that individual attractors are allowed to move in response to obstacles while constantly updating the connections between them. In these references, there is a distinction between end-effector placement and position-constrained end-effector motion and it is recognized that, depending on the task at hand, the end-effector orientation may also be constrained throughout the motion. Nevertheless, integration of this method with a task planner and executer and the selection of the DoFs to constrain are not discussed.

Hybrid planning methods, *i.e.*, methods that plan in Cartesian space but verify the motions in joint space, can be found in, *e.g.*, Ojdanic and Graser (2007); Behnisch et al (2010) and Behnisch et al (2011). In Ojdanic and Graser (2007), a 3D cell decomposition is searched for a global path. For every visited cell, an inverse kinematics (IK) solution is computed. If no collision free IK solution exists for a certain cell, this will get infinite costs. The orientation of the end-effector during the motion is neither released nor explicitly specified but the end-effector is rotated gradually from its start towards its goal orientation.

In Behnisch et al (2010, 2011), an Expansive Space Tree (Hsu et al, 2002) is used as a global search component. The connectivity between attractors is determined by integrating the closed-loop system forward in time using the Jacobian pseudo-inverse. Here, tasks are defined by either the position (3D) or by both the position and the attitude (5D) of the end-effector.

In Toussaint et al (2007), a motion is represented by a sequence of task space attractors. The exact position of these attractors is optimized with respect to smoothness of the motion, collision distance measures and joint limit avoidance. In Gienger et al (2008), it is also recognized that a task can be described by individual positions or orientations. Nevertheless, these references do not formally describe how to choose these DoFs. Furthermore, the constrained DoFs are also constant over the computed trajectories.

A completely different approach, not relying on whole-body planning and control, is applying parameterized motion primitives which are called from a task executer. An example can be found in, *e.g.*, Stückler et al (2012). In this reference, it is also argued that a direct reach toward the object is often collision-free, making a time-consuming joint-space planner superfluous. This corresponds to our experience in the RoboCup@Home competition (Wisspeintner et al, 2009): in AMIGO's (Lunenburg et al, 2014) grasping and placing motions, the yaw of the gripper is usually left unconstrained because many of the objects that are manipulated fit in the robot's grippers in any orientation. By moving the end-effector up close to the body before a grasping or placing motion and by retracting to a height above the surface where the object is located, collision-free grasping and placing motions can be performed, even *without* an explicit collision-avoidance algorithm. In the approach AMIGO used during previous RoboCup competitions, these motions are called separately by a pre-programmed state machine. Appropriate values for the DoFs that are irrelevant for the task at hand are selected ad-hoc rather than optimized in

realtime. Although this approach works in practice, it limits the re-usability for other tasks.

In the references mentioned in this section, either i) the number of DoFs that need to be constrained is constant, *e.g.*, three, five or six DoFs are constrained over the entire motion ii) the integration with task planner or executer is not discussed or iii) motions are pre-programmed in a state-machine.

To obtain a planner that can be used for multiple tasks while constraining the minimum number of DoFs during the motion, one can find inspiration in navigation of mobile robots. In navigation systems for mobile robots, an additional hierarchical layer on top of the metric planners can be added in the form of a topological map, which is an abstract representation that describes relationships among features of the environment, without any absolute reference system (Zavlangas and Tzafestas, 2002b). A topological map is usually represented in graph form, where the nodes represent sectors and the edges represent gateways. These sectors and gateways commonly have a semantic label associated with them, such as a room, a hallway or a door.

The contribution of this chapter is to apply a topological planner for manipulation. The nodes of the topological graph represent the pose of the attractors and also the stiffness and the tolerances. Unlike the aforementioned planners, different nodes can constraint different DoFs. Additionally, the nodes have a semantic label, *e.g.*, pre-grasp, grasp and retract, associated with them. This approach:

- minimizes the number of constrained DoFs during motions, resulting in motions that are closer to optimality and more robust to external disturbances since the redundant DoFs can be used to optimize secondary motion objectives. Furthermore, the motions are more human-like compared to sampling-based joint space planners because no random sampling is involved.

- formalizes the various motion stages, offloading the calls to the various motion primitives from the task executer and thereby enhancing re-usability.

This chapter is organized as follows: in Section C.3 whole-body control will be briefly introduced. The topological motion planner will be introduced in Section C.4, including verification of the computed attractors. Since the implementation of the planner and controller have not been completed, only a preliminary simulation result is shown in Section C.5. This chapter ends with the discussion in Section C.6.

C.3 Whole-body control

The whole-body controller that is used in this chapter is similar to the approach in Dietrich et al (2012), see Figure C.1. In this scheme, a Cartesian impedance is present for task execution. As mentioned in the introduction, it basically spans a virtual spring and damper in up to six DoFs between the desired and the actual end-effector position, resulting in a wrench $\mathbf{W}$ on that specific link. Using the transpose of the relevant Jacobian matrices $J^T(\mathbf{q})$, joint torques τ are computed using

$$\tau = J^T(\mathbf{q})\,\mathbf{W}, \tag{C.1}$$

where $\mathbf{q}$ are the n joint positions and τ the corresponding desired joint torques. Since AMIGO currently does not have torque controlled joints, an admittance coupling is used to computed desired joint velocities and joint positions:

$$M_{\mathbf{a}}\ddot{\mathbf{q}} + D_{\mathbf{a}}\dot{\mathbf{q}} = \tau. \tag{C.2}$$

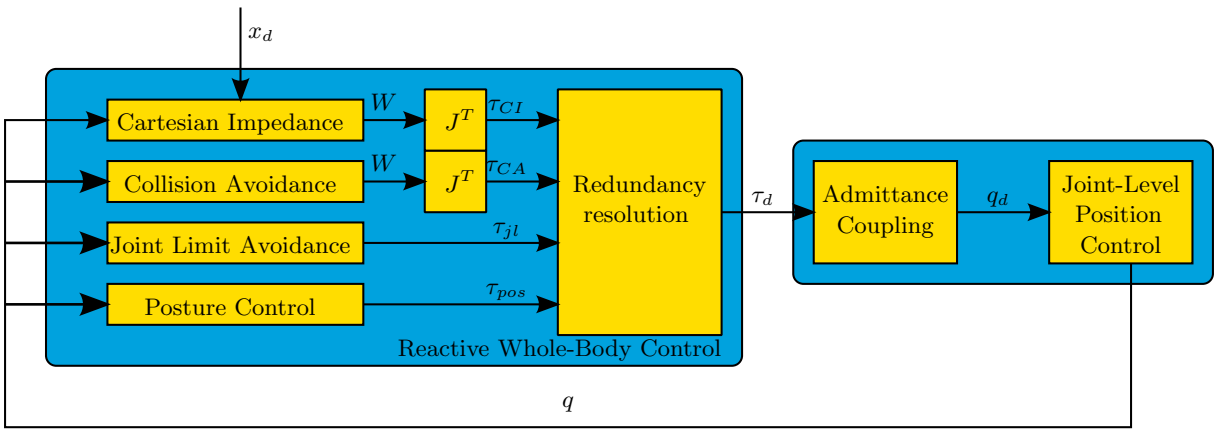


Figure C.1: The whole-body controller.

Here, $M_{\mathbf{a}}$ and $D_{\mathbf{a}}$ are a virtual diagonal mass and damping matrix. The parameters are chosen such that the closed-loop position controlled joints are able to track the desired references q_d.

The (self-) collision avoidance algorithm works in a similar fashion: it puts a repulsive force on a robot link if the distance between this two links or between a link and the environment decreases below a certain safety threshold.

As secondary objectives, joint limit avoidance and posture control are present. The joint limit avoidance algorithm puts a torque on the joints if q_i gets too close to its limits:

$$\begin{cases} \tau_{i,jl} = k_{i,jl} \frac{q_{\text{min,thresh},i} - q_i}{(q_{\text{max}} - q_{\text{min}})^2} & \text{for } q_i < q_{\text{min,thresh},i} \\ \tau_{i,jl} = k_{i,jl} \frac{q_{\text{max,thresh},i} - q_i}{(q_{\text{max}} - q_{\text{min}})^2} & \text{for } q_i > q_{\text{max,thresh},i} \end{cases} \tag{C.3}$$

where q_i is the joint position of joint i, $k_{i,jl}$ is a gain, q_{min} and q_{max} are the lower and upper joint limits and $q_{\text{min,thresh},i}$ and $q_{\text{max,thresh},i}$ are the thresholds below/above which a repulsive torque is applied.

The posture controller works similarly to keep the robot configuration as close as possible to its desired configuration:

$$\tau_{i,pos} = k_{i,pos} \frac{q_{0,i} - q_i}{(q_{\text{max}} - q_{\text{min}})^2} \tag{C.4}$$

where $k_{i,pos}$ is a gain and $q_{0,i}$ is the desired joint position of joint i.

In principle, more motion objectives could be integrated that, *e.g.*, minimize torques or keep a certain object in view of the camera. As some objectives such as collision avoidance are more important than others such as posture control, a hierarchy is constructed. Hereto, the torques of less important objectives are projected into the nullspace of more important objectives. This is called the redundancy resolution.

C.4 Topological arm navigation

The attractors that are fed to the Cartesian impedance in Section C.3 are computed by a topogical planner, which is discussed in this section. First, the topological graph is introduced, followed by the specific properties that must be defined for each node, *i.e.*, grounding the node. Thereafter, it is discussed how the computed motion is validated in joint space.

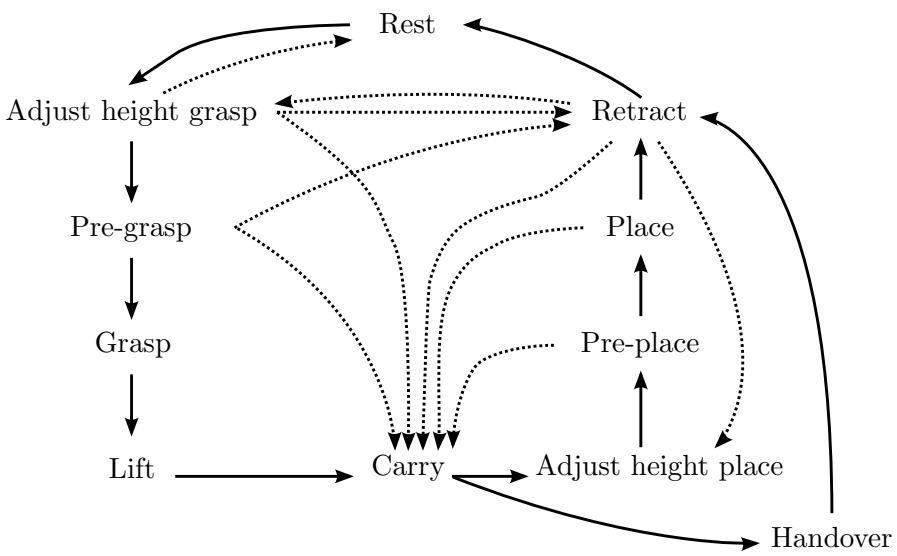


Figure C.2: Graph of the various manipulation (intermediate) goals used in RoboCup 2014. The solid arrows denote motions that were performed in practice, while the dotted arrows represent feasible motions that are not common in practice.

C.4.1 Topological graph

According to Zavlangas and Tzafestas (2002b), a topological map is an abstract representation that describes relationships among features of the environment, without any absolute reference system. A topological map is typically represented in graph form. In our implementation of a topological planner for manipulation, the topological graph is an abstract representation of possible robot configurations that describes possible sequences of motions.

In Figure C.3, an overview of the various manipulator (intermediate) goals that the AMIGO robot Lunenburg et al (2014) performed at the 2014 RoboCup@Home competition Wisspeintner et al (2009) is presented. During the competition, these goals were issues separately by a task executive, which implies that all transitions were pre-programmed. By appropriately selecting these intermediate points collision-free trajectories were obtained, even *without* an explicit collision-avoidance algorithm.

In this chapter, we propose to formalize these goals in a topological graph. A task executer can then, *e.g.*, ask the motion planner for a 'grasp' motion, resulting in three subsequent attractors (adjusting the height, pre-grasping and grasping). At this point, the graph only contains nodes for pick-and-place actions but more nodes can be added as the robot is supposed to perform more complicated manipulation tasks. A rule of thumb when adding nodes is that a collision-free trajectory is often possible if the robot always retracts its end-effector to close to its body before proceeding to the next task. The definition of the attractors, *i.e.*, grounding the nodes, is discussed in the next section.

C.4.2 Attractors

All nodes in the graph in Figure C.3 have to be grounded to specific end-effector attractors. Each attractor can constrain up to six DoFs and possesses a number of properties:

- Target pose: As mentioned in De Schutter et al (2007), *at least* two object frames and two feature frames are required to define a constraint. The target pose represents the first object frame and the first feature frame. Up to three position and three orientation DoFs can be specified.

Table C.1: Arm motions during RoboCup 2014

Motion	Constrained DoFs	Comments
Reset	-	Arm is at rest
Adjust height grasp	$x,\ y,\ z$	
Pre-grasp	$x,\ y,\ z,\ r_x,\ r_y,\ r_z$	Less with axissymmetric objects
Grasp	$x,\ y,\ z,\ r_x,\ r_y,\ r_z$	Less with axissymmetric objects
Lift	z	r_x and r_y are constrained in case an object needs to be carried level
Carry	$x,\ y$	r_x and r_y are constrained in case an object needs to be carried level
Adjust height place	$x,\ y$	r_x and r_y are constrained in case an object needs to be carried level
Pre-place	$x,\ y,\ z$	r_x and r_y are constrained in case an object needs to be carried level
Place	$x,\ y,\ z$	r_x and r_y are constrained in case an object needs to be carried level
Retract	$x,\ y$	r_x and r_y are constrained in case an object needs to be carried level
Handover	$x,\ y,\ z$	r_x and r_y are constrained in case an object needs to be carried level

- Target offset: The target offset represents the second object frame and feature frame.
- Stiffness: Defines the stiffness of the virtual spring between the current end-effector pose and the desired end-effector pose. If the stiffness is zero, this DoF is unconstrained.
- Goal tolerance: The position tolerance is currently defined by a box, sphere or cylinder. If the end-effector position is within the position tolerance, the position constraint is met. If a certain DoF is free, the tolerance of this DoF should be chosen such that it is always met. The orientation tolerance is defined by roll, pitch and yaw tolerance. If the end-effector orientation is within these tolerances, the orientation constraint is met. Again, if a certain DoF is free, the tolerance of this DoF should be chosen such that it is always met. In case of orientations, 2π is always sufficient. In the future, it should be possible to define these tolerances in a more flexible way that is not limited to these simple shapes. Furthermore, force tolerances would also be a valuable addition.

In Table C.1 the DoFs that are constrained in case of the motions in Figure C.2 are summarized. Furthermore, Figure C.3 shows three examples where specific parameters for the atrractors and tolerances are defined for the AMIGO robot. Here, the attractors in Figure C.3a and Figure C.3c are defined with respect to the robot and the attractor in Figure C.3b is inside the object to grasp.

C.4.3 Forward integration

By updating the graph based on object to grasp and searching the graph, a sequence of attractors with accompanying stiffness is obtained. Next, it must be validated whether these attractors result in a feasible motion. This is done by forward integration of the closed-loop whole-body controller, similar to Behnisch et al (2010).

The forward integration simply means the sequence of attractors is fed to a simulator of the system in Figure C.1. As soon as the end-effector meets the specified goal tolerances, the next

(a) Adjust height grasp: since mainly the height of the end-effector is important, the constraint volume is a cylinder with a large diameter and small height.

(b) Grasp: the attractor and constraint volume are inside the wire frame.

(c) Retracting: since the exact height is irrelevant, the constraint volume is a narrow but tall cylinder.

Figure C.3: Visualization of three attractors. The red spheres indicate the exact location of the attractor. The green volume indicates the constraint, *i.e.*, once the end-effector is inside the volume, it can proceed to the next attractor.

attractor is fed to the system. If the goal tolerance of one of the attractors is not met after $ni_{\max}$ iterations, this motion is invalidated. Since the admittance coupling is designed such that the closed-loop position controlled joints are able to track the desired trajectories, $q = q_d$ to speed up the simulation.

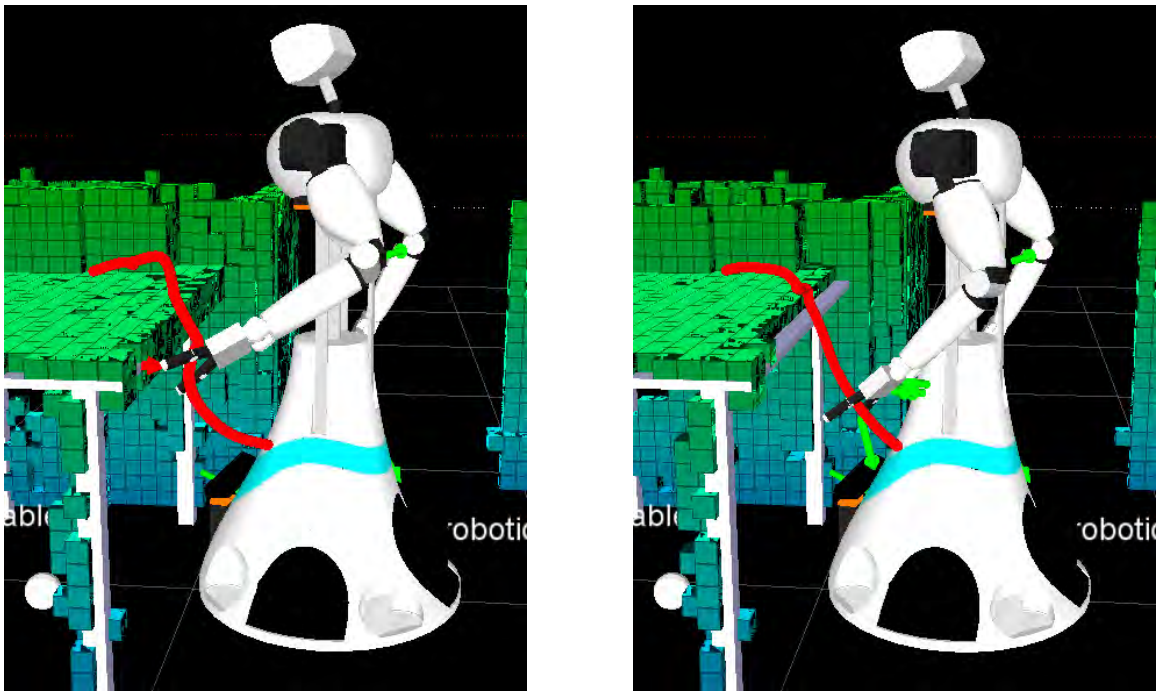
If the entire sequence of attractors is feasible it is fed to the real robot in exactly the same way. In case a part of the trajectory is not feasible, a possible solution is to repair this interval using a geometric motion planning method such as a probabilistic roadmap (Amato et al, 1998; Laumond and Nissoux, 2000; Kurniawati and Hsu, 2004; Yeh et al, 2012) or a rapidly exploring random tree (Hsu et al, 1997; Kuffner and LaValle, 2000). This is left as future work.

C.5 Simulation results

Although the implementation has not yet been finalized, a simulation has been performed to illustrate the approach. AMIGO is the domestic service robot of the Eindhoven University of Technology. Its base platform has four omniwheels and is hence fully holonomic. It is equipped with two 7-DoF Philips Experimental Robotic Arms. These have the dimensions of the arms of a large person and are placed on an upper body that can move up and down with a telescopic spine. In its lower position, AMIGO can grasp objects from the floor while in its upper position it has the size of a child and can therefore operate most features in a domestic environment. The kinematic structure of AMIGO is redundant: the two arms and torso have a $2 \times 7 + 1 = 15$ DoFs.

In this simulation, AMIGO is performing a typical grasping move. As a comparison, we have also performed this simulation with waypoints constructed in a more common probabilistic roadmap. Here, a visibility-based sampling method (Laumond and Nissoux, 2000) has been used in 6-DoF. The resulting end-effector trajectory can be seen Figure C.4a, while the trajectory of

the topological planner is displayed in Figure C.4b. A striking difference in these trajectories is that the maximum height of the random trajectory equals 1.13 m, while the actual target is more than 20 cm lower at 0.90 m. As a result, the motion using the PRM planner looks very unnatural and not human-like. Furthermore, the PRM planner does not enforce the end-effector to approach the object to grasp from the correct direction. Finally, the topological planner will always result in a similar trajectory, while a PRM-based planner will result in a different trajectory every time a new roadmap is constructed.



(a) Using a visibility-based probabilistic roadmap. **(b)** Using the topological planner.

Figure C.4: End-effector trajectories when using a visibility-based probabilistic roadmap and using the topological planner.

C.6 Conclusions & future work

This chapter demonstrates the use of a topological graph for motion planning for manipulation. Although the implementation of the entire pipeline has not yet been finished, preliminary simulations show promising results. A first prerequisite to see the benefits of this planner in practice is to have a decent implementation of the whole-body controller. This allows to make a proper comparison between a geometric planner and a topological planner to assess whether the decreased number of constrained DoFs indeed leads to improved performance. The graph in Figure C.2 and Table C.1 only contain nodes for simple pick-and-place operations. To use this approach in practice, more nodes should be added to the graph so that more tasks are covered.

As discussed in Section C.4.2, the nodes are grounded using pose constraints. Extending the definition of the attractors, *i.e.*, the grounding of the nodes, offers numerous possibilities to improve performance. One of the possible extensions is to include force constraints. Consider, for example, the task of placing an object. The 'place' move, can then be implemented with the attractor position at the table surface and a force constraint in vertical direction. As a result, the robot will move the object down until it 'feels' that it has reached the table.

Bibliography

Akgun B, Stilman M (2011) Sampling heuristics for optimal motion planning in high dimensions. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2640–2645

Alaerds R (2010) Mechanical design of the next generation Tech United TURTLE. Master's thesis, Eindhoven University of Technology

Alami R, Siméon T, Madhava Krishna K (2002) On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol 3, pp 2395–2400

Albu-Schäffer A, Haddadin S, Ott C, Stemmer A, Wimböck T, Hirzinger G (2007) The dlr lightweight robot: Design and control concepts for robots in human environments. Industrial Robot 34(5):376–385

Amato N, Bayazit O, Dale L, Jones C, Vallejo D (1998) OBPRM: an obstacle-based PRM for 3D workspaces. Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective pp 155–168

Anderson GF, Hussey PS (2000) Population aging: A comparison among industrialized countries. Health Affairs 19: 3:191–203

Arslan O, Tsiotras P (2013) Use of relaxation methods in sampling-based algorithms for optimal motion planning. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 2421–2428

Asfour T, Regenstein K, Azad P, Schröder J, Bierbaum A, Vahrenkamp N, Dillmann R (2006) ARMAR-III: An integrated humanoid platform for sensory-motor control. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp 169–175

Beetz M, Mösenlechner L, Tenorth M (2010) Cram – a cognitive robot abstract machine for everyday manipulation in human environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 1012–1017

Behnisch M, Haschke R, Gienger M (2010) Task space motion planning using reactive control. In: 23rd IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, Taipei, pp 5934–5940

Behnisch M, Haschke R, Ritter H, Gienger M (2011) Deformable trees - exploiting local obstacle avoidance. In: 2011 IEEE-RAS International Conference on Humanoid Robots, pp 658–663

van den Berg J, Ferguson D, Kuffner J (2006) Anytime path planning and replanning in dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 2366–2371

van den Berg J, Abbeel P, Goldberg K (2011) LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. International Journal of Robotics Research 30(7):895–913

Bischoff R, Huggenberger U, Prassler E (2011) Kuka youbot - a mobile manipulator for research and education. In: IEEE International Conference on Robotics and Automation, pp 1–4

Blumenthal S, Bruyninckx H, Nowak W, Prassler E (2013) A scene graph based shared 3d world model for robotic applications. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 453–460

Brock O, Khatib O (1999) High-speed navigation using the global dynamic window approach. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 1, pp 341–346

Brock O, Khatib O (2002) Elastic strips: A framework for motion generation in human environments. International Journal of Robotics Research 21(12):1031–1052

Bugmann G, Copleston SN (2011) What can a personal robot do for you? Towards Autonomous Robotic Systems 6856:360–371

Burns B, Brock O (2007) Sampling-based motion planning with sensing uncertainty. In: IEEE International Conference on Robotics and Automation, pp 3313–3318

Canny JF (1988) Complexity of robot motion planning. MIT press

Chiesa A (2013) Dynamics aware 3d occupancy grid map with semantic information. In: International Conference on Advanced Robotics, pp 1–6

Choset H (2001) Coverage for robotics - A survey of recent results. Ann Mathematics and Artificial Intelligence 31(1):113–126

Choset H (2005) Principles of robot motion: theory, algorithms, and implementation. MIT Press

Chung W, Kim G, Kim M, Lee C (2004) Integrated navigation system for indoor service robots in large-scale environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 5, pp 5099–5104

Clephas T (2011) Design and control of a service robot. Master's thesis, Eindhoven University of Technology, Department of Mechanical Engineering, Control Systems Technology Group

Coenen S, Lunenburg J, van de Molengraft M, Steinbuch M (2014) A representation method based on the probability of collision for safe robot navigation in domestic environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems

Şucan I, Chitta S (2012) Motion planning with constraints using configuration space approximations. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 1904–1910

Şucan I, Kavraki L (2012) A sampling-based tree planner for systems with complex dynamics. IEEE Transactions on Robotics 28(1):116–131

Dadkhah N, Mettler B (2012) Survey of motion planning literature in the presence of uncertainty: considerations for UAV guidance. Journal of Intelligent & Robotic Systems pp 1–14

D'Andrea R, Kalmár-Nagy T, Ganguly P, Babish M (2001) The Cornell RoboCup team. Lecture Notes in Computer Science 2019

De Best J, Van de Molengraft R, Steinbuch M (2011) A novel ball handling mechanism for the RoboCup Middle-Size League. Mechatronics 21(2):469 – 478, special Issue on Advances in intelligent robot design for RoboCup MSL

De Schutter J, De Laet T, Rutgeerts J, Decré W, Smits R, Aertbeliën E, Claes K, Bruyninckx H (2007) Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. International Journal of Robotics Research 26(5):433–455

Di Marco D, Tenorth M, Hussermann K, Zweigle O, Levi P (2013) Roboearth action recipe execution. In: Advances in Intelligent Systems and Computing, vol 193, Springer Berlin Heidelberg, pp 457–466

Dietrich A, Wimböck T, Albu-Schäffer A, Hirzinger G (2012) Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. IEEE Robotics & Automation Magazine 19(2):20–33

Dirkx F (2004) Philips CFT RoboCup team description. In: RoboCup 2004: Robot Soccer World Cup VIII, Springer

Dolgov D, Thrun S, Montemerlo M, Diebel J (2008) Practical search techniques in path planning for autonomous driving. Ann Arbor 1001

Donald B, Xavier P, Canny J, Reif J (1993) Kinodynamic motion planning. Journal of the Association for Computing Machinery 40(5):1048–1066

van Duin C, Stoeldraijer L (2013) Bevolkingsprognose 2012-2060: Langer leven, langer werken. Bevolkingstrends 2013

Dwiputra R, Füller M, Hegger F, Hochgeschwender N, Paulus J, Schneider S, Bierbrauer A, Shpieva E, Ivanovska I, Deshpande N, Gaier A, Hagg A, Sanches Loza J, Ozhigov A, Ploeger P, Kraetzschmar G (2013) The b-it-bots RoboCup@Home team description paper

Elfring J, van den Dries S, van de Molengraft M, Steinbuch M (2013) Semantic world modeling using probabilistic multiple hypothesis anchoring. Robotics and Autonomous Systems 61(2):95–105

Elfring J, Van de Molengraft R, Steinbuch M (2014a) Semi-task-dependent and uncertainty-driven world model maintenance. Autonomous Robots 38(1):1–15

Elfring J, Van De Molengraft R, Steinbuch M (2014b) Learning intentions for improved human motion prediction. Robotics and Autonomous Systems 62(4):591–602

Ferguson D, Likhachev M, Stentz A (2005) A guide to heuristic-based path planning. In: Workshop on Planning under Uncertainty for Autonomous Systems at International Conference on Automated Planning and Scheduling

Fiorini P, Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research 17(7):760–772

Fox D (2001) KLD-sampling: Adaptive particle filters and mobile robot localization. Advances in Neural Information Processing Systems 14(1):26–32

Fox D, Burgard W, Thrun S (1997) The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine 4(1):23–33

Fraichard T (1998) Trajectory planning in a dynamic workspace: a 'state-time space' approach. Advanced Robotics 13(1):75–94

Fuchs M, Borst C, Giordano P, Baumann A, Kraemer E, Langwald J, Gruber R, Seitz N, Plank G, Kunze K, Burger R, Schmidt F, Wimböck T, Hirzinger G (2009) Rollin' Justin - design considerations and realization of a mobile platform for a humanoid upper body. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 4131–4137

Gienger M, Janssen H, Goerick C (2005) Task-oriented whole body motion for humanoid robots. In: 2005 5th IEEE-RAS International Conference on Humanoid Robots, pp 238–244

Gienger M, Toussaint M, Goerick C (2008) Task maps in humanoid robot manipulation. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2758–2764

Giesbrecht J (2004) Global path planning for unmanned ground vehicles. Tech. Rep. DRDC'Suffield TM 2004-272, Defence R&D Canada - Suffield

Goerzen C, Kong Z, Mettler B (2010) A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent & Robotic Systems 57(1-4):65–100

Goodman JE, O'Rourke J (2004) Handbook of discrete and computational geometry. Chapman & Hall/CRC

Graf B, Hans M, Schraft R (2004) Care-O-bot II - development of a next generation robotic home assistant. Advanced Robotics 16(2):193–205

Graf B, Parlitz C, Hägele M (2009) Robotic home assistant Care-O-bot®3 product vision and innovation platform. In: International Conference on Human-Computer Interaction , pp 312–320

Guibas LJ, Hsu D, Kurniawati H, Rehman E (2009) Bounded uncertainty roadmaps for path planning. In: Algorithmic Foundation of Robotics VIII, Springer, pp 199–215

Ha I, Tamura Y, Asama H, Han J, Hong D (2011) Development of open humanoid platform darwin-op. In: Proceedings of the 2011 SICE Annual Conference, pp 2178–2181

Ha I, Tamura Y, Asama H (2013) Development of open platform humanoid robot DARwIn-OP. Advanced Robotics

Hermann A, Sun J, Xue Z, Ruehl S, Oberlaender J, Roennau A, Zoellner J, Dillmann R (2013) Hardware and software architecture of the bimanual mobile manipulation robot hollie and its actuated upper body. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp 286–292

Hornung A, Phillips M, Gil Jones E, Bennewitz M, Likhachev M, Chitta S (2012) Navigation in three-dimensional cluttered environments for mobile manipulation. In: IEEE International Conference on Robotics and Automation, pp 423–429

Hornung A, Wurm KM, Bennewitz M, Stachniss C, Burgard W (2013) OctoMap: an efficient probabilistic 3d mapping framework based on octrees. Autonomous Robots pp 1–18

Hsu D, Latombe JC, Motwani R (1997) Path planning in expansive configuration spaces. In: Proceedings of the 1997 IEEE International Conference on Robotics and Automation, vol 3, pp 2719–2726 vol.3

Hsu D, Kavraki L, Latombe J, Motwani R, Sorkin S, et al (1998) On finding narrow passages with probabilistic roadmap planners. In: Proceedings of the International Workshop on Algorithmic Foundations Robotics

Hsu D, Kindel R, Latombe JC, Rock S (2002) Randomized kinodynamic motion planning with moving obstacles. International Journal of Robotics Research 21(3):233–255

Hwang Y, Ahuja N (1992) Gross motion planning - a survey. ACM Computing Surveys 24(3):219–291

IFR (2012) Executive summary world robotics. Tech. rep., World Robotics

Iwata H, Sugano S (2009) Design of human symbiotic robot TWENDY-ONE. In: IEEE International Conference on Robotics and Automation, pp 580–586

Janssen R, van Meijl E, Di Marco D, Van De Molengraft R, Steinbuch M (2013) Integrating planning and execution for ros enabled service robots using hierarchical action representations. In: International Conference on Advanced Robotics, pp 1–7

Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S (2011) STOMP: Stochastic trajectory optimization for motion planning. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 4569–4574

Kant K, Zucker SW (1986) Toward efficient trajectory planning: The path-velocity decomposition. International Journal of Robotics Research 5(3):72–89

Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. International Journal of Robotics Research 30(7):846–894

Kavraki L, Svestka P, Latombe J, Overmars M (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4):566–580

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research 5(1):90

Khatib O (1987) A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE Journal of Robotics and Automation RA-3(1):43–53

Kitano H, Asada M, Kuniyoshi Y, Noda I, Osawa E (1997) RoboCup: The robot world cup initiative. In: Proceedings of the 1st International Conference on Autonomous Agents, pp 340–347

Koenig S, Likhachev M, Liu Y, Furcy D (2004) Incremental heuristic search in AI. AI Magazine 25(2):99

Kuffner J JJ, LaValle S (2000) Rrt-connect: An efficient approach to single-query path planning. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA), vol 2, pp 995–1001 vol.2

Kuipers B, Modayil J, Beeson P, MacMahon M, Savelli F (2004) Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 5, pp 4845–4851

Kurniawati H, Hsu D (2004) Workspace importance sampling for probabilistic roadmap planning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 1618–1623

Latombe J (1990) Robot motion planning. Springer

Laumond J, Sekhavat S, Lamiraux F (1998) Guidelines in nonholonomic motion planning for mobile robots. Robot motion planning and control pp 1–53

Laumond JP, Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. Advanced Robotics 14(6):477–493

LaValle S (2006) Planning algorithms. Cambridge University Press

LaValle S, Kuffner Jr J (2001) Randomized kinodynamic planning. International Journal of Robotics Research 20(5):378–400

LaValle SM, Branicky MS, Lindemann SR (2004) On the relationship between classical grid search and probabilistic roadmaps. Int J Robotics Research 23(7-8):673–692

Le Tien L, Schaffer A, Hirzinger G (2007) Mimo state feedback controller for a flexible joint robot with strong joint coupling. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, pp 3824–3830

Levihn M, Nishiwaki K, Kagami S, Stilman M (2014) Autonomous environment manipulation to assist humanoid locomotion. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 4633–4638

Likhachev M, Gordon G, Thrun S (2003) ARA*: Anytime A* with provable bounds on sub-optimality. Advances in Neural Information Processing Systems 16

Likhachev M, Ferguson D, Gordon G, Stentz A, Thrun S (2008) Anytime search in dynamic graphs. Artificial Intelligence 172(14):1613–1643

Lindemann SR, LaValle SM (2005) Current issues in sampling-based motion planning. Robotics Research pp 36–54

Lingemann K, Nüchter A, Hertzberg J, Surmann H (2005) About the control of high speed mobile indoor robots. In: Proceedings of the European Conference on Mobile Robots, pp 218–223

Lu DV, Hershberger D, Smart WD (2014) Layered costmaps for context-sensitive navigation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems

Lunenburg J, Coenen S, van den Dries S, Elfring J, Janssen R, Sandee J, van de Molengraft M (2013a) Tech United Eindhoven team description 2013

Lunenburg J, Soetens R, Schoenmakers F, Metsemakers P, van de Molengraft R, Steinbuch M (2013b) Sharing open hardware through rop, the robotic open platform. In: RoboCup 2013: Robot Soccer World Cup XVII, Lecture Notes on Artificial Intelligence, vol 8371, Springer Berlin Heidelberg, URL `http://www.roboticopenplatform.org`

Lunenburg J, Coenen S, Derksen T, van den Dries S, Elfring J, van de Molengraft M (2014) Tech United Eindhoven @Home team description 2014

Lunenburg J, van den Dries S, Bento Ferreira L, van de Molengraft M (2015a) Tech United Eindhoven @Home team description 2015

Lunenburg J, van de Molengraft R, Steinbuch M (2015b) A representation method based on the probability of collision for safe robot navigation. Autonomous Robots (Under review)

Lunenburg JJM, Coenen SAM, Naus G, van de Molengraft MJG, Steinbuch M (2015c) A recipe to select a motion planner for navigation tasks of mobile robots. IEEE Robotics & Automation Magazine (Under review)

Magnenat S, Riedo F, Bonani M, Mondada F (2012) A programming workshop using the robot 'Thymio II'. In: Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts, pp 24–29

Maravall D, De Lope J, Serradilla F (2000) Combination of model-based and reactive methods in autonomous navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 3, pp 2328–2333

Marder-Eppstein E, Berger E, Foote T, Gerkey B, Konolige K (2010) The office marathon: Robust navigation in an indoor office environment. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 300–307

Martínez-Barberá H, Herrero-Pérez D (2010) Autonomous navigation of an automated guided vehicle in industrial environments. Robotics and Computer-Integrated Manufacturing 26(4):296–311

Meessen K, Paulides J, Lomonova E (2010) A football kicking high speed actuator for a mobile robotic application. In: Annual Conference on IEEE Industrial Electronics Society, DOI 10.1109/IECON.2010.5675433

Metta G, Sandini G, Vernon D, Natale L, Nori F (2008) The icub humanoid robot: An open platform for research in embodied cognition. In: Workshop on Performance Metrics for Intelligent Systems, pp 50–56

Milford M, Wyeth G (2010) Persistent navigation and mapping using a biologically inspired slam system. International Journal of Robotics Research 29(9):1131–1153

Minguez J, Montano L (2000) Nearness diagram navigation (ND): a new real time collision avoidance approach. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol 3, pp 2094–2100

Minguez J, Montano L, Simeon T, Alami R (2001) Global nearness diagram navigation (GND). In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 1, pp 33–39

Minguez J, Montesano L, Montano L (2004) An architecture for sensor-based navigation in realistic dynamic and troublesome scenarios. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol 3, pp 2750–2756

Missiuro P, Roy N (2006) Adapting probabilistic roadmaps to handle uncertain maps. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 1261–1267

Moore D, Huang A, Walter M, Olson E, Fletcher L, Leonard J, Teller S (2009) Simultaneous local and global state estimation for robotic navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 3794–3799

Moravec HP (1988) Sensor fusion in certainty grids for mobile robots. AI Mag 9(2):61

Mori M (1970) Bukimi no tani (the uncanny valley). Energy 7(4):33–35

Nadarajah S, Sundaraj K (2013) Vision in robot soccer: a review. Artificial Intelligence Review pp 1–23, article in Press

Nagasaka K, Kawanami Y, Shimizu S, Kito T, Tsuboi T, Miyamoto A, Fukushima T, Shimomura H (2010) Whole-body cooperative force control for a two-armed and two-wheeled mobile robot using generalized inverse dynamics and idealized joint units. In: 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, pp 3377–3383

Nakanishi J, Mistry M, Peters J, Schaal S (2007) Towards compliant humanoids-an experimental assessment of suitable task space position/orientation controllers. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2520–2527

Ogren P, Leonard NE (2005) A convergent dynamic window approach to obstacle avoidance. IEEE Transactions on Robotics 21(2):188–195

Ojdanic D, Graser A (2007) A fast motion planning for a 7dof rehabilitation robot. In: 2007 IEEE 10th International Conference on Rehabilitation Robotics, pp 171–178

Okada H, Omori T, Watanabe N, Shimotomai T, Iwahashi N, Sugiura K, Nagai T, Nakamura T (2012) Team eR@sers 2012 in the @Home league team description paper

Ott C, Eiberger O, Friedl W, Bäuml B, Hillenbrand U, Borst C, Albu-Schäffer A, Brunner B, Hirschmüller H, Kielhöfer S, Konietschke R, Suppa M, Wimböck T, Zacharias F, Hirzinger G (2006) A humanoid two-arm system for dexterous manipulation. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp 276–283

Pandey A, Alami R (2013) Affordance graph: A framework to encode perspective taking and effort based affordances for day-to-day human-robot interaction. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2180–2187

Patil S, van den Berg J, Alterovitz R (2012) Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In: IEEE International Conference on Robotics and Automation, pp 3238–3244

Philippsen R, Siegwart R (2003) Smooth and efficient obstacle avoidance for a tour guide robot. In: IEEE International Conference on Robotics and Automation, vol 1, pp 446–451

Philippsen R, Jensen B, Siegwart R (2006) Toward online probabilistic path replanning in dynamic environments. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2876–2881

Philippsen R, Kolski S, Macek K, Jensen B (2008) Mobile robot planning in dynamic environments and on growable costmaps. In: Workshop on Planning with Cost Maps at the IEEE International Conference on Robotics and Automation

Phillips M, Likhachev M (2011) SIPP: safe interval path planning for dynamic environments. In: IEEE International Conference on Robotics and Automation, pp 5628–5635

Pivtoraiko M, Knepper RA, Kelly A (2009) Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics 26(3):308–333

Pronobis A, Jensfelt P (2012) Large-scale semantic mapping and reasoning with heterogeneous modalities. In: IEEE International Conference on Robotics and Automation, pp 3515–3522

Quinlan S, Khatib O (1993) Elastic bands: connecting path planning and control. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 2, pp 802–807

Qureshi A, Mumtaz S, Iqbal K, Ali B, Ayaz Y, Ahmed F, Muhammad M, Hasan O, Kim WY, Ra M (2013) Adaptive potential guided directional-rrt. In: IEEE International Conference on Robotics and Biomimetics, pp 1887–1892

Ratliff N, Zucker M, Bagnell J, Srinivasa S (2009) CHOMP: Gradient optimization techniques for efficient motion planning. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 489–494

Reynolds C, Wyatt J (2011) Open source, open standards, and health care information systems 13(1):–, URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3221346/`

Rijs R, Beekmans R, Izmit S, Bemelmans D (2010) Philips experimental robot arm, user instruction manual v1.1. Tech. Rep. APT536-09-9962, Philips Applied Technologies

Robustillo SA, Corsini V, Marcu M, Vasileva K, , Marchett E (2013) EU employment and social situation, quarterly review. Social Europe, Special Supplement on Demographic Trends

Rohrmüller F, Althoff M, Wollherr D, Buss M (2008) Probabilistic mapping of dynamic obstacles using markov chains for replanning in dynamic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2504–2510

Russell S, Norvig P (2010) Artificial intelligence: a modern approach. Prentice hall

Schwarz M, Schreiber M, Schueller S, Missura M, Behnke S (2012) NimbRo-OP humanoid teensize open platform. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots

Schwarz M, Pastrana J, Allgeuer P, Schreiber M, Schueller S, Missura M, Behnke S (2014) Humanoid teensize open platform NimbRo-OP. In: Lecture Notes in Computer Science, vol 8371, Springer Berlin Heidelberg, pp 568–575

Searock J, Browning B, Veloso M (2004) Segway CMBalance robot soccer player. Tech. rep., DTIC Document

Seib V, Kathe F, McStay D, Manthe S, Peters A, Jöbchen B, Memmesheimer R, Jakowlewa T, Vieweg C, Stümper S, Günther S, Müller S, Veith A, Kusenback M, Knauf M, Paulus D (2013) RoboCup - homer@UniKoblenz (Germany)

Sentis L, Khatib O (2005) Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. International Journal of Humanoid Robotics 02(04):505–518

Soucy M, Payeur P (2004) Robot path planning with multiresolution probabilistic representations: a comparative study. In: IEEE Canadian Conference on Electrical and Computer Engineering, vol 2, pp 1127–1130

Stachniss C, Burgard W (2002) An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In: IEEE/RSJ Int. Conf. on Intell. Robots and Systems, vol 1, pp 508–513

Stückler J, Behnke S (2009) Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp 506–513

Stückler J, Dröschel D, Gräve K, Holz D, Schreiber M, Behnke S (2011) NimbRo@Home 2011 team description

Stückler J, Holz D, Behnke S (2012) Robocup@home: Demonstrating everyday manipulation skills in robocup@home. IEEE Robotics & Automation Magazine 19(2):34–42

Tech United Eindhoven (2014) Website. `http://www.techunited.nl/en`

Tenorth M, Perzylo A, Lafrenz R, Beetz M (2013) Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. IEEE Transactions on Automation Science and Engineering 10(3):643–651

Thrun S, Bennewitz M, Burgard W, Cremers AB, Dellaert F, Fox D, Hahnel D, Rosenberg C, Roy N, Schulte J, et al (1999) MINERVA: A second-generation museum tour-guide robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol 3

Toussaint M, Gienger M, Goerick C (2007) Optimization of sequential attractor-based movement for compact behaviour generation. In: 2007 7th IEEE-RAS International Conference on Humanoid Robots, pp 122–129

Urmson C, Anhalt J, Bagnell D, Baker C, Bittner R, Clark M, Dolan J, Duggins D, Galatali T, Geyer C, et al (2008) Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics 25(8):425–466

Waibel M, Beetz M, Civera J, D'Andrea R, Elfring J, Gálvez-López D, Häussermann K, Janssen R, Montiel J, Perzylo A, Schießle B, Tenorth M, Zweigle O, Van De Molengraft R (2011) Roboearth - a world wide web for robots. IEEE Robotics & Automation Magazine 18(2):69–82

Weisshardt F, Reiser U, Parlitz C, Verl A (2010) Making high-tech service robot platforms available. In: Joint International Symposium on Robotics and German Conference on Robotics, vol 2, pp 1115–1120

Wisspeintner T, Van Der Zant T, Iocchi L, Schiffer S (2009) RoboCup@Home scientific competition and benchmarking for domestic service robots. Interaction Studies 10(3):392–426

Wurm K, Hennes D, Holz D, Rusu R, Stachniss C, Konolige K, Burgard W (2011) Hierarchies of octrees for efficient 3d mapping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 4249–4255

Wyrobek K, Berger E, Van Der Loos H, Salisbury J (2008) Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 2165–2170

Yang Y, Brock O (2006) Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In: Proceedings of Robotics: Science and Systems

Yang Y, Brock O (2010) Elastic roadmaps-motion generation for autonomous mobile manipulation. Autonomous Robots 28(1):113–130

Yeh HY, Thomas S, Amato NM (2012) UOPRM: A uniformly distributed obstacle-based PRM. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems

Zacharias F, Borst C, Hirzinger G (2007) Capturing robot workspace structure: representing robot capabilities. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 3229–3236

Zavlangas P, Tzafestas S (2002a) Integration of topological and metric maps for indoor mobile robot path planning and navigation. Methods and applications of artificial intelligence pp 746–746

Zavlangas P, Tzafestas S (2002b) Integration of topological and metric maps for indoor mobile robot path planning and navigation. In: Proceedings of the Second Hellenic Conference on AI: Methods and Applications of Artificial Intelligence, Springer-Verlag, pp 121–130

Zender H, Martínez Mozos O, Jensfelt P, Kruijff GJ, Burgard W (2008) Conceptual spatial representations for indoor mobile robots. Robotics and Autonomous Systems 56(6):493–502

Zhang H, Butzke J, Likhachev M (2012) Combining global and local planning with guarantees on completeness. In: IEEE International Conference on Robotics and Automation, pp 4500 – 4506

Ziegler L, Wittrowski J, Schöpfer M, Siepmann F, Wachsmuth S (2013) ToBI - Team of Bielefeld: The human-robot interaction system for RoboCup@Home

Nomenclature

Acronyms

AMCL	Adaptive Monte Carlo Localization
AMIGO	Autonomous Mat for IntelliGent Operations
CAD	Computer-Aided Design
CoG	Center of Gravity
DoF	Degree of Freedom
DWA	Dynamic Window Approach
GPGPU	General-Purpose computing on Graphics Processing Units
I/O	Input/output
IMU	Inertial Measurement Unit
LRF	Laser Range Finder
MSL	Middle-Size League
PSD	Power Spectral Density
RGB-D	Red Green Blue Depth
RMS	Root Mean Square
ROP	Robotic Open Platform
ROS	Robot Operating System
SERGIO	Second Edition Robot for Generic Indoor Operations
SPI	Serial Peripheral Interface
TURTLE	Tech United Eindhoven Limited Edition
TCP	Tool Center Point

Calligraphic symbols

$\mathcal{C}$	configuration space
$\mathcal{G}$	goal region
$\mathcal{N}$	normal distribution
$\mathcal{S}$	state space
$\mathcal{W}$	workspace

Greek symbols

$\alpha,\ \beta,\ \gamma,\ \delta$	scaling parameters
Δ_{dec}	rate of decay
$\boldsymbol{\mu}$	mean
σ	covariance
$\boldsymbol{\Sigma}$	covariance matrix
χ^2	chi-squared distribution

Roman symbols

a	acceleration
C	controller
c	costs
D	duration
$\mathbf{c}$	grid cell
d	distance
e	error
$\mathbf{e}$	error vector
J	objective function
k	time step
$L,\ l$	log-odds
n	index, degree of a polynomial function
$P,\ p$	plant, probability
q	joint
r	radius, resolution
R	range
s	cost decay rate
$T,\ t$	time
v	velocity
$\mathbf{v}$	velocity vector
x	forward direction
$\mathbf{x}$	robot position
y	sideward (left) direction

Sub- and superscripts

a	alignment
coll	collision
d	delay
free	free
g	goal
hc	high costs
infl	inflation
lc	low costs
max, m	maximum
min	minimum
obs	obstacle
occ	occupied
odom	odometry
p	path
real	real
rec	recovery
ref	reference
rob	robot
safe	safe
s	sensor
stop	stop
thresh	threshold
v, v	velocity

Dankwoord

"Wel of niet promoveren?", was de vraag die me na mijn afstuderen bezig hield. Van de keuze die ik toen heb gemaakt, heb ik nooit spijt gehad. Dit komt natuurlijk mede door iedereen die hierbij betrokken is geweest en ik wil hen hierbij graag bedanken.

Allereerst Maarten: bedankt voor de kans om deze promotieopdracht te doen. Ik kan me weinig plaatsen voorstellen waar je zoveel vrijheid krijgt om je eigen richting in je onderzoek te vinden en zoveel support krijgt om je onderzoek ook in de praktijk te brengen. René, bedankt voor de dagelijkse begeleiding. Dankzij onze discussies kwamen we steeds tot de essentie van de problemen waar we aan werkten en je feedback heeft het werk naar een veel hoger niveau getild. Ook wil ik de commissieleden bedanken voor het lezen van mijn proefschrift en hun deelname in de commissie.

In de afgelopen jaren heb ik natuurlijk het meest te maken gehad met mijn kamergenoten, met name Sjoerd, Jos, Rob, Pieter, Heico, Luis en de laatste maanden Yanick, Jesse en Javier. Niet alleen waren jullie goede sparringpartners over inhoudelijke vraagstukken maar jullie zorgden er ook voor dat er altijd een goede sfeer hing en er op zijn tijd ook andere onderwerpen ter sprake kwamen. Ook wil ik de overige mensen van DCT bedanken. Ondanks de 'afstand' tussen het roboticalab en gang -1 is het altijd plezierig werken geweest en werden de banden natuurlijk in stand gehouden tijdens de 24-uurs meetings en de Benelux meetings.

Tijdens mijn promotie heb ik veel studenten begeleid bij BEP, stage- en afstudeeropdrachten. Bedankt voor jullie inzet en de vele goede ideeën. Zonder jullie werk waren AMIGO en SERGIO nooit zover gekomen en tevens is de hand van een aantal van jullie projecten duidelijk terug te zien in de verschillende hoofdstukken van dit boekje.

Het onderzoek wat we in het lab deden werd meteen in de praktijk gebracht in de RoboCup @Home competitie. Daarom wil ik graag de teamleden van Tech United bedanken. De wekelijkse teamavonden en de workshops vormden een leuke afwisseling op het dagelijks werk en de hoogtepunten werden natuurlijk gevormd door de toernooien in Magdeburg, Istanbul, Mexico City, João Pessoa en thuis in Eindhoven. Ieder jaar hadden we weer een sterk team waardoor we altijd een stijgende lijn vast hebben kunnen houden, welke we hopelijk in Hefei kunnen bekronen!

Zonder technici hadden wij natuurlijk geen robots om mee te experimenteren en te RoboCuppen. Ruud en Gerard hielden AMIGO en SERGIO elektrisch en mechanisch in de lucht en bij het EPC bouwde men de robots en stond men altijd klaar in geval van problemen of als er groot onderhoud gepleegd moest worden. Bedankt hiervoor.

Ook wil ik graag mijn vrienden in en buiten Eindhoven bedanken, met name Cor, Dirk, John, Hein, Marco, Raymon en Tom. Dankzij de avondjes in de kroeg, de voetbalwedstrijden op het veld en vanaf de zijkant en andere activiteiten werd ik er steeds aan herinnerd dat er meer in het leven is dan robots en onderzoek.

Ook wil ik graag mijn familie bedanken. Pap en mam, het is nog steeds erg fijn thuiskomen in Loosbroek. Bedankt dat ik met alles bij jullie terecht kan en jullie altijd voor mij klaar staan. Anne, Luuk, Marloes en Ruben, bedankt voor jullie grote interesse in niet alleen mijn onderzoek maar ook alles daarbuiten.

En tenslotte natuurlijk Daniëlle. Je stond altijd klaar met je onvoorwaardelijke steun en onuitputtelijke geduld als alles moest wijken om dit boekje af te maken. Bij jou vind ik rust. Bedankt voor alles!

Curriculum vitae

Janno Lunenburg was born on September 27, 1985 in Veghel, the Netherlands. He completed his pre-university education in 2003 at Gymnasium Bernrode in Heeswijk Dinther. He obtained his B.Sc. degree in Mechanical Engineering at the Eindhoven University of Technology. After that, he started his M.Sc. program at the same department in the Control Systems Technology group. As part of this program, he spent three months at the University of California at Berkeley working on modeling and control of shape memory alloy actuators. He obtained his M.Sc. degree (with great appreciation) in 2010 with his thesis entitled "Inversion-Based MIMO Feedforward Design Beyond Rigid Body Systems". This research was performed at Philips Applied Technologies.

In November 2010 he started his Ph.D. research, also at the Eindhoven University of Technology. His research topics included motion planning for domestic service robots and the development of a modular service robot. The results of this research are presented in this thesis. During his Ph.D. research, he was team leader of the Tech United Eindhoven RoboCup@Home team. With the AMIGO robot, they participated in the annual RoboCup@Home competitions. Furthermore, he successfully completed the educational program of the DISC (Dutch Institute for Systems and Control) graduate school in 2013.