Navigation of the Smart Wheelchair Rolland

Christian Mandel

Kumulative Dissertation
zur Erlangung des Grades eines Doktors der Ingenieurwissenschaften
– Dr.-Ing. –

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)
Universität Bremen

08. November 2007

Datum des Promotionskolloquiums: 08. Januar 2008

Gutachter

Prof. Dr. Bernd Krieg-Brückner (Universität Bremen)
Prof. Dr. Kerstin Schill (Universität Bremen)

Abstract

This thesis develops methods and software modules that allow the paraplegic to navigate a computerized electrical wheelchair without hand use in everyday environments. The need for suitable user interfaces is tackled in two ways. On the one hand, the interpretation of coarse verbal route descriptions is realized by the mapping of spatial relations and referenced landmarks onto a Route Graph representation of the wheelchair's workspace. Despite the proof of practical feasibility within the bounds of formal syntax and closed dictionaries, the quest for everyday usability leads to the development of a second solution, i.e. a head-controlled joystick based on a 3-DOF orientation tracker (IMU). The lightweight device that is worn by a comfortable headband yields precise head posture angles. Using this information, a *quantitative* head-joystick places the operator in the control-loop, letting him/her control translational and rotational velocity by slight pitch and roll head movements. In a further application scenario the IMU is developed into a *qualitative* head-joystick, recognizing head gestures with which the user selects a desired target position in the vicinity. In addition, this thesis proposes the interpretation of qualitative instructional commands that describe a desired direction of movement. The implemented system is shown to reliably work with classified commands coming from a brain-computer interface.

This thesis proposes a geometric path planning algorithm as its second major contribution for both user interfaces. The obstacle avoiding approach employs cubic Bezier curves to search an optimal trajectory to the target pose. Unlike other popular approaches, necessary veer-off movements for navigating narrow passages are explicitly modeled, taking into account the platform's shape and kinematic properties.

Zusammenfassung

Die vorliegende Doktorarbeit entwickelt Methoden und Softwaremodule zur alltagstauglichen Unterstützung eines querschnittsgelähmten Patienten bei der Steuerung seines computerisierten Rollstuhls. Die dafür notwendige Benutzerschnittstelle wird zum einen mit der Interpretation von qualitativen verbalen Routenbeschreibungen bereitgestellt. Das im Rahmen dieser Arbeit entstandene Verfahren bildet die beteiligten räumlichen Relationen mitsamt den von ihnen referenzierten Landmarken auf die Routengraph-Darstellung der Umgebung ab. Trotz einer experimentellen Machbarkeitsstudie, die lediglich durch die formale Syntax und das begrenzte Wörterbuch potentieller Routenbeschreibungen eingeschränkt ist, führt die Frage nach der Alltagstauglichkeit zu der Entwicklung einer zweiten Benutzerschnittstelle in Form eines Kopf-Joysticks. Die leichtgewichtige Einheit, die mittels eines Orientierungssensors (IMU) die Lage des Patientenkopfes im Raum ermittelt, wird mittels eines bequemen Stirnbandes getragen. Der *quantitative* Kopf-Joystick platziert den Anwender in die Regelungsschleife des Systems und lässt ihn die translationale und rotationale Geschwindigekeit des Rollstuhls mittels leichten Nick- und Neigebewegungen des Kopfes beeinflussen. In einem weiteren Anwendungsszenario wird die IMU zu einem *qualitativen* Kopf-Joystick weiterentwickelt. Dieser interpretiert Kopfbewegungen des Benutzers als Zielpose auswählende Gesten. Darüber hinaus stellt diese Arbeit ein System zur Interpretation von qualitativen Instruktionen vor, die die gewünschte Fahrtrichtung spezifizieren. Es wird eine Implementierung vorgestellt, die entsprechende Kommandos von einer Gehirn-Computer Schnittstelle ausliest und robust verarbeitet.

Als zweiten Hauptbeitrag schlägt diese Doktorarbeit einen geometrischen Bahnplaner für die vorgestellten Benutzerschnittstellen vor. Der hindernisvermeidende Navigationsansatz setzt kubische Bezierkurven bei der Suche nach einer optimalen Trajektorie zur gewünschten Zielpose ein. Im Gegensatz zu aktuellen Verfahren werden notwendige Ausholmanöver beim Einbiegen in schmale Passagen explizit modelliert, sowie der genaue Umriss des Rollstuhls und seine kinematischen Eigenschaften berücksichtigt.

Contents

Chapter 1

General Introduction

Since 1994 scientists and students have worked under the direction of Prof. Dr. Bernd Krieg-Brückner at the University of Bremen on the development of a computerized wheelchair for the elderly and the disabled. The application of information technology in the fields of rehabilitation robotics and human-robot interaction is motivated by the goal to provide extra functionality to people who depend on a wheelchair in their everyday life. In this context, possible application scenarios are designed with respect to a reasonable degree of autonomy. They range from *Safety Layers* that prevent the autarkic user of an automated wheelchair from hitting surrounding obstacles up to *Navigation Modules* that autonomously pilot to user selected goals.

As a work that originated at the *Department of Mathematics and Computer Science*[1], the thesis presented here is concerned with the technical implementation of a system that is applicable in the domains sketched above. Despite the broad spectrum of involved techniques, it thereby focuses on the development of human-robot interfaces dedicated to the paralyzed, along with the generation and execution of spatially meaningful locomotion. In order to assess its contribution, this general introduction will first of all characterize state of the art projects in rehabilitation robotics that focus on smart wheelchairs.

1.1 Smart Wheelchairs: State of the Art in 2007

Automated wheelchairs that are equipped with sensors and a data processing unit constitute a special class of wheeled mobile robots. Not only the necessity of sophisticated algorithms that work on environmental data, but rather the shared spatial reference system of the operator and the smart wheelchair give rise to a variety of scientific questions. Consequently, a number of research groups have worked on the development of smart wheelchairs since the early 1980s. General literature overviews [32, 28, 9, 46] classify the realized projects mainly according to the type of applied hardware, implemented user interfaces, and the supported navigational modes. For each of these categories subsections 1.1.1-1.1.3 now highlight state of the art developments.

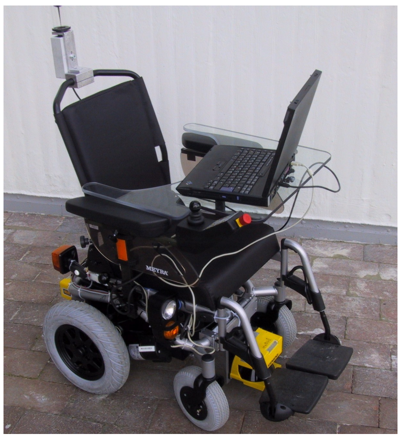
[1]This work has been funded by the German Research Foundation (DFG) in context of the Sonderforschungsbereich/Transregio 8 *Spatial Cognition*, project A1-[RoboMap]. Additional thanks go to the Deutsches Forschungszentrum für Künstliche Intelligenz for a doctoral scholarship.

(a) *MAid* (1999) - University of Ulm, Germany. [42]

(b) *NavChair* (1999) - University of Michigan, Ann Arbor, USA. [30]

(c) Smart wheelchair (2005) - University of Zaragoza, Spain. [34]

(d) *Vulcan* (2006) - University of Texas, Austin, USA. [25]


(e) *Rolland III* (2006) - University of Bremen, Germany.

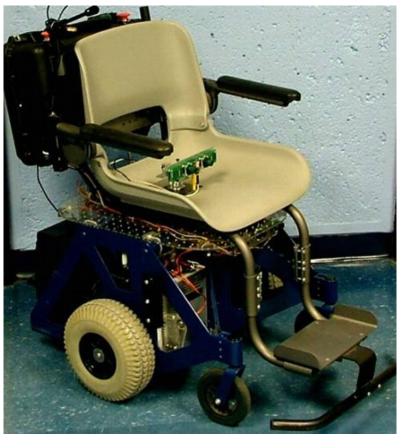

(f) *Wheelesley* (2007) - University of Massachusetts, Lowell, USA. [55]

Figure 1.1: Smart wheelchairs that feature fully autonomous navigation, simultaneous localization and mapping, and sophisticated interface techniques.

1.1.1 Sensorial Equipment

In order to perceive the surrounding area and to measure their own state, smart wheelchairs need sensors like any other robot. Actual developments restrict the choice of employed hardware in consideration of information gain and financial costs. The latter point, although not academically motivated, should not be underestimated as the development of the rehabilitation robot is intended to find a wide spread among the desired audience. In the following, the most prominent sensor types that are currently integrated into smart wheelchairs are presented. The overview thereby lists concrete projects with the focus on smart wheelchairs and the addressed application domains of the sensors.

One of the most eclectic sensor technologies is *computer vision*. Low-priced and small-sized cameras with still increasing resolution however need sophisticated real-time image processing algorithms. Prominent areas of application within the field of rehabilitation robotics are general landmark detection tasks [15], gesture recognition for user interface design [7], and simultaneous localization and mapping (SLAM) [3].

Since April 1995 the *global positioning system (GPS)* provides worldwide localization information. Small-size receiver units analyze the time of travel of electromagnetic signals from at least three out of currently 31 satellites. Due to the restricted application to outdoor scenarios and a best possible precision of about 1m (without expensive differential GPS transmitters), GPS has been rarely used in smart wheelchair projects for navigation or mapping purposes, e.g. in [26].

Dead reckoning for wheeled mobile robots is done by observing and integrating the system's own motion in space. For this purpose, *incremental encoders* measure the rotational speeds of the actuated wheels, and forward this proprioceptive information to the kinematic equations. Despite the accumulating error that is due to slipping wheels and varying wheel diameter, nearly all current and recent smart wheelchairs use this kind of odometry for motion estimation, cf. [48] for an overview.

An *inertial measurement unit (IMU)* is a proprioceptive device that senses inertial information, namely acceleration and rate of turn, by using its internal accelerometers and gyroscopes. Attached to an external carrier, an IMU estimates the spatial posture of the compound. The current generation of IMU devices can be built as small-size micro electromechanical systems (MEMS), thus easily attached to smart wheelchairs. Here they support odometry information for the estimation of the system's heading [18], or facilitate user interfaces that for instance interpret the user's head posture [6].

The most popular proximity sensor for general robotic applications is the *laser range finder (LRF)*. Measuring the time of flight of pulsed infrared laser beams that are emitted from the scanner and reflected from object surfaces, the LRF computes the metrical distance to an obstacle. Typical LRFs like the Sick LMS 200 (cf. [56] for a detailed survey) deflect the laser beam by a rotating mirror, yielding a 2-D fan-shaped field of scanning. Smart wheelchairs use information coming from LRFs to build up occupancy grids [10] that are employed by obstacle avoidance methods like the *Velocity Obstacle Concept* [42] (cf. Fig. 1.1(a) for the used platform), localization approaches, or by SLAM algorithms [45].

Like LRFs, *infrared range finders (IRFs)* emit light in the infrared band and receive reflections from nearby objects. In contrast to LRFs, simple IRFs do not compute the distance to the obstacle by measuring the time of flight, but by analyzing the intensity of the reflected beam. This makes IRFs susceptible for surface and shape properties of the scanned object. Nevertheless IRFs are often used because they are cheap and compact (cf. [49] for a broad analysis of IRFs in the context of feature detection).

Ultrasonic acoustic range finders, or *sonars* for short, have been frequently used on smart wheelchairs in the past. The low-cost sensor that emits an ultrasonic sound wave receives reflections of this signal from nearby objects, and computes the distance to the obstacle from the signal's time of flight. Despite their sensitivity to so-called cross talks, i.e. the event when one sensor receives an echo of a signal that was generated by a different sensor, sonars are still in use by actual intelligent wheelchair projects (cf. [48]). Beside their application in navigation and map-building tasks, sonars are also employed for precise localization in augmented environments (cf. [20]).

1.1.2 User Interfaces

People who depend on the use of smart wheelchairs for rehabilitation purposes or in everyday life suffer from a variety of disabilities that often limit their potential to access the device in a comfortable way. At this starting point the research of human-robot interaction (HRI) helps by providing sophisticated interface techniques. In [50], Thrun elaborates on the current generation of service robots that are characterized by increasing autonomy and an enormous reduction of financial costs. He points out past and current developments in HRI and poses open questions that have to be answered in order to make available robot techniques accessible to the rapidly aging society. The following survey on state of the art approaches in HRI focusses on methods that enable the paralyzed to control an automated wheelchair.

Brain Computer Interfaces

Well-established findings in neurophysiology have raised the question whether the electroencephalographic activity of the human's brain can be utilized in order to establish a brain computer interface (BCI). In a broad overview on actual techniques [54] Wolpaw et al. differentiate dependent BCIs from independent BCIs. Whereas the former rely on stimuli in the neural pathways in order to evoke a measurable brain activity, the latter do not depend in any way on signals within the nervous system. An example for the first class of BCIs is given by Rebsamen et al. [43]. The authors describe a user interface that is based on a graphical screen on which randomly flashing buttons are labeled by navigational goals. The operator selects a desired action by focusing on the desired target. When the aimed button flashes, a so-called P300 signal coming along with the surprise is emitted in the brain of the user and detected by a standard electro-encephalogram (EEG). A comparable system is presented by Friman et al. [12]. Their approach employs steady-state visual evoked potentials (SSVEPs), i.e. signals that are measurable by EEG on the visual cortex when the subject focuses his/her visual attention on light sources that flicker at frequencies above 4 Hz.

Vanacker et al. present an independent BCI that also applies a non-invasive EEG [51]. The contributed classifier inputs preprocessed EEG information and outputs the estimated probability distribution over the classes *left*, *forward*, and *right* at a rate of 2Hz. The experimental setup involves a shared-control wheelchair platform for which recognized forward-, left-, or right-commands result in a stepwise amplification of the translational or rotational velocity. After a sufficiently long period of time without any recognized command, the system initiates a safe stop manoeuvre.

Head Posture or Gaze Interpretation

The control of automated wheelchairs by analyzing the user's head posture or direction of gaze is appealing for patients that suffer from a reduced flexibility of the upper extremities. Gips presents in [14] *EagleEyes*, a system that places five electrodes around the right eye of the user. The measured electrooculographic potential between the retina and cornea corresponds to the posture of the eye relative to the head. EagleEyes has been used by Yanco to interface the wheelchair platform *Wheelesley* [55] (cf. Fig. 1.1(f)). By moving the cursor of a graphical user interface to four directional and one stop button, the user selects a behavioral navigation mode with his/her gaze. It should be mentioned that the overall concept requires the head of the patient to stay in a fixed position since control commands are caused by relative head-eye displacements.

In [6] Bureau et al. describe a system that integrates two MEMS-sensors, namely an accelerometer and a gyroscope, into a single head-mounted sensor unit. The small-size device is capable of measuring the acceleration of the patient's head in pitch direction as well as the speed of the head's yaw movement. Apart from interfacing selected components of an intelligent home environment, the authors present the control of an automated wheelchair by interpreting the patient's head movements. Therefore single pitch up/down or roll left/right movements are used to trigger an increasing/decreasing of the discretized translational or rotational velocity of the wheelchair. The major drawback of this approach is the fixed number of optional speeds in each dimension of control. This problem is due to accumulating errors in the integration of the input signals. These intrinsic errors avoid an application that evaluates the global deflection of the user's head like a common proportional joystick.

A straightforward application that exploits ultrasonic acoustic range finders for detecting a patients's head posture and subsequently derives velocity commands for an automated wheelchair is proposed by Jaffe [19]. The system that places two sonar sensors at the headrest of an automated wheelchair is able to detect the translation of the patient's head within a horizontal plane. A clinical evaluation with 17 participants revealed poor overall performance. For example 42%(41%) rated the precision of commanded forward motion as poor and turns as very poor.

As an example for wheelchair user interfaces that employ computer vision, the system of Canzler and Kraiss [7] detects the user's commands by analyzing facial features such as head posture, direction of gaze, and lip movement. The authors show the possibility to recognize at least four gesture dependent commands such as *go*, *stop*, *left*, and *right*.

Natural Language Communication

One of the classical approaches in HRI is to use verbal control since there has always been the ambitious goal to make machines act and communicate like humans. Although the straightforward concept of interfacing a computerized artifact by means of natural language has attracted a lot of research, there still exist essential challenges in functional and comprehensive implementations. The following overview on smart wheelchair projects that have been augmented by voice control stresses feasibility issues of this technique.

Simpson et al. present in [47] a voice control mechanism that has been implemented within the *NavChair Assistive Wheelchair Navigation System* [30] (cf. Fig. 1.1(b)). The applied continuous-speech voice recognition system *SpeechCommander* has to be trained prior to usage by a particular user before it can be employed to recognize a set of nine navigational commands, i.e. *Stop, Go Forward, Soft Left, Hard Left, Rotate left* and the symmetric counterparts. Due to the restriction to qualitative directional commands, the underlying navigation system has to provide sophisticated navigational assistance, e.g. avoidance of obstacles and compensation for misinterpreted commands.

A more ambitious approach has been started by Mueller et al. [38]. The authors propose a framework that, instead of interpreting short termed utterances, analyzes qualitative route descriptions such as "Follow this corridor, take the second corridor branching off on the right-hand side and stop at its end.". Implemented on the *Bremen Autonomous Wheelchair Rolland*, the system detects prominent features within a frequently updated obstacle grid and maps them onto five supported categories, i.e. *wall in front, corridor left, corridor right, door left* and *door right.* The recognized environmental situation is then mapped onto the given route description, that has been formalized in terms of *starting-point, reorientation, path/progression* and *goal* beforehand.

1.1.3 Autonomous Navigation

The ability of mobile robots to move to a desired target location autonomously, or to navigate in a given behavioral mode while automatically avoiding static and dynamic obstacles, is essential in the development of effective service robot systems. This fundamental characteristic is also a vital object of research in the development of smart wheelchairs. Simpson's general literature overview [46] lists 19 out of 46 smart wheelchair projects that address different approaches to autonomous navigation. In contrast to research studies on specialized experimental wheeled mobile platforms, the quoted projects have often to deal with non-circular shaped vehicles, non-holonomic constrained kinematics, and inaccurate driving characteristics. The latter reason is due to factory-made actuations that are primarily designed to achieve soft and comfortable use for the patient.

Within this thesis, the term *autonomous navigation* corresponds to general *collision avoiding methods* that typically work on data structures that represent the immediate neighborhood of the robot. In this regard, planning approaches that employ global workspace knowledge are explicitly excluded.

Reactive Steering Methods

One of the most prominent navigation methods applied for wheeled mobile robots in general and for electric wheelchairs in particular is the *Nearness Diagram Navigation (NDN)* by Minguez et al. [37, 34] (cf. Fig. 1.1(c) for the development platform). The basic approach can be categorized into the class of reactive, goal oriented, and collision avoiding methods. It classifies a given obstacle configuration along with the desired target location as a unique element of an exclusive and complete set of situations. Each of the six defined situations is associated with a specific control law that determines the translational and rotational speed to be applied for the actual cycle of computation. In order to define a situation, the workspace is divided into sectors centered at the robot's origin. By maintaining a list of distances to the closest obstacles in each of the typically 2° wide sectors, i.e. the nearness diagram, the system is now able to compute free regions in-between two consecutive gaps of the nearness diagram. Finally a navigable region closest to the goal location is selected. This free walking area along with its determining obstacles depicts a specific situation.

Since its first publication, NDN has been extended in various aspects that try to overcome the initial strong assumption of a circular-shaped holonomic robot. The resulting problems for vehicles that do not meet these requirements are specified in section 4.1. In this regard, heuristic extensions of the NDN are described in subsection 4.1.1.

Several appealing properties of reactive navigation strategies, like low computational requirements and the avoidance of explicit modeling of all possible trajectories led to a wide acceptance in the field of autonomous wheelchair navigation. Beside NDN, the *Virtual Force Field Method (VFFM)* by Borenstein et al. [5] has become very popular, for instance as the navigational basis for the NavChair [30]. The steering method models the robot as a point-shaped particle influenced by repulsing forces from the surrounding obstacles, and an attracting force based on the desired target location. The sum of all forces determines the imminent direction of motion and thus the translational and rotational velocity to be commanded. Like other reactive navigation approaches, VFFM can be criticized for its drawback to be potentially trapped within local minima, e.g. as they occur with u-shaped obstacles.

Geometric Path Planner

In contrast to the reactive navigation strategies presented above, geometric path planning algorithms not only determine a desired spatial locomotion for the next time step, but compute feasible and obstacle-free paths from the current pose of the system towards the aimed target. In [29] Latombe categorizes the available approaches into *roadmap methods*, *cell decomposition methods*, *potential field methods*, and planning methods that explicitly employ mathematical curves to model the motion of the robot. The first two approaches capture the connectivity of the robot's free space $\mathcal{C}_{free}$ by constructing a network of feasible paths, or by describing $\mathcal{C}_{free}$ using a set of non-overlapping regions along with its connectivity information. Actual path planning is done by common graph search algorithms. Roadmap methods and cell decomposition methods have been rarely used in smart wheelchair projects for dynamic obstacle avoidance, since they rather stand for global path search algorithms. A state of the art overview is given in [21]. Potential field methods originated from on-line collision avoidance methods like the VFFM. By combining them with graph search techniques that employ further workspace knowledge, the problem to be trapped within local minima can be avoided. Exemplary implementations of potential field navigation algorithms for automated wheelchairs have been described for the *GRASP Laboratory SmartChair* [40], *Robchair* [41], and in [52].

The last class of geometric path planning approaches realizes goal-oriented motion by the computation of mathematical curves that are obstacle-free in the sense that a contour of the robot shifted tangentially along the path does not intersect with any obstacle point from a given occupancy map. The *Dynamic Window Approach (DWA)* [11] that has become very popular for a certain class of wheeled mobile robots, employs circular arcs of varying curvature to model the search space of translational and rotational velocities, reachable by the robot within a short time interval. To the best of the author's knowledge, the original DWA has not been successfully implemented on automated wheelchairs. A reason for this can be seen in the original assumption of a circular-shaped synchro-drive robot. Common non-holonomic wheelchairs do not meet this constraint, i.e. they cannot independently control their translational and rotational velocity.

1.2 Contribution of this Work

This doctoral thesis elaborates the conception and implementation of a smart wheelchair, dedicated to bridging the gap between laboratory demonstrators and real life application. Despite the broadness of the addressed tasks, ranging from hardware integration to user interface design and implementation of navigational algorithms, particular scientific advancements can be clearly highlighted. The following paragraphs summarize the major contributions.

The initial assembly of the *Bremen Autonomous Wheelchair Rolland III* (cf. Fig. 1.1(e)) has been realized in context of this work. First described in [MHV05], the installation of the two *Siemens LS4* laser range finders can be characterized in that both the front and the back scanner sense the environment at a height of approximately 12 cm, parallel to the ground. Scanning beneath the patient's feet does not affect any wheelchair utilization, hence can be seen as a constructive advancement. The second and probably more important contribution in [MHV05] is the implementation of the *Route Graph* concept (cf. chapter 2). This publication describes the annotation of a multi-layered and graph-structured spatial representation of the environment with automatically detected landmarks, here doors. For this reason the experiments performed verified a sensor fusion algorithm that employed classifiers based on data from computer vision and laser range finders. The latter one has been developed in the scope of this thesis.

In [KBFL$^+$05] the work on route graphs is deeply elaborated. The semi-automatic generation from CAD-blueprint maps, and the automatic on-line generation from laser range finder measurements can be ascribed to this work. The related algorithms are based on the generation of *Voronoi Graphs* that include metrical and topological information. Based on the stable and modular implementation, the scientific contribution is given by a comprehensive spatial representation that can be easily utilized by subsequent algorithms. Especially the intrinsic embedding of fuzzy heuristics that implement miscellaneous spatial relations can be seen as an advancement.

The most eminent contribution of this thesis is the investigation of adequate user interfaces for the operator of a smart wheelchair (cf. chapter 3). The methods that originated in this context cover natural language interpretation as well as the design of the head-joystick, a special input device for the paralyzed. First described in [MFR06], the development of an algorithm for the interpretation of qualitative verbal route descriptions denotes a significant advancement for the semantic interpretation of human way descriptions in the mobile robot navigation domain. The overall algorithm is able to map segments of the described route onto the route graph of the environment. It uses fuzzy heuristics to interpret spatial relations along with their referenced landmarks. The most plausible target location is finally computed by a newly developed depth-first branch-and-bound search function. This algorithm operates on a search tree the nodes of which represent fuzzy rated places that result from the interpretation of atomic elements of the route description. By means of several experimental test runs the entire approach has proven its applicability, resulting in an implementation as the core engine of the *Qualitative Spatial Action Model* [SMR07].

Despite the promising results from the proposed speech interface, the inherent complexity of natural language and the resulting difficulties in comprehensive computational treatment have led to the development of a new interface device that better suits the needs of daily use. For patients that cannot move their arms, e.g. people suffering from high level quadriplegia, a head-joystick based on a small-size 3-DOF inertial measurement unit has been developed in the scope of this work. First described in [MRF07], the device is mounted at the back of the user's head by an easy-to-wear headband where it permanently measures the tilt angles in pitch and roll direction. In the primary usage scenario the user is directly embedded in the navigational control loop of the wheelchair, yielding a quantitative joystick controlled by head movements. In an experimental evaluation phase more than 30 untrained participants tested the basic head-joystick. The observed shortcomings led to several design modifications that go beyond comparable state of the art approaches in terms of navigational precision and ease of use, see [MFR07]. Examples here are higher dimensional transfer functions that map head posture angles onto translational and rotational velocities, and an algorithmic damping mechanism that attenuates mechanical feedback effects between pitching the head down and increasing the translational velocity.

A further usage scenario has been developed by the introduction of a qualitative head-joystick. In this case, coarse tilt-gestures of the user's head are classified as directional commands, e.g. *"turn left"*, *"go ahead"*, or *"go forward right"*. Mapped onto the route graph representation of the surrounding environment by utilizing fuzzy heuristics for the spatial relations involved, the system triggers autonomous navigation to the computed target. Beside initial experimental validation of this approach, a slightly modified version in which the operator verbally selects the desired target within the local environment has been evaluated by 15 participants. The gathered findings of both concepts have been published in [MRF07, MF07].

The final contribution dealing with the development of sophisticated user interfaces is given by a prototypical implementation of a navigation module that handles directional commands coming from a brain-computer interface (BCI). First published in [GAM+07], the underlying navigational approach accepts single navigational instructions such as *"take the next turn to the right"*, or *"go straight ahead"*, and safely executes them in hallway areas of a typical office building. Although there has been only one subject assessed to date, it could be shown that low-bandwidth control data coming from a BCI ($\sim 30-60\,bits/min$) suffices to implement a reliable navigation module for unmodified environments. With an upcoming publication that is in preparation at the time, the proposed approach will show its advantages compared to similar approaches like that of Rebsamen et al. [43].

The second major part of this thesis deals with the development of an autonomous navigation architecture that comprises a geometric path planning method along with an underlying path and velocity controller (cf. chapter 4). The essential part, the design of the *Cubic Bezier Curve Path Planner (CBCPP)*, was first published in [MFR06]. It features the planning of an obstacle-free trajectory that moves the wheelchair from its actual state, i.e. position (x_s, y_s), rotation θ_s and translational velocity v_s, into a desired goal configuration $(x_g, y_g, \theta_g, v_g)$. Unlike comparable approaches, necessary veer-off movements for entering narrow passages are explicitly modeled. It shall also be pointed out that in contrast to reactive steering methods, the CBCPP is capable of incorporating the desired heading of the robot at the target location. Thus global path planning algorithms, e.g. A*-based graph search on the route graph representation of the environment, can facilitate smooth transitions between locally planned paths.

Chapter 2

Route Graphs as Spatial Representations

The internal representation of a robot's workspace is the most important source of information for planning, execution, and monitoring operations. Constructed online from available sensors, or given a priori by means of manual or automatic training sessions, information is held by data structures that are required to be *comprehensive*, *well-structured*, and *memory saving*. Within this chapter the concept of a graph-structured and multi-layered *RouteGraph* [53, 23] is shown to fulfill the given requirements. On the basis of a modular and stable implementation that has been used exhaustively in the context of the interpretation of verbal route descriptions (cf. subsection 3.1.2), advantages with respect to different approaches are presented.

2.1 Spatial Representations for Wheeled Mobile Robots

Several decades of wheeled mobile robot research have led to a variety of approaches dealing with the administration of environmental data suitable for miscellaneous navigation tasks. They range from purely metrical over topological representations up to hybrid approaches that try to integrate the advantages of both concepts. Kuiper's *Spatial Semantic Hierarchy (SSH)* [24] is a prominent example for a topological knowledge representation. The SSH models large-scale space information that extends the sensorial horizon by the following four layers:

1. *Control Level* – At the control level, a set of distinct *control laws* like "*follow the left wall*" are associated with distinguishable segments of the robot's workspace. Control laws directly work on the sensorial input and transfer the agent to so-called *distinct states*, i.e. unambiguous states of the robot in its workspace. Typical control laws are hill-climbing or trajectory following.

2. *Causal Level* – The causal level consists of abstractions of sequences from the control level in the form $< V, A, V' >$. V and V' denote sensorial *views* of two distinct states, and A corresponds to the sequence of *actions* or control laws that are to be executed in order to transfer the robot from the first distinct state to the next one.

3. *Topological Level* – The topological level now describes the environment in terms of zero-dimensional *places*, one-dimensional *paths* that connect places, and two-dimensional *regions* that may be defined by one or more bounding paths. Views and actions are

used to abduct the connectivity properties of places and paths. Regions basically define sets of places and can be abstracted to places at a higher level of granularity of the topological map.

4. *Metrical Level* – In the basic SSH, the metrical level represents a global metrical map of the environment. Although the single frame of reference allows to associate quantitative metrical information with places, paths, regions, actions and views, the metrical level seems somewhat loosely connected to the SSH's remaining components.

In order to incorporate the vast developments on metrical SLAM approaches in the last years, Kuipers et al. have more recently proposed the *Hybrid Spatial Semantic Hierarchy (HSSH)* [25]. Note that the HSSH has been evaluated on the experimental robotic wheelchair *Vulcan* of the University of Texas at Austin (cf. Fig. 1.1(d)). In the HSSH, *local perceptual maps (LPMs)* of simple local regions that result from the application of an online SLAM method are used to construct the topological level that itself represents large scale space, potentially including multiple nested loops. The process of constructing global consistent topological maps from LPMs is described in detail by Beeson et al. [4].

2.2 Route Graphs

The instantiation of the route graph concept applied throughout this work complies with the formal specification presented in [KBFL$^+$05]. Generally, a route graph is a multi-layered and graph-structured representation of the environment in which each graph layer describes the workspace on a different level of abstraction. The abstraction of several zero-dimensional places on a lower layer to two-dimensional regions represented by places on an upper layer is represented by connections between the corresponding nodes of both layers. In the following subsection, the structural elements of the implemented route graph are defined.

2.2.1 Structural Route Graph Elements

The implemented *Route Graph* $\mathcal{RG}$ (2.1) is a triple containing two layers of directed subgraphs, i.e. the *Voronoi Layer* $\mathcal{VL}$ and the *Topological Layer* $\mathcal{TL}$, as well as a set of *Transitions* T, establishing the connections between both layers.

$$\mathcal{RG} := (\mathcal{VL}, \mathcal{TL}, T) \tag{2.1}$$

Both the Voronoi layer $\mathcal{VL}$ and the topological layer $\mathcal{TL}$ (2.2) share common graph properties defined by the abstract *Graph Layer* $\mathcal{GL}(k)$. All structural route graph elements that depend on the type of graph layer they belong to are parameterized by a certain *Kind* k, the possible values of which are enumerated in the set K (2.3) for this presentation.

$$\begin{array}{lcl} \mathcal{VL} & := & \mathcal{GL}(voronoi) \\ \mathcal{TL} & := & \mathcal{GL}(topological) \end{array} \tag{2.2}$$

$$\begin{array}{lcl} K & := & \{voronoi, topological\} \\ VK & := & \{withGP, annotationBase\} \\ TK & := & \{room, region\} \end{array} \tag{2.3}$$

An abstract graph layer $\mathcal{GL}(k)$ (2.4) is defined by a set of *Places* $P(k)$, a set of *Route Segments* $RS(k)$, and a set of *Routes* $R(k)$. While places define spatial locations or regions in the environment, route segments connect two places of the same kind, stating the possibility to navigate in-between these two locations.

$$\mathcal{GL}(k) \;:=\; (P(k), RS(k), R(k)), k \in K \tag{2.4}$$

The definition of a place $p(k)$ (2.5) depends on its kind k. A Voronoi place $p(voronoi)$ is basically a node from the Voronoi graph representation of the environment (cf. subsection 2.3.1 for the description of its construction). Thus $p(voronoi)$ is specified by its x and y coordinates with respect to a global coordinate frame, and the distance δ to the next obstacles whose representation is given by the set of generating points GP. Finally $p(voronoi)$ is enhanced by its *Subkind* $vk \in VK$ (2.3). Possible assignments are *withGP* indicating a common place resulting from the underlying Voronoi graph representation, and *annotationBase* denoting the place within the Voronoi layer's frame of reference where an automatic feature detector or the user had instructed the system to annotate the route graph by a sensorial view (cf. subsection 2.3.3).

A topological place $p(topological)$ on the topological layer abstracts a set of Voronoi places to a region. Although including x and y coordinates that are derived from the center of the convex hull of the constituting Voronoi places, its main intention of usage is to exploit connectivity information among multiple topological places. The final flag within the definition of $p(topological)$ is its subkind $tk \in TK$ (2.3). The current implementation allows for the assignment of the labels *room*, and *region*.

$$\begin{array}{lll} P(voronoi) & \subset & \{(x, y, \delta, GP \subset \{(x, y) \mid x, y \in \mathbb{R}\}, vk) \mid x, y, \delta \in \mathbb{R}, vk \in VK\} \\ P(topological) & \subset & \{(x, y, tk) \mid x, y \in \mathbb{R}, tk \in TK\} \end{array} \tag{2.5}$$

A directed route segment $rs(k)$ (2.6) specifies a connectivity relation between a start-place p^s, and a goal-place p^g, both of the same kind. Whereas a route segment on the Voronoi layer explicitly guarantees a minimal clearance to the surrounding obstacles, route segments on the topological layer solely state the possibility to navigate from p^s to p^g. It shall be mentioned that pathological route segments with $p^s = p^g$ are not allowed.

$$RS(k) \subset \{(p^s, p^g) \mid p^s, p^g \in P(k), p^s \neq p^g\}, k \in K \tag{2.6}$$

Several directed route segments of the same kind can be linked together, yielding a route $r(k)$ that defines a potentially complex navigational pattern. Nontrivial routes that consist of more than one element require two consecutive route segments to share one common place. Definition (2.7) restricts routes to be acyclic.

$$R(k) \;:=\; \left(RS^i(k) \mid i \in \mathbb{N}, k \in K, \begin{array}{l} RS^i(k).p^g = RS^{i+1}(k).p^s \wedge \\ RS^i(k).p^g \neq RS^j(k).p^s \forall i \geq j \end{array} \right) \tag{2.7}$$

The abstraction of precise metrical information at the Voronoi layer along with its global frame of reference to elements at the topological layer is provided by the route graph's set of transitions. A single transition $t \in T$ is defined by a pair of places (2.8), i.e. a Voronoi place $p^a \in P(voronoi)$ and a topological place $p^b \in P(topological)$.

$$T \;\subset\; \left\{(p^a, p^b) \mid p^a \in P(voronoi), p^b \in P(topological)\right\} \tag{2.8}$$

2.3 Route Graph Algorithms

2.3.1 Voronoi Graph Construction

The Voronoi graph representation of the environment that makes up the fine grained part of the route graph can be seen as the medial axis transformation of a two dimensional embodiment of the surrounding's free and occupied space. In order to describe its construction and basic properties (cf. [2] for a detailed survey), the *Evidence Grid* and the *Distance Grid* are first described as the Voronoi graph's computational basis.

The evidence grid (cf. Fig. 2.1(b)), either derived from local sensor views or a global CAD-blueprint (cf. Fig. 2.1(a)), is basically defined as a stochastic occupancy grid. It maintains a two dimensional array of cells such that each cell stores the evidence that it is occupied by an obstacle. In its formal definition (2.9) $EGC(x, y) = 0$ denotes a surely unoccupied cell, and $EGC(x, y) = 1$ denotes a surely occupied cell.

$$EGC : \mathbb{N} \times \mathbb{N} \to [0...1] \tag{2.9}$$

The distance grid (cf. Fig. 2.1(c)) consists of cells as defined in (2.10) and has the same dimensions and resolution as the evidence grid. It is calculated by a fast two pass sweep line algorithm that computes for each free cell the metrical distance to the next occupied cell.

$$\begin{aligned} &DGC : \mathbb{N} \times \mathbb{N} \to \mathbb{R} \\ &DGC(x, y) = \min_{x',y' : EGC(x',y') > 0.5} \mid \overrightarrow{(x, y)} - \overrightarrow{(x', y')} \mid \end{aligned} \tag{2.10}$$

The subsequent computation of the Voronoi layer $\mathcal{VL}$ is done in two steps. First of all the algorithm computes for every $DGC(x, y)$ whether the distance between two of its generating points, e.g. the occupied cells $EGC(x', y')$ and $EGC(x'', y'')$ that gave $DGC(x, y)$ its value, is greater than a given threshold ϵ. In the formal definition of the resulting *Voronoi Diagram* VD (2.11), the constant value ϵ determines the minimal free space that is required to mark a region as navigable by a route segment of the Voronoi layer.

$$VD := \left\{ (x, y) \;\middle|\; \begin{array}{c} x, y \in \mathbb{N}, \exists\, x', y', x'', y'' \in \mathbb{N} : \\ EGC(x', y') > 0.5 \wedge EGC(x'', y'') > 0.5 \wedge \left| \begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} x'' \\ y'' \end{pmatrix} \right| > \epsilon \\ \wedge \left| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x' \\ y' \end{pmatrix} \right| = \left| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x'' \\ y'' \end{pmatrix} \right| = DGC(x, y) \end{array} \right\} \tag{2.11}$$

The final step searches the Voronoi diagram VD for elements that hold one or more than two neighbours in VD. These cells correspond to terminating or branching places respectively, and are inserted into the Voronoi layer's set of places $\mathcal{VL}.P(voronoi)$. The Voronoi layer's set of route segments $\mathcal{VL}.RS(voronoi)$ comprises pairs of references to elements in $\mathcal{VL}.P(voronoi)$ that are connected by points out of VD. The resulting Voronoi layer $\mathcal{VL}$ is illustrated in Fig. 2.1(d).

2.3.2 Embedded Spatial Relations

Implemented on the smart wheelchair Rolland, the route graph representation of the environment primarily serves as a knowledge representation for the vehicle's navigational tasks.

In order to satisfy the structural condition of a shared reference system between the robot and its operator, a systematic approach that allows to negotiate amongst human's spatial concepts and the system's metrical and topological representations of the environment has been developed. Applied in the tasks of route graph annotation, i.e. verbal object localization by the user that enables the system to expand its knowledge base (cf. subsection 2.3.3), and for the interpretation of coarse verbal route descriptions (cf. subsection 3.1.2), fuzzy spatial relations serve as the computational basis.

In the current implementation, the spatial relations *after*, *along*, *between*, *out of*, *through*, *until*, and the generic relation *natural-language-direction* have been implemented (cf. Fig. 2.2 for an illustration). All of them judge how well the spatial configuration of a given *relatum* can be described by the tested spatial relation to a given *ego*, possibly involving the location of a given *referent*. With regard to the coarse nature of the spatial relations, the underlying fuzzy functions (cf. [MFR06] and [SMR07] for a detailed elaboration) map the situation to be analyzed to the domain [0...1].

2.3.3 Route Graph Annotation

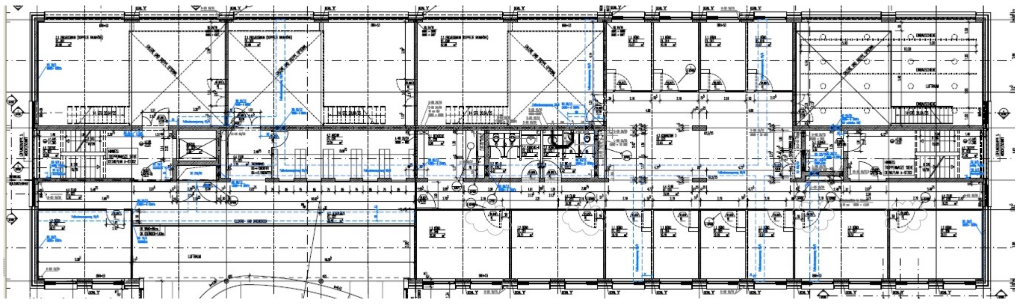
Apart from the application of automatic feature detectors (cf. [MHV05] for the conception of laser range finder and vison-based door detectors), the evaluation of indications from the operator of the smart wheelchair can force the system to augment its knowledge about the environment at runtime. Spatial statements such as *"the fridge is to the right"* induce the system to insert a place of type $\mathcal{P}(voronoi)$ with subkind $vk = annotationBase$ into the route graph's Voronoi layer $\mathcal{RG}.\mathcal{VL}$. The inserted place is located at the position calculated by the global localizer module [44] and augmented by the laser scans taken at the time the user gave this statement, as well as the label of the mentioned landmark (cf. Fig. 2.1(d) for the illustration of an annotated route graph). A key-feature of this approach is the possibility to employ the route graph's embedded spatial relations. The most relevant one, the so-called natural-language-direction that has been first described in [MHV05], can be used either as a binary or as a ternary relation such as *"the fridge is to the left of the sink"*. To take account of the fact that humans use directions at different levels of granularity, four kinds of natural-language-direction have been defined, each of a different resolution. They range from *NLD::2Clock* which solely differentiates left and right, to *NLD::12Clock* in analogy to the directions pointed to by a watch-hand denoting clock hours. All kinds of natural-language-direction support queries that interrelate the structural route graph components place and route segment. Since routes divide a given plane into two regions, one located to the left of the directed route and one to the right, only *NLD::2Clock* queries are allowed here.

2.4 Contribution of the Corresponding Publications

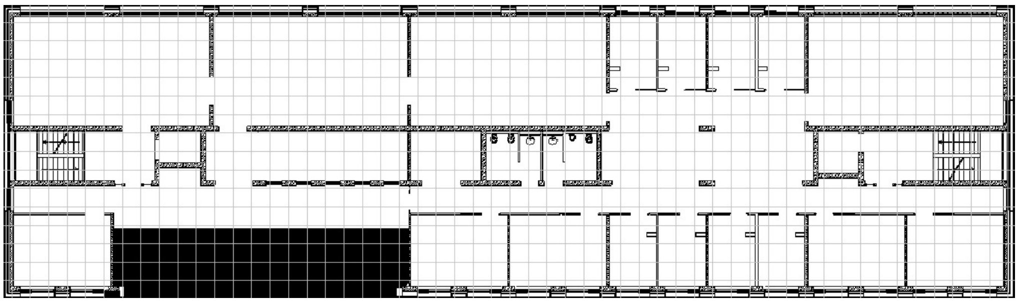
A major contribution of this thesis's publications that can be associated with this chapter is the complete application driven implementation of the route graph concept. First proposed in [MHV05], the presented framework accounts for a scalable spatial data structure that can be calculated either from local sensor-based grid maps or from pre-existing global grid maps derived from CAD blueprints. It has been shown that spatial annotations triggered by sensor-driven feature detectors or human utterances provide an important source of information for subsequent queries and navigational tasks.

The theoretical contribution to the underlying route graph structure has been incorporated into [KBFL$^+$05]. It comprises the adoption of Voronoi graphs as the most fine grained route graph layer for wheeled mobile indoor robots. In this context, the specification of requirements for topological Voronoi layer attributes, e.g. distance information along route segments and directional information at junction places, lays the foundation for the implementation of fuzzy spatial relations that interrelate route graph entities.

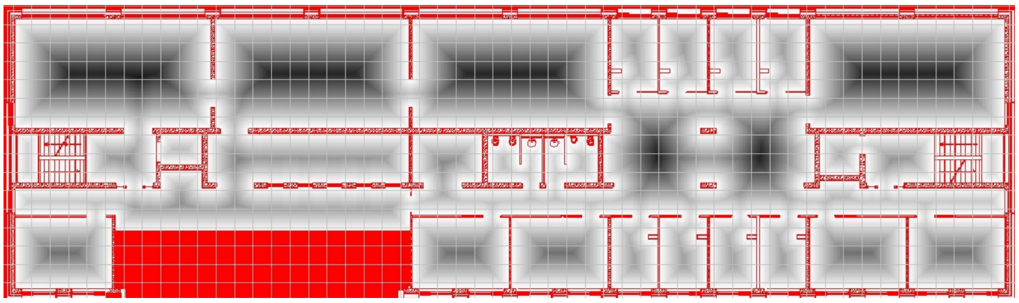
The set of investigated spatial relations that has been described in [MFR06, SMR07] not only covers the functionality of common directional calculi, but extends it by the algorithmic treatment of further spatial prepositions that frequently appear in human way descriptions. By means of a continuous integration of the corresponding heuristics into the route graph framework, their straightforward application by subsequent algorithms is provided for.



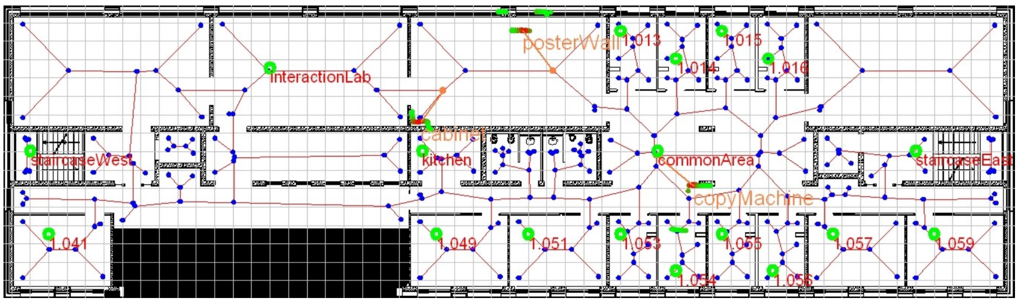
(a) Raw information of the global workspace is given by a vectorial CAD-blueprint, annotated by a variety of architectural data.



(b) A manual thining operation and a rasterization of the vector format yields a bitmap, i.e. the global evidence grid.

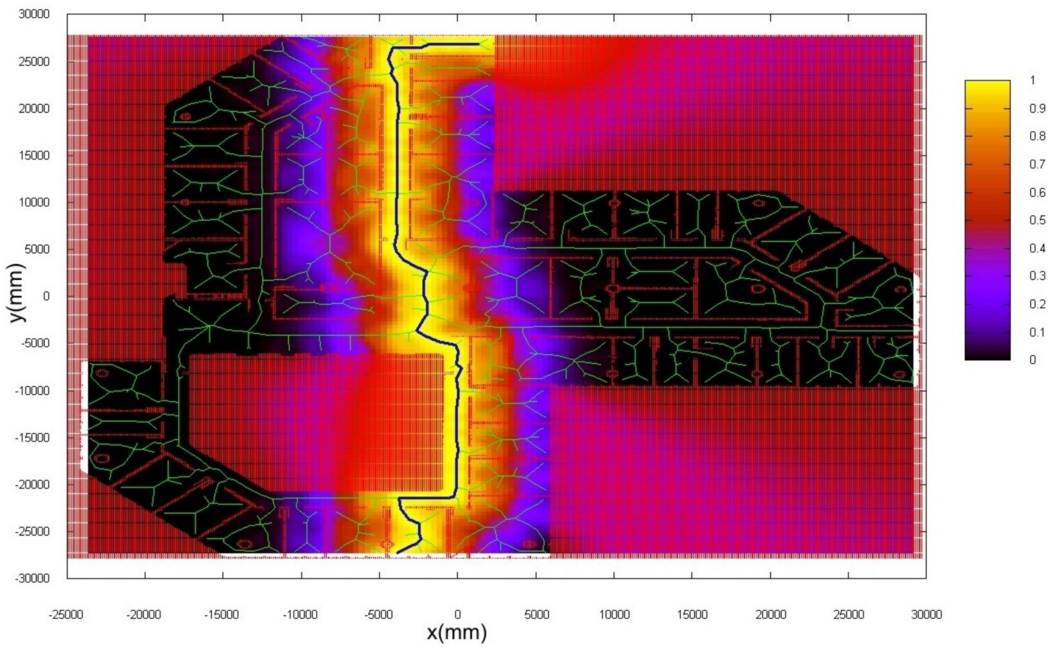


(c) The global distance grid DG_g stores for each free cell the distance (ranging from close/white to far/black) to the closest obstacle(red).

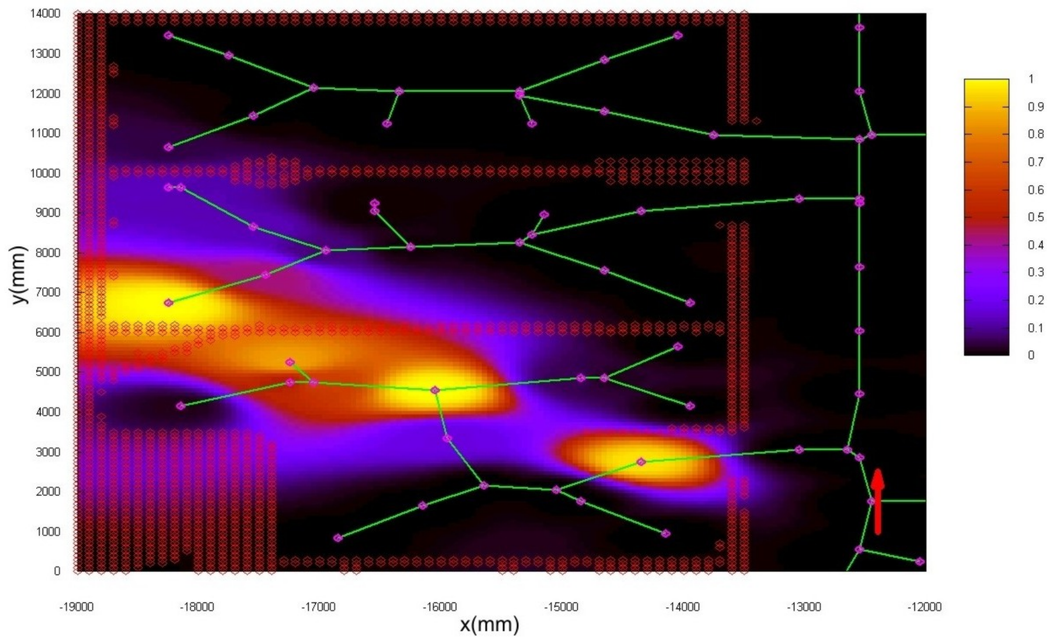


(d) The current implementation of the Voronoi graph generation approximates the ridge of the global distance grid DG_g. User initiated or automatic annotations of rooms, regions and distinctive landmarks are stored at the topological layer.

Figure 2.1: The semi-automatic generation of a buildings's route graph representation involves several steps ranging from manual thinning of CAD-blueprint information, over Voronoi graph generation, place and region labeling, to the annotation of sensorial views.



(a) Places situated *along* the blue coloured route.



(b) Places to the *front-left* of the red coloured pose.

Figure 2.2: Illustration of exemplary spatial relations that correlate an agent's pose within the environment to the route graph's structural components, i.e. places, route segments, and routes. Yellow coloured parts of the environment mark regions that match best with the combination of the analyzed spatial relation, ego, referent, and relatum. The non-uniform circumference of positively rated areas, e.g. the somewhat irregular sector to the front-left of the red coloured pose in Fig. 2.2(b), results from the interpolation of the functional of the route graph's sparse base data.

Chapter 3

Interfaces for Navigation Control

Usually electrical wheelchairs are operated by joystick. From that perspective they handle the handicapped's inability to walk by a control loop that embeds the operator's remaining sensomotor capabilities into the vehicle's actuating body. Since the most common interface i.e. pushing a joystick into the desired direction of motion, only works for patients that are still able to move their hands, several kinds of interface techniques have been investigated that serve e.g. the quadriplegic (cf. subsection 1.1.2 for an overview). This chapter presents the design of two approaches that facilitate the control of a smart wheelchair by interpreting speech commands and head movements, respectively.

3.1 Verbal Control

3.1.1 Path Selection from an Offered Route-Set

The essential idea behind the navigation via verbal path selection from an offered route-set (cf. [MF07]) is a simple speech interface with which the operator selects a desired movement from a set of graphically presented navigable routes. Derived from the sensor-based Voronoi layer that represents the direct neighbourhood of the wheelchair, the proposed routes are selected by verbally calling their name. In order to improve the performance of the recognition process, each proposed route is labeled by a token out of the ICAO-alphabet[1].

Algorithm 1 sketches the computation of the set of navigable routes. It inputs the current pose of the wheelchair *ego*, and the sensor-based route graph $\mathcal{RG}$, representing the direct vicinity. The actual search for suitable routes (lines 11-16) is done by calling an A* graph search (line 12) that tries to connect the place that represents the current position of the vehicle, i.e. $odometryPlace$ (line 1), to each other place at the Voronoi layer, i.e. $voronoiPlaces$ (line 2). This straightforward strategy requires the temporary modification of the local Voronoi layer. Due to the laser range finder's blind zone to the left and right of the wheelchair,[2] there may be no valid route segments in this area, hence two separated com-

[1]During experimental evaluation, the off-the-shelf speech recognizer *Vocon* from *Nuance Communication Technologies* worked fine with the recognition of elements of the alphabet of the *I*nternational *C*ivil *A*viation *O*rganization. The selected alphabet tries to reduce speech recognizer failures since all words out of the alphabet (*alpha*, *bravo*, *charlie*, ...) are selected in such a way that they are easy to distinguish, e.g. few monosyllabic words that have a mutually different sound.

[2]Fig.2(b) in [MF07] depicts the laser range finder's blind zone by two dark gray areas that face to the left and right of the wheelchair's contour. The outline itself is centered in the pictured evidence grid while facing to the right. During the vehicle's movement, consecutive laser scans are integrated into

Algorithm 1 Compute_Navigable_Route_Set

```
Input: Pose ego, RouteGraph $\mathcal{RG}$
Output: vector <Route> navigableRoutes
1: Place odometryPlace ← Place(voronoi, ego.translation)
2: vector <Place> voronoiPlaces ← $\mathcal{RG}.\mathcal{VL}$.Get_Places()
3: vector <RouteSegment> tempRouteSegments
4: for $i = 1$ to voronoiPlaces.Size() do
5:     if voronoiPlaces[i].distanceTo(odometryPlace) < maxDist2RouteStart then
6:         RouteSegment tmpRS ← RouteSegment(
           voronoi, odometryPlace, voronoiPlaces[i])
7:         $\mathcal{RG}.\mathcal{VL}$.addRouteSegment(tmpRS)
8:         tempRouteSegments.push_back(tmpRS)
9:     end if
10: end for
11: for $j = 1$ to voronoiPlaces.Size() do
12:     Route route ← $\mathcal{RG}.\mathcal{VL}$.graphSearch(odometryPlace, voronoiPlaces[j])
13:     if route.Size() > 1 && !navigableRoutes.containsSuperimposedRoute(route) then
14:         navigableRoutes.push_back(route)
15:     end if
16: end for
17: for $k = 1$ to tempRouteSegments.Size() do
18:     $\mathcal{RG}.\mathcal{VL}$.removeRouteSegment(tempRouteSegments[k])
19: end for
20: return navigableRoutes
```

ponents of the Voronoi layer. The interim inserted route segments (lines 4-10) that connect *odometryPlace* with each place directly surrounding the wheelchairs current position (line 5) are stored in *tempRouteSegments* and assure a successful graph search to potential target places within the circumference of the Voronoi layer. Before returning the vector *navigableRoutes* (line 20), *tempRouteSegments* are removed from the route graph for consistency (lines 17-19).

Apart from the computation of the set of navigable routes, the consistent denomination of spatially equivalent solutions is a major issue. In order to guarantee that the user can select a desired route that equals a formerly proposed one by calling the same token each time, computed routes are currently compared to routes from the former frame. It proved sufficient to judge two routes to be spatially equivalent if the metrical distance between the positions of their target places is less than a given threshold.

3.1.2 Interpretation of Coarse Qualitative Route Descriptions

The approach of interpreting coarse qualitative route descriptions (CRDs) has first been published in [MFR06] and is based on preliminary work by Mueller et al. [38]. The common

the evidence grid, yielding temporal knowledge about the obstacle situation within the blind zone. This knowledge fades away when the vehicle comes to a stop and no new sensor measurements cover the hidden area.

underlying idea is that a verbal route description can be split into parts that describe the start, the progression along the route, intermediate reorientation actions, and the goal. This observation has been validated by empirical studies in which subjects had to give in-advance route descriptions to a designated goal [17, 16], and gives rise to a basic formalization of CRDs as sequences of tuples as in (3.1) (cf. [MFR06] for a comprehensive formalization of CRDs along with supporting corpus examples).

$$< \ [\ \{ \ \mathit{controlmarks} \ \} \ \mathit{router} \ ] \ \mathit{reorientation} \ > \tag{3.1}$$

Controlmarks and *router* specify the location of distinctive landmarks along and at the end of a route segment, respectively. In contrast, a *reorientation* block stands for the directional instruction that aligns the agent to the new route segment. The key feature of both the controlmarks and the router is their definition as a combination of a spatial relation and a referenced landmark. This valuable information is the basis for a newly developed algorithm that maps an agent's start pose and a CRD onto the global route graph representation of the environment, yielding the most plausible target pose described by the underlying verbal route description.

Algorithm 2 basically builds up a search tree, the nodes of which represent fuzzy rated places resulting from the evaluation of single CRD-elements, or rather the interpretation of the associated spatial relations via their fuzzy functions. The recursive algorithm primarily uses the following data structures:

- *crd*: The list of CRD-elements to be evaluated.
- *crdIt*: A pointer to the CRD-element to be evaluated in the current incarnation.
- *st*: The search tree built up so far.
- *stIt*: A pointer to a leaf of *st* that corresponds to *ego*.
- *ego*: Fuzzy rated pose to be considered at the current incarnation.

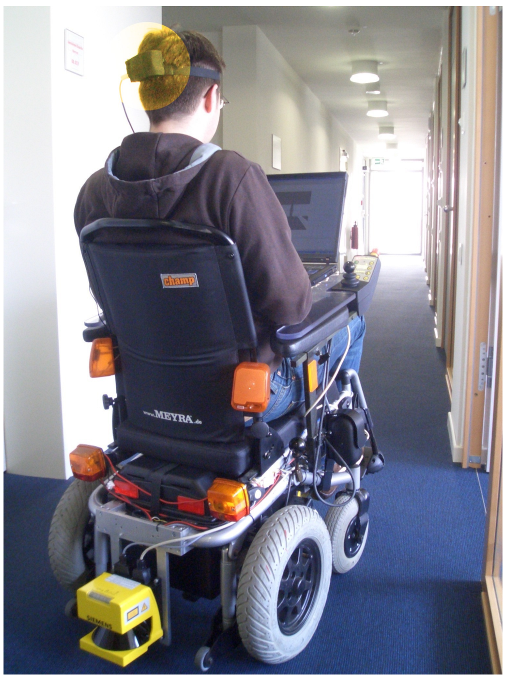
By dint of the spatial relation associated with *crdIt* (line 4), one computes for every possible $referent$ (line 3), or rather neighbouring place of *ego* (line 2), the fuzzy rated pose $actualNode$. If the score of $actualNode$ is bigger than a given threshold (line 5), i.e. the plausibility that the current $referent$ fits sufficiently well with *ego* and the spatial relation to be analyzed, the referent is multiplied by $ego's$ own score (line 6). This mechanism propagates the uncertainty of former fuzzy ratings down to lower levels of the search tree. The algorithm now recursively continues in a depth-first branch-and-bound manner (lines 9,13,25) with the evaluation of the remaining CRD-elements in *crd*. In order to keep the size of the search tree small, the evaluation at a given node only continues if its cumulative score is greater than the highest score of a leaf resulting from the evaluation of the last CRD-element (line 11).

3.2 Control via Head Posture Tracking

People suffering from certain spinal cord injury dysfunctions can only move body parts located above their shoulders. Apart from the proposed speech interfaces (cf. section 3.1),

Algorithm 2 Process_Coarse_Route_Description

```
Input: $FuzzyRatedPose\ ego$,                 $bool\ \&reachedGoal$,
       $list <CRDElement>\ crd$,               $CRDElement\ \&crdIt$,
       $tree <FuzzyRatedPose>\ \&st$,          $FuzzyRatedPose\ \&stIt$,
       $RouteGraph\ \mathcal{RG}$
Output: $FuzzyRatedPose\ bestReturn$
1:  if $crdIt.type = Controller \vee crdIt.type = Router$ then
2:    $vector <Place>\ referents \leftarrow \mathcal{RG}.\mathcal{VL}.Get\_Nearby\_Places(ego)$
3:    for $i = 1$ to $referents.Size()$ do
4:      $FuzzyRatedPose\ actualNode \leftarrow Eval\_Sp\_Rel(crdIt.Get\_Sp\_Rel(),$
            $ego, crdIt.Get\_Relatum(), referents[i])$
5:      if $actualNode.score > MIN\_PROP\_FOR\_SINGLE\_EVAL$ then
6:        $actualNode.score \leftarrow ego.score * actualNode.score$
7:        if $reachedGoal =$ false then
8:          $stIt \leftarrow st.Append\_Child(stIt, actualNode)$
9:          $bestReturn \leftarrow Process\_Coarse\_Route\_Description($
                $actualNode, reachedGoal, crd, ++crdIt, st, stIt)$
10:       else
11:         if $actualNode.score \geq bestReturn.score$ then
12:           $stIt \leftarrow st.Append\_Child(stIt, actualNode)$
13:           $FuzzyRatedPose\ nextNode \leftarrow Process\_Coarse\_Route\_Description($
                                  $actualNode, reachedGoal, crd, ++crdIt, st, stIt)$
14:           if $NextNode.score > bestReturn.score$ then
15:             $bestReturn \leftarrow nextNode$
16:           end if
17:         end if
18:       end if
19:     end if
20:   end for
21: else if $crdIt.type = Action$ then
22:   if $crdIt.actionType = Turn$ then
23:     $ego.rotate(crdIt.Get\_Rot\_Angle())$
24:     $stIt \leftarrow st.Append\_Child(stIt, ego)$
25:     $bestReturn \leftarrow Process\_Coarse\_Route\_Description($
              $ego, reachedGoal, crd, ++crdIt, st, stIt)$
26:   else if $crdIt.actionType = Stop$ then
27:     $reachedGoal \leftarrow$ true
28:     $bestReturn \leftarrow ego$
29:   end if
30: end if
31: return $bestReturn$
```


(a) Subject steering Rolland III by means of the quantitative head-joystick.

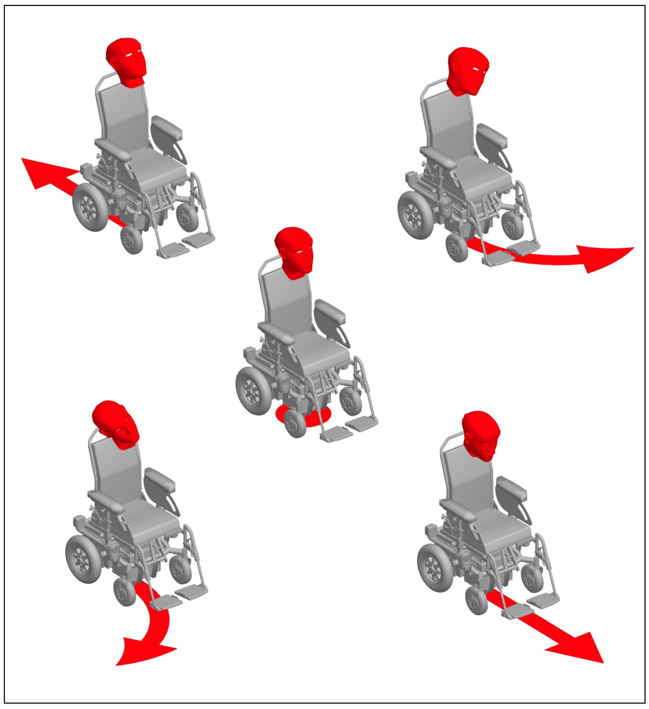

(b) Illustration of a user head's pitch and roll movements triggering translational and rotational movements.

Figure 3.1: Both the quantitative and the qualitative head-joystick employ information about the operator's head posture, coming from a head-mounted 3-DOF inertial measurement unit (cf. yellow highlighted area in Fig. 3.1(a)). For the quantitative head-joystick, the user head's pitch- and roll movements control the wheelchair's translational and rotational velocity (cf. Fig. 3.1(b)).

this thesis proposes the application of a 3-DOF orientation tracker (IMU) as suitable control equipment for the paralyzed. Mounted at the back of the operator's head by means of an easy to wear headband, the small-size device continually monitors the user's head posture and acceleration w.r.t. the pitch-, yaw-, and roll-axis. The following subsections present the so-called quantitative and qualitative head-joystick. Both developments employ the IMU's output data with which they realize a proportional joystick-like interface device and a discrete gesture-based interface technique, respectively.

3.2.1 Quantitative Control via Proportional Control Commands

First proposed in [MRF07] and later evaluated in [MFR07, MF07], the basic idea of the quantitative head-joystick is to let the user of an automated wheelchair control the translational and rotational velocity by continuous pitch and roll movements of his/her head. Still able to observe the environment by turning the head around the free yaw-axis without causing any control commands, the user's head movements around the remaining two axes must exceed a so-called dead-zone in order to evoke a desired movement. Algorithm 3 outlines the quantitative head-joystick's main loop. It uses the following data structures:

- *pitch*, *roll*: The current pitch- and roll-angle of the user's head.

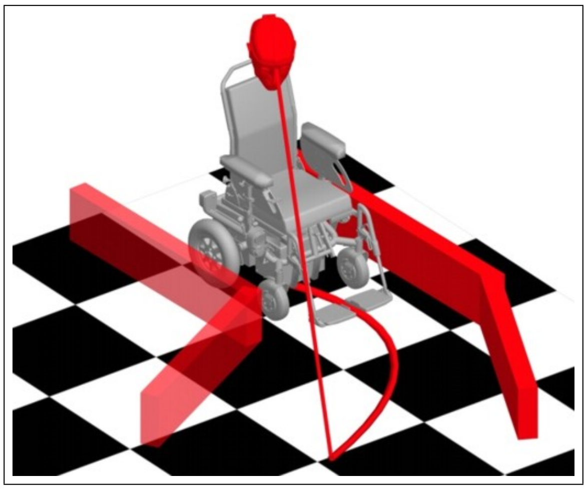
Algorithm 3 Quantitative_Head_Joystick_Main_Loop

```
Input: $Float\ pitch,\ Float\ roll,\ Float\ hPI,\ Float\ hRI,$
       $Float\ dZP,\ Float\ dzR$
Output: $vector <Float>\ speedRequest$
 1: $Float\ V \leftarrow 0$
 2: $Float\ W \leftarrow 0$
 3: if $actualPitch > deadZonePitch$ then
 4:     $V \leftarrow f_V((pitch - dZP)/(hPI - dZP)) * C1$
 5: else if $actualPitch < -deadZonePitch$ then
 6:     $V \leftarrow f_V((pitch + dZP)/(hPI - dZP)) * C1$
 7: end if
 8: $deadZoneRollAdd \leftarrow C2 * ((\mid pitch \mid -dZP)/(maxPitch - dZP)) * (maxRoll - dZR)$
 9: $deadZoneRoll \leftarrow deadZoneRoll + deadZoneRollAdd$
10: if $actualRoll > deadZoneRoll$ then
11:     $W \leftarrow f_W((roll - dZR)/(hRI - dZR)) * C3$
12: else if $actualRoll < -deadZoneRoll$ then
13:     $W \leftarrow f_W((roll + dZR)/(hRI - dZR)) * C3)$
14: end if
15: $speedRequest.push\_back(V)$
16: $speedRequest.push\_back(W)$
17: return $speedRequest$
```

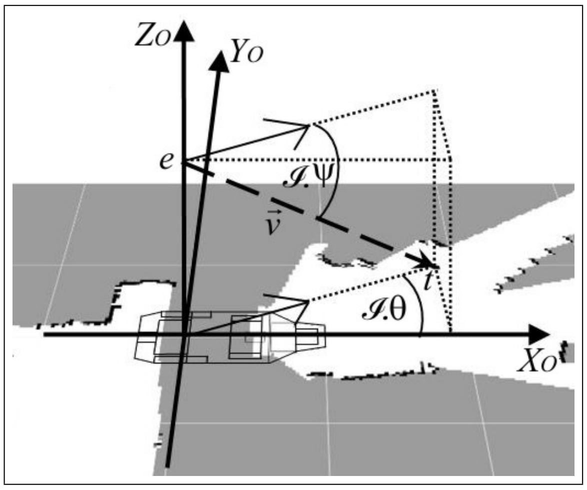
- hPI, hRI: Half of the difference between minimal and maximal pitch and roll deflections that the user was able to accomplish during a calibration phase.

- dZP, dZR: Static dead-zone around the user's head rest position defines the minimal pitch and roll deflection that the user's head has to exceed in order to generate a control command.

The actual computation of the translational velocity V and the rotational velocity W to be returned is done in lines 3-7 and lines 10-14 respectively. Both velocities result from the quotient of the part of the according deflection that exceeds the dead-zone, divided by the magnitude of the deflection's interval valid for generating a control command (lines 4,5,11,13). The resulting normalized control command is subsequently fed into the transfer function f_V or rather f_W, and finally mapped onto the vehicle's velocity domain by multiplying it with $C1$ and $C3$, respectively. In [MFR07] the application of linear, quadratic or cubic transfer functions is discussed w.r.t. the reduction of unintentional oscillations in the driven trajectory. In contrast to the static dead-zone that gives a lower bound for the head's pitch movements resulting in a valid control command, extensive experimental evaluations gave rise to the implementation of a dynamic roll dead-zone (cf. [MFR07]). Originating from the observation that untrained participants tended to oversteer the rotational velocity in situations of high translational speed, e.g. during straight corridor movement, a dynamic modification of the roll dead-zone has been implemented in order to overcome this effect. In lines 8-10 the roll dead-zone is linked to the head's pitch deflection, yielding a magnification of $deadZoneRoll$ for increasing values of $pitch$. Experimental results validate this refinement.



(a) Illustration of target pose selection by intersecting the operator's line of sight with a plain floor. The path connecting the wheelchair's current pose with the selected goal is computed w.r.t. the local obstacle situation.



(b) Evidence grid including the shape of the wheelchair, heading to the right. The selected target position t is determined by the assumed eye-point e, and the direction of the operator's view $\vec{v}$. Vectors and angles are given in the odometry coordinate system O.

Figure 3.2: The current implementation of the qualitative head-joystick lets the user select a desired target pose by facing his/her head to the desired target-position, followed by an affirmative utterance. It is planned to replace the verbal confirmation by head gestures that are easy to recognize.

3.2.2 Qualitative Control by Means of Head Gesture Interpretation

Instead of putting the user of the quantitative head-joystick in a continuous control loop, the developed qualitative head-joystick lets the user issue discrete driving commands by facing his/her head to the desired target-position once, and lets the system autonomously execute the given task. First proposed in [MRF07] as a proof of concept, the underlying idea of letting the user choose a local target position by computing the intersection point between his/her line of sight and the floor is illustrated in Fig. 3.2(a).

The computation of the target pose t comprises two steps (cf. Fig. 3.1(b) for a geometric interpretation). First, a virtual ray of view $\vec{v}$ is assumed to be based in the point e that is located in-between the pilot's eyes. Initially aligned with the wheelchair's heading, $\vec{v}$ represents a shared axis of both the IMU's and the wheelchair's odometry coordinate system. After rotating $\vec{v}$ by the user's head pitch angle Ψ around Y_O, $\vec{v}$ is rotated by the user's head yaw angle θ around Z_O. The resulting vector $\vec{v}$ now represents the direction of the user's view within O and can be intersected with the ground plane, yielding the aimed target position $t = (x_t, y_t)$. In the second step, t is augmented by an appropriate target heading θ_t. Under the assumption that the desired orientation is parallel to the surrounding obstacles, θ_t is calculated as the angle between the tangent to the iso distance lines of the local distance grid at t, and the x-axis of the odometry coordinate system O. The final target pose $t = (x_t, y_t, \theta_t)$ is forwarded to the geometric path planner (cf. section 4.2) and autonomously executed.

It is planned to modify the current implementation into a real gesture-based interpretation of the patient's head movements. Within this updated version the operator will be able to command a desired target position by facing his/her head into the desired direction of movement, followed by a triggering pitch down/up gesture. The head's yaw direction will be discretized into a qualitative directional command, e.g. *left*, *front-left*, *front*, etc., and interpreted as presented in section 3.3.

3.3 Control via Brain-Computer Interface

Brain-Computer interfaces, or BCIs for short, are tools that facilitate communication with artificial artefacts via direct measures of the human brain's activity (cf. subsection 1.1.2 for a brief introduction). This section presents the navigational concept of the smart wheelchair Rolland that was applied in a pilot study, demonstrating the applicability of BCIs as an appropriate interface device. The overall system that has been first descriped in [GAM+07], uses SSVEPs in order to derive directional commands from EEG data. The interpretation of discrete driving commands such as *go straight ahead*, *go left*, *go right*, or *turn around* in a local exploratory navigation scenario basically asks for an adequate projection of the given direction onto the local route graph. The following paragraph describes the algorithmic treatment of the directional commands. A brief illustrative example is given in Fig. 3.3.

As a primary source of information, algorithm 4 inputs the vector of navigable routes *navRoutes* calculated by algorithm 1 and evaluates it against the given directional command *cmd*, i.e. the route *bestRoute* is searched for that best fits the actual instruction. In this context, the principle idea is to iterate over the ordered sequence of each route's (line 2) route segments (line 4), and to assess the angle between two consecutive sections by dint of the generic spatial relation *natural language direction* (cf. subsection 2.3.2). This approach differentiates two situations. In the case of evaluating a *go straight ahead* command (lines 5-10), a single route's score Σ, describing the plausibility to comply with the given instruction, is formulated as the product of all intermediate scores σ_{front} (lines 7,9). In this case σ_{front} results from the evaluation of *NatLangDir.front*. Accordingly, an exemplary route consisting out of three route segments would gain an overall score of 1, if all three sections are perfectly aligned to each other, and a score of 0, if each but the first route segment turns off in a 90° angle from its predecessor. The treatment of commands that introduce a bending manoeuvre (lines 11-16), e.g. *go right*, has to be different because of the additional necessity to choose the best suiting place of a given route for the triggering directional command. The actual approach iterates over all places of a route to be evaluated, and calculates for each choice the route's score Σ as the product of the scores σ_{front} that arise from the evaluation of *NatLangDir.front* between all pairs of consecutive route segments (lines 13,14). The only exception is given by the two sections connected to the selected branching place, where the score's factor σ_{turn} is given by the evaluation of the directional relation that corresponds to the given command (line 12).

Algorithm 4 proceeds by saving the best possible score for the actual route (lines 18-28), and finally returns the route with the maximal overall score (line 36), found by searching the vector *routesScores* of all Σ (lines 30-35). Although not algorithmically sketched, it shall be mentioned that a given turn command is converted into a straight ahead command once the target place of the corresponding route is reached. This behaviour seems adequate since the user probably wants to continue with a straight ahead motion after taking a specific turn. In contrast, a straight ahead command only expires by a subsequent command from the user.

Algorithm 4 Process_Directional_Route_Command

```
Input: $vector <Route>\ navRoutes,\ \ NatLangDir\ cmd$
Output: $Route\ bestRoute$
1: $vector <float>\ routesScores$ // containing a $\Sigma$ for each route
2: for $i = 1$ to $navRoutes.Size()$ do
3:    $vector <float>\ routeScores$ // containing a $\sigma$ for each route segment
4:    for $j = 1$ to $navRoutes[i].Size() - 1$ do
5:       if $cmd = NatLangDir.front$ then
6:          if $j = 1$ then
7:             $routeScores.push\_back(Get\_Nat\_Lang\_Dir\_Score(NatLangDir.front,$
                        $navRoutes[i][j], navRoutes[i][j+1]))$
8:          else
9:             $routeScores[j] \leftarrow routeScores[j] * Get\_Nat\_Lang\_Dir\_Score($
                  $NatLangDir.front, navRoutes[i][j], navRoutes[i][j+1])$
10:         end if
11:      else if $cmd = NatLangDir.left \vee cmd = NatLangDir.right$ then
12:         $routeScores.push\_back(Get\_Nat\_Lang\_Dir\_Score(cmd,$
                     $navRoutes[i][j], navRoutes[i][j+1]))$
13:         for $k = 1$ to $routeScores.Size() - 1$ do
14:            $routeScores[k] \leftarrow routeScores[k] * Get\_Nat\_Lang\_Dir\_Score($
                  $NatLangDir.front, navRoutes[i][j], navRoutes[i][j+1])$
15:         end for
16:      end if
17:   end for
18:   if $cmd = NatLangDir.left \vee cmd = NatLangDir.right$ then
19:      $int\ bestScore \leftarrow 1$
20:      for $l = 2$ to $routeScores.Size()$ do
21:         if $routeScores[l] > routeScores[bestScore]$ then
22:            $bestScore \leftarrow l$
23:         end if
24:      end for
25:      $routesScores.push\_back(routeScores[bestScore])$
26:   else
27:      $routesScores.push\_back(routeScores.back())$
28:   end if
29: end for
30: $int\ bestReturn \leftarrow 1$
31: for $m = 2$ to $routesScores.Size()$ do
32:    if $routesScores[m] > routesScores[bestReturn]$ then
33:       $bestReturn \leftarrow m$
34:    end if
35: end for
36: return $navRoutes[bestReturn]$
```

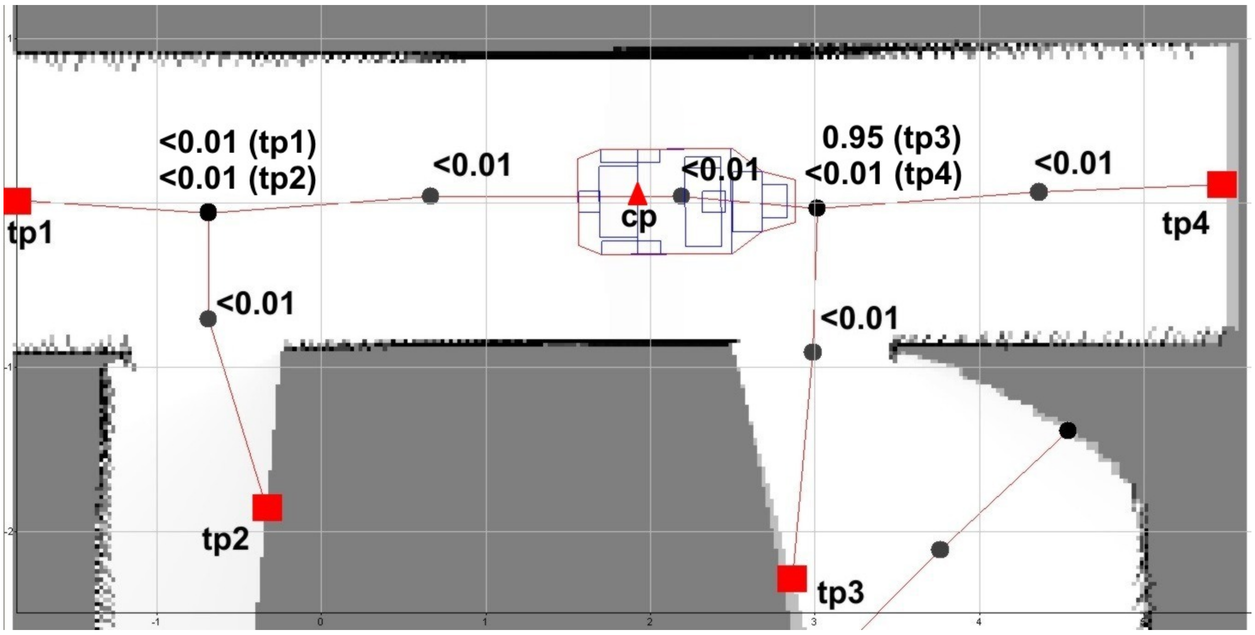


Figure 3.3: Illustrative example for the interpretation of the qualitative instruction *go right*, produced from simulated data. In a first step the algorithm computes all routes from the place *cp* that represents the current position of the wheelchair, to the potential target places $tp1, ..., tp4$. After this, each place of a single route is assessed by a score δ that determines its plausibility to be a place where the incoming and outgoing route segments branch to the commanded direction, i.e. to the right. The final score Δ of a specific route is maximized over the selected branching place, and computed as the product over the single scores δ_i. Note that we multiply δ_i for the assumed branching place and $(1-\delta_i)$ for the remaining places. This yields $\Delta = (< 0.01) * 0.95 * (< 0.01) >= 0.93$ for the selected route from *cp* to *tp*3.

3.4 Contribution of the Corresponding Publications

The major contribution of this thesis's publications dealing with the investigation of human-robot interaction is the development of two novel user interfaces for the smart wheelchair Rolland. From the perspective of verbally instructed robot navigation, the approach for the interpretation of coarse qualitative route descriptions enhances comparable methods in several ways. While previous work (e.g. Kyricao et al. [27]) solely interprets single verbal instructions by mapping them onto the current sensorial view, the algorithm introduced in [MFR06, SMR07] is able to process sequences of atomic instructions w.r.t. local and global world knowledge. The resulting search tree not only povides the most suitable target pose for a subsequent navigation task, but also the complete set of potential interpretations of the route instruction.

The second presented user interface, a head-joystick based on a 3-DOF orientation tracker, improves comparable devices in both of its described designs, i.e. configured as a quantitative and as a qualitative head-joystick. The former one, whose different stages of evaluation have been published in [MRF07, MFR07, MF07], lets the user control the wheelchair's translational and rotational speeds continuously. By contrast, similar systems described in

[6] by Bureau et al., and in [8] by Chen et al., only allow for a small set of discretized speeds, triggered by unnatural repetitive head movements. The qualitative head-joystick that has been first proposed in [MRF07] extends the quantitative interpretation of the user's head posture in that it evaluates coarse head gestures in order to derive a suitable target pose that can be subsequently piloted. Despite loosely related approaches that involve more complicated hardware setups, e.g. a computer vision based system [7] by Canzler et al., there exist no comparable systems that build on the intelligent analysis of the user's head gestures.

With the presentation of a navigational concept that is able to execute coarse directional commands comming from a BCI, this thesis concludes its contributions to interfaces techniques for a smart wheelchair. First proposed in [GAM$^+$07], the presented approach can be seen as a proof of concept that validates the feasibility of mapping instructions like *go right* onto the route graph representation of the environment. A journal paper that comprises exhaustive experimental data is in preparation at this point in time.

Chapter 4

Autonomous Local Navigation

Autonomous navigation approaches that provide the ability to deal with dynamic and unforeseen obstacles typically employ a sensor-based map of the robot's local neighbourhood for planning purposes. By means of this data structure, geometric path planning algorithms and reactive steering methods (cf. subsection 1.1.3 for a state of the art overview) either calculate a complete trajectory that leads the vehicle to the designated target, or just determine a set of velocities to be applied in the next time frame. In this chapter a geometric path planning algorithm (CBCPP) is presented that applies cubic Bezier curves in order to model a collision free path to the goal. First proposed in [MFR06], the CBCPP considers a path progression that transfers the vehicle from its current pose $p_s = (x_s, y_s, \theta_s)$ to the selected target pose $p_g = (x_g, y_g, \theta_g)$, and thereby takes into account necessary veer-off movements in case of navigating narrow passages. After an introductory problem description and an overview on state of the art strategies, the CBCPP is characterized and evaluated against a highly sophisticated representative of the class of reactive steering methods, the *Nearness Diagram Navigation (NDN).*

4.1 Problem Description

The Bremen autonomous wheelchair Rolland is a good example for a non-circular shaped mobile robot that cannot independently control its kinematic degrees of freedom, i.e. its x- and y-position, and its heading θ. The top view in Fig. 4.1 shows Rolland with its differential drive whose two independently actuated wheels are located at each side of the main axle represented by the red-coloured y-axis in each pose. While the vehicle's rotational movements turn around a center of rotation that is located on the y-axis of its own coordinate system, a translational direction of movement is aligned with the red-coloured x-axis.

Considering now the exemplary task of autonomous navigation from $p_s = (x_s, y_s, \theta_s)$ to $p_g = (x_g, y_g, \theta_g)$ in Fig. 4.1, it becomes obvious that a simple movement from p_s to the center above the free passage env_2, followed by a 90° right-hand turn, and a final straight ahead movement to p_g would lead to a collision of the wheelchair's front-left edge with the door's outer frame. Thus any obstacle avoidance method has to look ahead for the upcoming turn manoeuvre and must initiate a preparatory veer-off movement that increases the distance to the obstacle env_1 in Fig. 4.1. While geometric path planning algorithms provide an elegant way of modelling a movement that becomes necessary in the future, reactive steering methods generally do not provide these explicit planning properties. The following subsection describes state of the art strategies that tackle the described problem.

4.1.1 State of the Art Strategies

Autonomous navigation approaches that have to deal with complex-shaped vehicles and non-holonomic constrained kinematics need to model these properties in order to achieve a precise navigational behaviour. One of the most prominent reactive steering methods, the *Nearness Diagram Navigation (NDN)* (cf. subsection 1.1.3), has initially been developed with the focus on circularly-shaped holonomic vehicles. Recent developments [35, 36, 33] have led to an extended framework that modifies the motion commands generated by the basic NDN with respect to the restrictions described in section 4.1. The underlying idea is to interpret the basic NDN commands as desired velocities for a holonomic and circularly-shaped robot. In a first stage, these velocities are modified by a dynamic model-based controller for differential drive robots [1]. The so-called *Motion-Generator* generates feasible velocities for non-holonomic constrained vehicles and passes them to a second module that takes into account the shape of the actual robot. The so-called *Shape Corrector* is based on the assumption that the virtual circularly-shaped vehicle taken into account so far can be inscribed into the physical robot's real shape. Necessary corrections for the final set of velocities are derived from an explicit collision test of a safety area that surrounds the robot's shape with nearby obstacles. Depending on the location of a potential collision w.r.t. the safety area, the approach selects an appropriate manoeuvre. It shall be noted that the design of the safety area is crucial for the achievable performance of the overall approach. An implementation on the smart wheelchair Rolland [22] spent major effort on the analysis of this point.

4.2 Geometric Path Planning Using Cubic Bezier Curves

The basic principle of the CBCPP algorithm is to continuously check sets of paths for a solution path in order to move the vehicle from its current pose to a given goal-pose. Therefore the algorithm evaluates a cost function that minimizes the length of the path to execute, while maximizing the distance to the obstacles along the selected path. A candidate path p that is to be taken into account must ideally fulfill the following four requirements:

1. Path p must connect $\vec{p_s} = (x_s, y_s)$ with $\vec{p_g} = (x_g, y_g)$ in order to define the positions of the wheelchair's reference point.

2. Path p must be smoothly aligned with θ_s in $\vec{p_S}$ and with θ_g in $\vec{p_g}$.

3. The underlying curve c of path p must provide a continuous curvature, i.e. c'' must be continuous.

4. Path p must be obstacle-free in the sense that a contour of the robot shifted tangentially along the path does not intersect with any obstacle point from a sensor-based evidence grid.

While requirements 1-3 define mandatory properties of the employed type of curve, requirement 4 solely describes the desired property of a path to be obstacle-free and is part of the algorithm's cost function. By contrast the disregarding of requirements 2 and 3 leads to a discontinuous evolution of the wheelchair's orientation along the path p and yields subsequent problems in the path controlling phase. In a first implementation that followed the work in [31], paths had been composed of straight line segments, clothoids, and circular arc segments. Albeit paths constructed this way are able to generate veer-off movements that are necessary to pass narrow door frames, several experiments showed problems in this situation. It turned out that the cost function's tradeoff between maximal clearance to the surrounding obstacles and the close goal-pose approach tended to prefer paths that led to misaligned configurations within the door frame. In addition the calculation of the candidate paths had been computationally expensive due to the huge search space, induced by the large set of the curve's determining parameters.

Starting from these insights, the second implementation draws on cubic Bezier curves. The following subsections show their appropriateness w.r.t. requirements 1-4, and prove their practical applicability within the sketched path planning framework.

4.2.1 Properties of Cubic Bezier Curves

Cubic Bezier curves are members of the family of parameterized algebraic curves that had initially been developed in order to interpolate plane curves in Computer Aided Design (CAD) processes (cf. [13] for an elementary introduction to algebraic curves). Specified by a sequence of four control points $\vec{cp}_0, ..., \vec{cp}_3$, a cubic Bezier curve as defined in (4.1) connects the two points $\vec{cp}_0$ and $\vec{cp}_3$ by a cubic polynomial and therewith complies with requirement 1 in section 4.2.

$$\begin{aligned} \vec{bc}(\lambda) &= \vec{a}\lambda^3 + \vec{b}\lambda^2 + \vec{c}\lambda + \vec{cp}_0, \ \lambda \in [0..1] \\ \vec{c} &= 3(\vec{cp}_1 - \vec{cp}_0) \\ \vec{b} &= 3(\vec{cp}_2 - \vec{cp}_1) - \vec{c} \\ \vec{a} &= \vec{cp}_3 - \vec{cp}_0 - \vec{b} - \vec{c} \end{aligned} \tag{4.1}$$

The selection of the free control points $\vec{cp}_1$ and $\vec{cp}_2$ determines the evolution of the Bezier curve by fixing the curve's direction in $\vec{cp}_0$ and $\vec{cp}_3$ as the direction of the vectors $\vec{cp}_1 - \vec{cp}_0$ and $\vec{cp}_3 - \vec{cp}_2$, respectively. Assuming an appropriate selection of $\vec{cp}_1$ and $\vec{cp}_2$, this property allows for the compliance with requirement 2 in section 4.2. The third mandatory requirement to mathematical curves representing candidate paths in the proposed path planning framework is the question for a continuous curvature. Cubic Bezier curves fulfill this requirement, with the exceptional case of two coinciding control points $\vec{cp}_1$ and $\vec{cp}_2$. This case results in a curve's cusp, i.e. a discontinuity in the second derivation of its parameter λ.

4.2.2 Search Space Definition

The sum of all cubic Bezier curves taken into account as a candidate path within a single cycle of computation makes up the search space of the core path planning algorithm. Considering the Bezier's properties defined in subsection 4.2.1, the underlying idea is to iterate over a suitable number of assignments for control points $\vec{cp}_1$ and $\vec{cp}_2$, while using the current position (x_s, y_s) of the vehicle as $\vec{cp}_0$, and the desired target position (x_g, y_g) as $\vec{cp}_3$. The definition of the search space in (4.2) now specifies $\vec{cp}_1$ and $\vec{cp}_2$ as points lying on vectors passing through $\vec{cp}_0$ and $\vec{cp}_3$, aligned to θ_s and θ_g respectively.

$$\begin{aligned} \vec{cp}_1(l_1) &= \vec{cp}_0 \pm l_1 \begin{pmatrix} \cos(\theta_s) \\ \sin(\theta_s) \end{pmatrix}, \; l_1max > l_1 > 0 \\ \vec{cp}_2(l_2) &= \vec{cp}_3 \mp l_2 \begin{pmatrix} \cos(\theta_g) \\ \sin(\theta_g) \end{pmatrix}, \; l_2max > l_2 > 0 \end{aligned} \tag{4.2}$$

Depending on the desired direction of motion, i.e. moving forwards or backwards, the upper or the lower algebraic sign of both sums in (4.2) is selected. The upper bounds l_1max and l_2max for the free parameters l_1 and l_2 are used to scale the computational payload of the path planning algorithm, since they finally determine the number of paths to be processed by the collision detection algorithm (cf. subsection 4.2.3). Typical values, which have been used for the experimental test runs described in subsection 4.2.4, are given in Table 4.2.4.

4.2.3 Additional Requirements

The calculation of the first point of a given cubic Bezier curve, where the shape of the wheelchair, tangentially located with its reference point, overlaps with a cell out of a sensor-based evidence grid, is done by a fast collision detection algorithm. Necessary optimizations of the basic approach are due to a high computational payload[1]. They include a cheap preluding collision check of a circle that inscribes the wheelchair's complex outer shape, potentially followed by a smart procedure that tests only a fraction of the real contour points. The underlying idea is to skip the test of a contour cell cc' that is located nearby a contour cell cc from which we know that its minimal distance to the next obstacle is greater than the distance $\mid cc' - cc \mid$. Both optimizations lead to a performance increase of about a factor 10 (cf. [MF07] for an in-depth analysis).

After choosing a certain Bezier curve as a solution for the search space defined in subsection 4.2.2, an upper translational velocity-bound $vb(\lambda)$ is computed for every $\vec{bc}(\lambda)$. It is determined by the minimal distance between the robot contour tangentially located at $\vec{bc}(\lambda)$ to any obstacle point, and the curvature $c(\lambda)$. The translational speed $v_d(\lambda)$ that considers $vb(\lambda)$ as well as the vehicle's translational and rotational acceleration limits, and the rotational velocity $w_d(\lambda)$ resulting from the derivation of the heading $\theta_d(\lambda)$, yield the final trajectory as in (4.3).

$$traj(\lambda) = (\vec{bc}(\lambda), \theta_d(\lambda), v_d(\lambda), w_d(\lambda)) \tag{4.3}$$

A path tracking controller has been implemented in order to compensate for physical inaccuracies between the wheelchair and its ideal differential drive kinematics, e.g. slippage and variable wheel diameter. The controller that has been adopted from the trajectory tracking controller proposed in [39] by Oriolo et al. uses dynamic feedback linearization to reduce the translational error between the vehicle's current configuration $(x(t), y(t), \theta(t))$ and the reference configuration $traj(\lambda)$. Note that $traj(\lambda)$ is the configuration lying on the Bezier curve that is closest to $(x(t), y(t))$.

[1]Considering a typical number of 4400 Bezier curves, each of which is discretized into 100 successive curve points, and a discretized wheelchair contour that comprises 132 contour points, an upper bound of about $O(58 \cdot 10^6)$ pixel-wise collision tests has to be performed per frame.

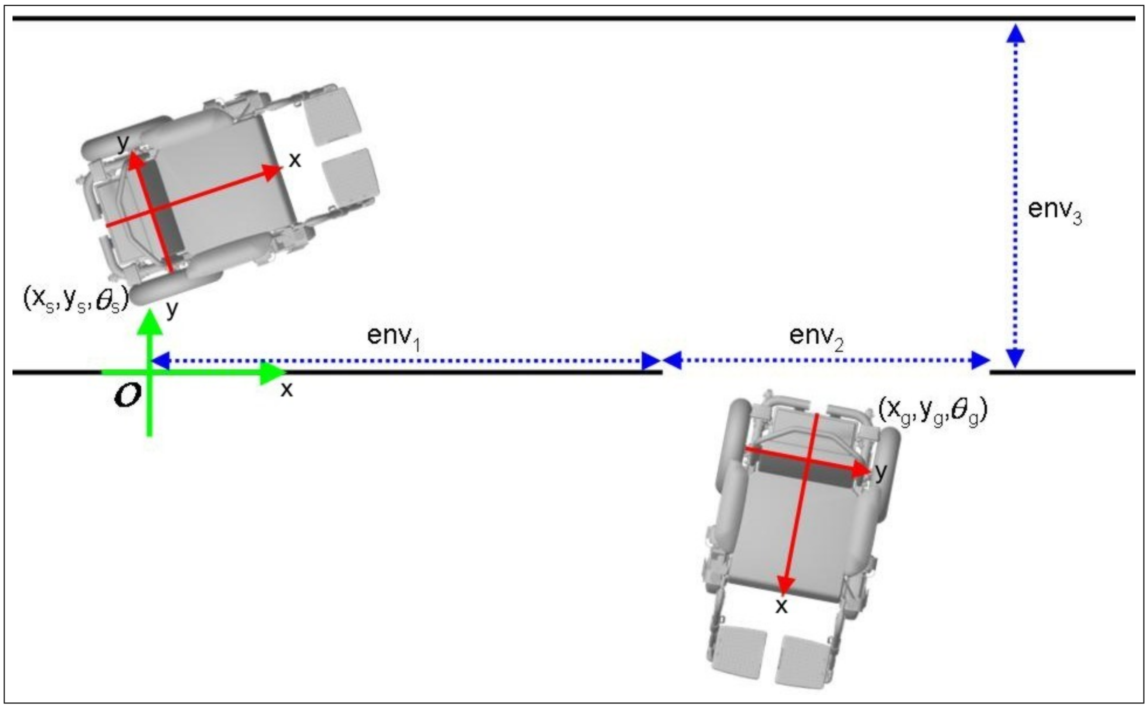


Figure 4.1: Experimental setup for the comparison of the *Geometric Path Planner Using Cubic Bezier Curves* and the *Nearness Diagram Navigation*. Both approaches had to prove their performance for the task of entering a narrow passage. Start and goal configurations, environmental and algorithmic parameters for the test runs are given in table 4.2.4.

$$\begin{aligned}\hat{v}(t) &= v_d(\lambda)\cos(\theta_d(\lambda) - \theta(t)) \\ \hat{w}(t) &= w_d(\lambda) + c_2\,\mathrm{sgn}(v_d(\lambda))((\vec{bc}.y(t) - y(t))\cos(\theta(t)) - \\ &\qquad (\vec{bc}.x(t) - x(t))\sin(\theta(t))) + c_3(\theta_d(\lambda) - \theta(t))\end{aligned} \tag{4.4}$$

Finally the wheelchair's actual velocities $(v(t), w(t))$ are fed into a PID velocity controller (4.5) in order to correct for the desired velocities $(\hat{v}(t), \hat{w}(t))$. The resulting velocities (v_{cmd}, w_{cmd}) are passed to the motor controllers.

$$\begin{aligned}v_{cmd}(t) &= -c_{Pv}(v(t) - \hat{v}(t)) - c_{Dv}(\dot{v}(t) - \dot{\hat{v}}(t)) - c_{Iv}(\int\limits_{t'=0}^{t} v(t') - \int\limits_{t'=0}^{t} \hat{v}(t')) \\ w_{cmd}(t) &= -c_{Pw}(w(t) - \hat{w}(t)) - c_{Dw}(\dot{w}(t) - \dot{\hat{w}}(t)) - c_{Iw}(\int\limits_{t'=0}^{t} w(t') - \int\limits_{t'=0}^{t} \hat{w}(t'))\end{aligned} \tag{4.5}$$

4.2.4 Experimental Evaluation

Apart from the application in different test scenarios of the proposed user interfaces (cf. subsections 3.1.1, 3.1.2, 3.2.2), the CBCPP has been compared with the NDN in an experimental evaluation, and furthermore proven its robustness during a long-term test run. Both

	GoalPose			**Environment**			**CBCPP**			**NDN**	
#	x_g (cm)	y_g (cm)	θ_g (deg)	env_1 (cm)	env_2 (cm)	env_3 (cm)	l_1max (cm)	l_2max (cm)	δ (cm)	1 (cm)	2 (cm)
1	250	-100	-90	200	100	150	450	450	5	55	7.5
2	275	-100	-90	200	150	150	450	450	7.5	55	7.5
3	250	-100	-90	200	100	230	450	450	5	55	7.5
4	250	-100	-90	200	150	230	450	450	7.5	55	7.5

Table 4.1: Variations of algorithmic and environmental parameters for the experimental comparison between the *Geometric Path Planer Using Cubic Bezier Curves (CBCPP)* and the *Nearness Diagram Navigation (NDN).* For an illustration of the experimental setup confer Fig. 4.1. The start-pose has been selected as $p_s = (0, 75, 0)$.

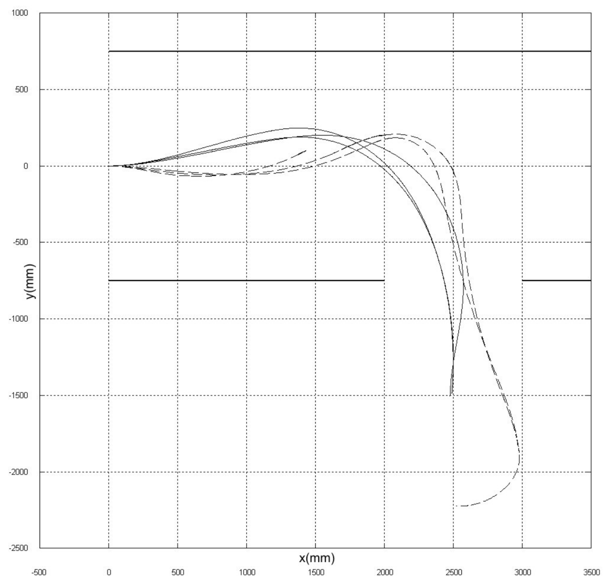
experiments demonstrate the applicability of the proposed path planning approach w.r.t. precise close range navigation and reliable long-term performance, respectively.

Comparison to Nearness Diagram Navigation

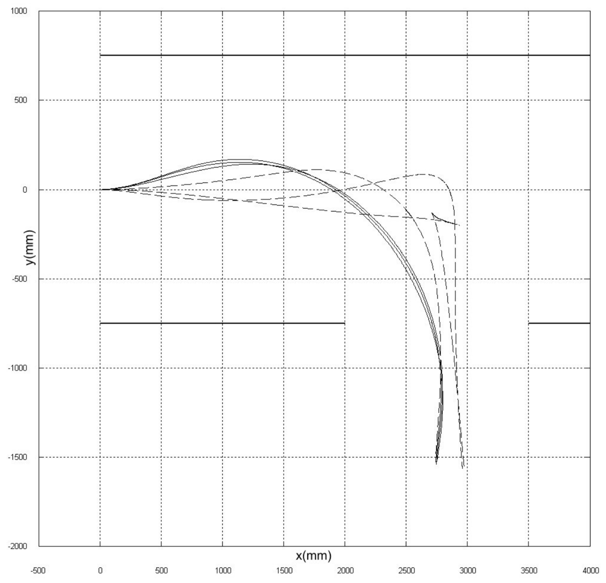
The purpose of the comparative test between the CBCPP and the NDN is to evaluate both approaches in a restricted close range navigation scenario. Fig. 4.1 illustrates the experimental setup that simulates the situation of passing through a narrow door from within a corridor. Several environmental parameters like the widths of the door (env_2) and the the corridor (env_3) have been tested (cf. Table 4.2.4). Illustrated in Fig. 4.2, the odometry plots of the paths executed by the CBCPP (solid lines) and the NDN (dashed lines) give a good performance indication of both approaches. Characteristic for the NDN results is the unpredictable heading of the wheelchair in the target pose, resulting from the algorithm's negligence of this parameter. By contrast, the CBCPP reaches the desired target heading with sufficient precision in all but the fourth test run (cf. Fig. 4.2(d)). A further difference can be seen in that the NDN always tends to initiate the necessary veer-off movement at a later point in time than the CBCPP. This behaviour results in necessary chunking manoeuvres (cf. Fig. 4.2(b)), or badly aligned door frame transitions. It shall be noted that experimental test run 1 (cf. Fig. 4.2(a)) and 3 (cf. Fig. 4.2(c)) required a fine tuning of the CBCPP's search space increment δ, since higher values led to fewer solution paths passing through the smallest point of the passage.

Long-Term Test

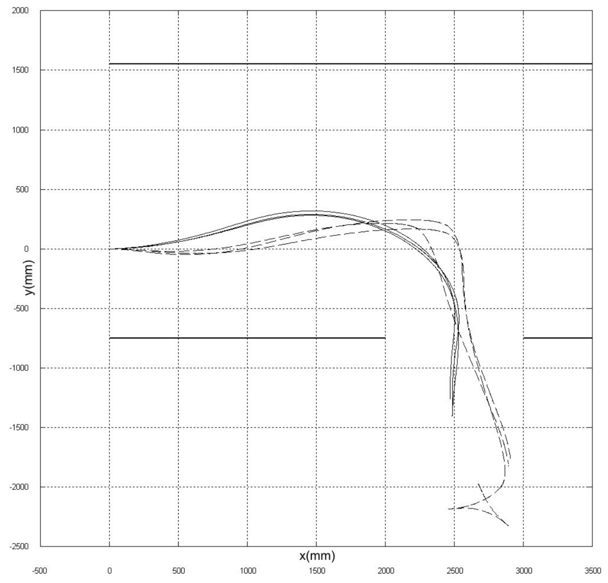
In order to prove the implementation of the CBCPP for stability and robustness against frequently appearing dynamic obstacles, the proposed approach has been checked in a long-term test run that involved the autonomous execution of a 1110m long trajectory in a populated office building. Fig. 4.3 illustrates the executed path as estimated by the applied Monte Carlo Localization Approach [44]. The course that included three closed loops has been computed by a higher level path planner that employed the route graph representation of the environment, and permanently forwarded local target poses to the CBCPP.



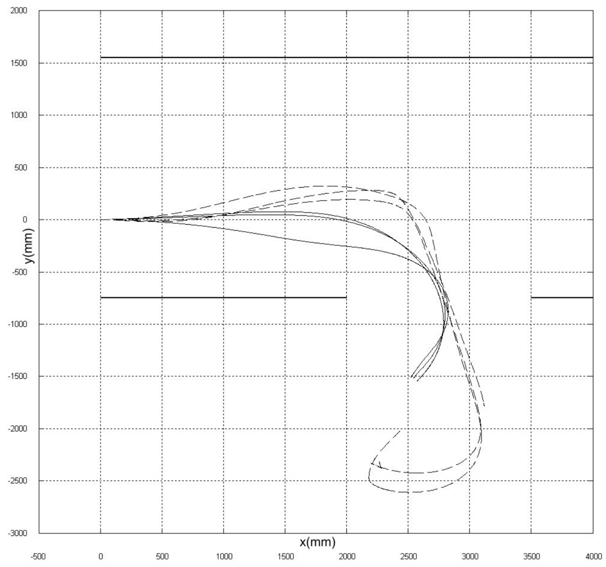
(a) Experimental test run 1.



(b) Experimental test run 2.



(c) Experimental test run 3.



(d) Experimental test run 4.

Figure 4.2: Odometry plots of an experimental comparison between the *Geometric Path Planner Using Cubic Bezier Curves* (solid lines) and the *Nearness Diagram Navigation* (dashed lines). The numbering of the test runs corresponds to the ordering in Table 4.2.4.

4.3 Contribution of the Corresponding Publications

The contribution of this thesis's publications that describe the design and the implementation of the *Cubic Bezier Curve Path Planner (CBCPP)* (cf. [MFR06, MRF07]), is primarily given by an extension of the basic principles of the *Dynamic Window Approach (DWA)* (cf. Fox et al. [11]). In contrast to the DWA that utilizes circular arcs of varying radius, the CBCPP applies cubic Bezier curves in order to model the vehicle's movement. This type of curve not

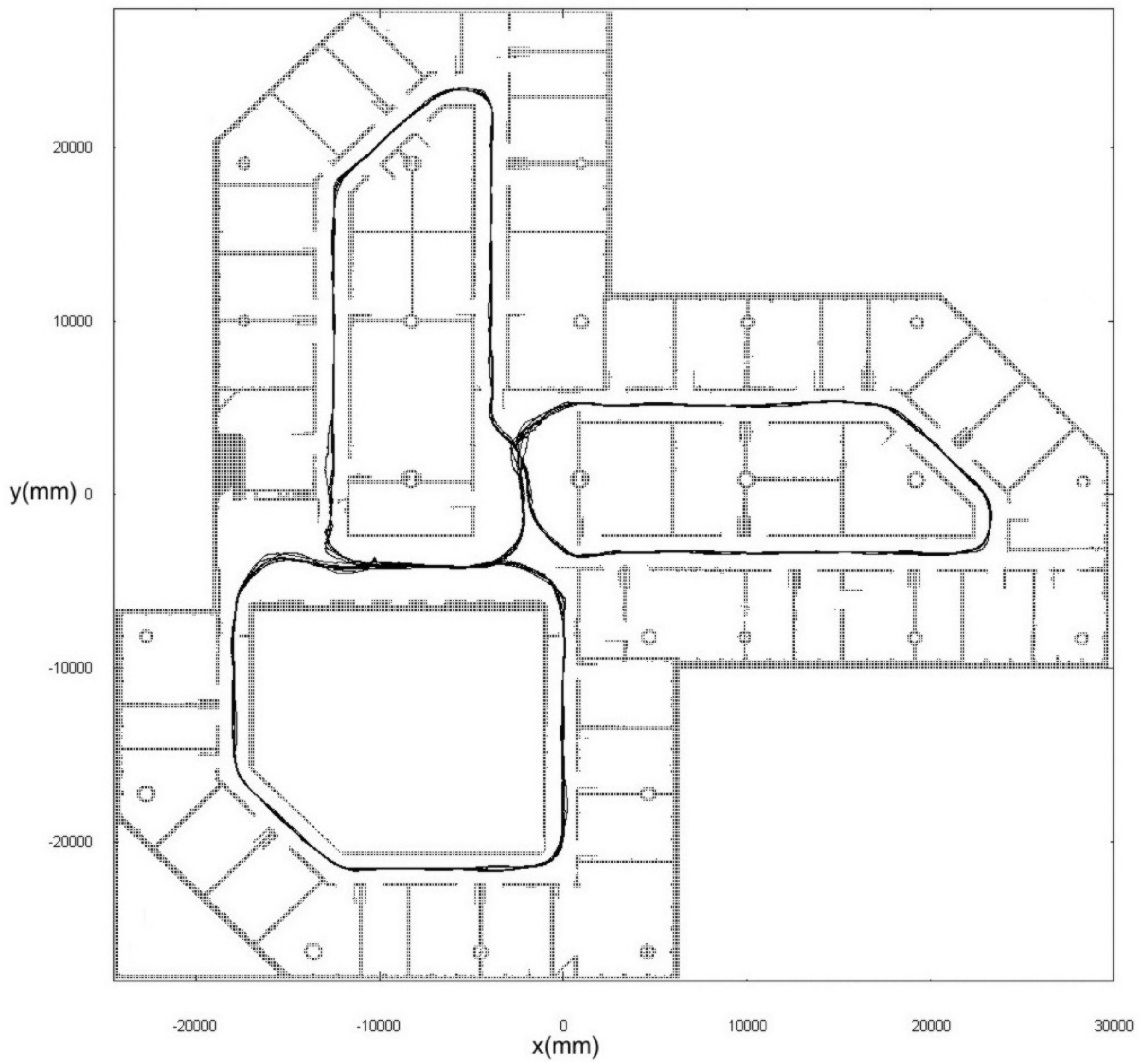


Figure 4.3: Trajectory of a long-term test run as estimated by the applied Monte Carlo Localizer [44]. The autonomously executed path has an overall length of about $1110m$ and was executed in circa $38min$. Note that the experiment has been conducted within a populated environment, i.e. it included several safe-stop and obstacle avoidance manoeuvres due to dynamic obstacles.

only provides the explicit modeling of necessary veer-off movements in certain navigational situations, but also assures a continuous curvature along the computed trajectory, given the search space definition in subsection 4.2.2. Furthermore, the CBCPP improves DWA's collision checking. While the standard DWA assumes a circularly shaped robot and therefore gets by doing simple and fast geometric computation, the approach developed in this thesis's publications had to be adapted because of the wheelchair's irregular shape and its center of rotation that is located in the rear. Thus the implemented algorithm employs a bitmap describing the outer shape of the wheelchair that is matched against the local obstacle map in order to determine when a collision will occur.

List of Publications by the Author

[GAM+07] B. Graimann, B. Allison, C. Mandel, T. Lueth, D. Valbuena, and A. Gräser. *Robust Intelligent Systems*, chapter Non-invasive Brain-Computer Interfaces for Semi-Autonomous Assistive Devices. Springer Verlag, 2007. submitted.

[KBFL+05] B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, and R. Ross. Specification of an Ontology for Route Graphs. In C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, and T. Barkowsky, editors, *Spatial Cognition IV*, volume 3343 of *Lecture Notes in Artificial Intelligence*, pages 390–412. Springer-Verlag, 2005.

[MF07] C. Mandel and U. Frese. Comparison of Wheelchair User Interfaces for the Paralysed: Head-Joystick vs. Verbal Path Selection from an Offered Route-Set. In W. Burgard and H.M. Gross, editors, *Proceedings of the 3rd European Conference on Mobile Robots (ECMR 2007)*, pages 217–222. University of Freiburg, 2007. [Online]. Available: http://ecmr07.informatik.uni-freiburg.de/accepted_p.html.

[MFR06] C. Mandel, U. Frese, and T. Röfer. Robot Navigation based on the Mapping of Coarse Qualitative Route Descriptions to Route Graphs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, pages 205–210. IEEE Xplore, 2006.

[MFR07] C. Mandel, U. Frese, and T. Röfer. Design Improvements for Proportional Control of Autonomous Wheelchairs Via 3DOF Orientation Tracker. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, Lecture Notes in Computer Science, pages 1052–1059. Springer-Verlag, 2007.

[MHV05] C. Mandel, K. Huebner, and T. Vierhuff. Towards an Autonomous Wheelchair: Cognitive Aspects in Service Robotics. In U. Nehmzow, C. Melhuish, and M. Witkowski, editors, *Proceedings of Towards Autonomous Robotic Systems (TAROS 2005)*, pages 165–172. Imperial College London, 2005. ISBN 0-905247-03-5.

[MRF07] C. Mandel, T. Röfer, and U. Frese. Applying a 3DOF Orientation Tracker as a Human-Robot Interface for Autonomous Wheelchairs. In *Proceedings of the IEEE 10th International Conference on Rehabilitation Robotics (ICORR 2007)*. IEEE Xplore, 2007. to appear.

[SMR07] Hui Shi, Christian Mandel, and Robert J. Ross. Interpreting Route Instructions as Qualitative Spatial Actions. In T. Barkovsky, M. Knauff, G. Ligozat, and D.R. Montello, editors, *Spatial Cognition V*, volume 4387 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2007. to appear.

Bibliography

[1] J.R. Asensio and L. Montano. A Kinematic and Dynamic Model-Based Motion Controller for Mobile Robots. In E.F. Camacho, L. Basanez, and J.A. de la Puente, editors, *Proceedings of the 15th Triennial World Congress of the International Federation of Automatic Control (IFAC)*, 2002.

[2] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[3] M. Bailey, A. Chanler, B. Maxwell, M. Micire, K. Tsui, and H. Yanco. Development of vision-based navigation for a robotic wheelchair. In *Proceedings of the 10th IEEE Intl. Conf. on Rehabilitation Robotics (ICORR)*, 2007.

[4] P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers. Exploiting local perceptual models for topological map-building. In *Proceedings of IJCAI-2003 Workshop on Reasoning with Uncertainty in Robotics*, pages 15–22, 2003.

[5] J Borenstein and Y Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, - 1989.

[6] M. Bureau, J.M. Azkoitia, G. Ezmendi, I. Manterola, and H. Zabaleta. Non-invasive, wireless and universal interface for the control of peripheral devices by means of head movements. In *Proceedings of the 10th IEEE Intl. Conf. on Rehabilitation Robotics (ICORR)*, 2007.

[7] U. Canzler and K.-F. Kraiss. Person-adaptive facial feature analysis for an advanced wheelchair user-interface. In *Proceedings of the IEEE Intl. Conf. on Mechatronics and Robotics*, pages 871–876, 2004.

[8] Y.-L. Chen, S.-C. Chen, W.-L. Chen, and J.-F. Lin. A head orientated wheelchair for people with disabilities. *Disability and Rehabilitation*, 25(6):249–253, 2003.

[9] D. Ding and R.A. Cooper. Electric-Powered Wheelchairs. *IEEE Control Systems Magazine*, 25(2):22–34, 2005.

[10] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie-Mellon University, 1989.

[11] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *Robotics & Automation Magazine, IEEE*, 4(1):23–33, 1997.

[12] O. Friman, I. Volosyak, and A. Gräser. Multiple channel detection of steady-state visual evoked potentials for brain-computer interfaces. *IEEE Transactions on Biomedical Engineering*, 54(4):742–750, April 2007.

[13] C.G. Gibson. *Elementary Geometry of Algebraic Curves.* Cambridge University Press, 1998.

[14] J. Gips. On Building Intelligence into EagleEyes. In *Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces and Natural Language Processing*, pages 50–58. Springer-Verlag, 1998.

[15] K. Hübner. *Symmetriesignaturen für Bildbasierte Anwendungen in der Robotik.* PhD thesis, University of Bremen, January 2007.

[16] Shi Hui and Thora Tenbrink. Telling Rolland where to go: HRI dialogues on route navigation. In *Proc. Workshop on Spatial Language and Dialogue (5th Workshop on Language and Space)*, 2005.

[17] I1-[OntoSpace]. Human-Robot Interaction, 2004.

[18] G. Ippoliti, L. Jetto, and S. Longhi. Localization of Mobile Robots: Development and Comparative Evaluation of Algorithms Based on Odometric and Inertial Sensors. *Journal of Robotic Systems*, 22(12):725–735, 2005.

[19] David L. Jaffe. An ultrasonic head position interface for wheelchair control. *Journal of Medical Systems*, 6(4):337–342, 1982.

[20] H.J. Kim and Ryu D.H. Smart Wheelchair Based on Ultrasonic Positioning System. In *Computers Helping People With Special Needs*, volume 4061/2006, pages 1014–1020. Springer Berlin/Heidelberg, 2006.

[21] J. Kim. *A Framework for Roadmap-Based Navigation and Sector-Based Localization of Mobile Robots.* PhD thesis, Texas A&M University, August 2004.

[22] P. Kraetsch. Entwicklung einer reaktiven steuerung für mobile roboter auf basis der nearness-diagram-methode für navigation in innenräumen. Master's thesis, Universität Bremen, 2007. (submitted).

[23] B. Krieg-Brückner and H. Shi. Orientation calculi and Route Graphs: Towards semantic representations for route descriptions. In *Proc. International Conference GIScience*, pages 234–250, 2006.

[24] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000.

[25] B. Kuipers, J. Modayil, P. Beeson, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4845–4851, 2004.

[26] Y. Kusuda. A navigation system for wheelchairs. *Industrial Robot: An International Journal*, 30(2):162–165, 2003.

[27] T. Kyriacou, G. Bugmann, and S. Lauria. Vision-Based Urban Navigation Procedures for Verbally Instructed Robots. In *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1326–1331, 2002.

[28] A. Lankenau and T. Röfer. Smart wheelchairs - state of the art in an emerging market. *Künstliche Intelligenz. Schwerpunkt Autonome Mobile Systeme*, 4:37–39, 2000.

[29] Jean-Claude Latombe. *Robot Motion Planning*. Cluwer Academic Press, 1991.

[30] S.P. Levine, D.A. Bell, L.A. Jaros, R.C. Simpson, Y. Koren, and Borenstein J. The navchair assistive wheelchair navigation system. *IEEE Transactions on Rehabilitation Engineering*, 7(4):443–451, December 1999.

[31] Christian Mandel. Trajektorienplanung und Trajektorienfolgeregelung im Konfigurationsraum nich-holonomer Fahrzeuge. Master's thesis, Universität Bremen, 2002.

[32] David P. Miller. Assistive robotics: An overview. In *Assistive Technology and Artificial Intelligence*, pages 126–136, 1998.

[33] J. Minguez. Integration of planning and reactive obstacle avoidance in autonomous sensor-based navigation. In *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2486–2492, 2005.

[34] J. Minguez and L. Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1), 02 2004.

[35] J. Minguez and Luis Montano. Robot Navigation In Very Complex, Dense, and Cluttered Indoor/Outdoor Environments. In *Proceedings of the International Federation of Automatic Control (IFAC)*, 2002.

[36] J. Minguez and Luis Montano. The ego-kinodynamic space (ekds): Shape, kinematics and dynamics of the vehicle. In *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 637–643, 2003.

[37] J. Minguez, J. Osuna, and L. Montano. A divide and conquer strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2004*, pages 3855–3862, 2004.

[38] R. Müller, A. Lankenau, A. Musto, K. Stein, and A. Eisenkolb. Coarse qualitative descriptions in robot navigation. In *Spatial Cognition II, Lecture Notes in Artificial Intelligence*, pages 265–276, 2000.

[39] G. Oriolo, A. Luca, and M. Vendittelli. WMR Control via Dynamic Feedback Linearization: Design, Implementation and Experimental Validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, November 2002.

[40] S.P. Parikh, V.Jr. Grassi, V. Kumar, and J. Okamoto. Incorporating User Inputs in Motion Planning for a Smart Wheelchair. In *Proceedings of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2043–2048, 2004.

[41] G. Pires, U. Nunes, and A.T. de Almeida. Robchair - A Semi-Autonomous Wheelchair for Disabled Persons. In *Proceedings of the 3rd. IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, pages 648–652, 1998.

[42] E. Prassler, J. Scholz, and P. Fiorini. Navigating a Robotic Wheelchair in a Railway Station during Rush Hour. *International Journal of Robotics Research*, 18(7):711–727, 1999.

[43] B. Rebsamen, E. Burdet, C. Guan, H. Zhang, C. Leong Teo, M. Ang, and C. Laugier. Controlling a wheelchair using a BCI with low information transfer rate. In *Proceedings of the 10th IEEE Intl. Conf. on Rehabilitation Robotics (ICORR)*, 2007.

[44] T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 856–861, 2003.

[45] T. Röfer. Using Histogram Correlation to Create Consistent Laser Scan Maps. In *Proceedings of the IEEE International Conference on Robotics Systems (IROS-2002)*, pages 625–630. EPFL; Lausanne, Switzerland, 2002.

[46] R. C. Simpson. Smart wheelchairs: A literature review. *Journal of Rehabilitation Research & Development*, 42(4):423–436, 2005.

[47] R.C. Simpson and S.P. Levine. Voice control of a powered wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(2):122–125, June 2002.

[48] R.C. Simpson, E. LoPresti, S. Hayashi, and I. Nourbakhsh. The Smart Wheelchair Component System. *Journal of Rehabilitation Research & Development*, 41(38):429–442, May/June 2004.

[49] A. Tayfun, Y. Cagri, and B. Billur. Statistical Pattern Recognition Techniques for Target Differentiation using Infrared Sensor. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 468–473, 2006.

[50] S. Thrun. Towards a framework for human-robot interaction. *Human Computer Interaction*, 19:9–24, 2003.

[51] G. Vanacker, J.d.R. Millan, E. Lew, P.W. Ferrez, F.G. Moles, J. Philips, H. Van Brussel, and M. Nuttin. Context-Based Filtering for Assisted Brain-Actuated Wheelchair Driving. *Computational Intelligence and Neuroscience*, 2007, 2007.

[52] P. Veelaert and W. Bogaerts. Ultrasonic Potential Field Sensor for Obstacle Avoidance. *IEEE Transactions on Robotics and Automation*, 15(4):774–779, 10 1999.

[53] S. Werner, B. Krieg-Brückner, and T. Herrmann. Modelling navigational knowledge by route graphs. In *Spatial Cognition II, Lecture Notes in Artificial Intelligence*, pages 295–316, 2000.

[54] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, June 2002.

[55] H.A. Yanco. Wheelesley, a robotic wheelchair system: Indoor navigation and user interface. In *Lecture Notes in AI: Assistive Technology and Artificial Intelligence*, 1998.

[56] C. Ye and J. Borenstein. Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation. In *Proceedings of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2512–2518, 2002.

Appendix A

Accumulated Publications

Towards an Autonomous Wheelchair: Cognitive Aspects in Service Robotics

Christian Mandel, Kai Huebner, Tilman Vierhuff
Department of Computer Science
University of Bremen
P.O.Box 330 440
Germany - 28334 Bremen
{cman, khuebner, tv}@informatik.uni-bremen.de

Abstract

As the origin of this work comes from the areas of *Artificial Intelligence* and *Cognitive Robotics* we basically describe the sensorimotor capabilities of an autonomous robot i. e. an automated wheelchair. Beginning with the technical aspects of sensing the environment and extracting its essential features, we relate these techniques to the terminology of *Cognitive Science.* We continue this practice in describing the way the robot represents its local neighbourhood as well as the global workspace. By describing algorithms for global localisation, navigation, and a natural language interface to the overall system, we conclude the description of our state of affairs in the development of a fully autonomous wheelchair.

1. Introduction

Today's robots have found their way from sealed working stations in factories to populated places such as museum halls, railway stations and hospitals. Here they perform useful tasks for their operators, either side by side or in cooperation with human beings. The gained benefit comes along with the necessity to design the system in a way that it is able to respond to a list of complex situations. This includes at least the variety to be situated in an unknown, unstructured and dynamic environment, to handle failure of single components of the robot or to negotiate inconsistent objectives. The resulting demand on interactivity and adaptability asks for:

1. Sensors which perceive the environment and the own state.
2. Modelling of the sensorial information in order to represent the ascertainable environment as well as the state of the robot.
3. Integration of significant local information to a global and consistent description.
4. Methods to exploit the gathered knowledge to accomplish tasks of different kinds.

In this paper we give an overview on how the *Bremen Autonomous Wheelchair Rolland III* (cf. figure 1), standing as an example for autonomous service robots, approaches the above stated demands. Section 2. starts with a survey of significant representatives in the field of service robotics, followed by a brief description of established system designs. In section 3. we describe the technical aspects of Rolland III's sensorial components, which allow us to talk about the basis of perceptual processes. Section 4. addresses the basic data structures which hold the information immediately received by the sensors as well as more abstracted ones collecting the results of first filtering and integration processes. These two classes of information are characterised in context of our experimental platform's tasks. In the following section 5. we introduce the *Route Graph* notion i. e. a multi-layered graph structure reflecting the hierarchy of different types of information. We identify this data structure as *Long-Term Memory* since it is used to store global knowledge. Then, in section 6. we present algorithms which address problems like global localisation, local and global navigation as well as query mechanisms, all of them exploiting the acquired spatial knowledge. After section 7. addresses basic safety issues and section 8. describes our natural language based interface to the robot, we conclude in section 9.with still unaddressed problems and future work.

2. Related Work

Service robots which act in populated environments are successful since they use a variety of sensors and sophisticated system designs based on statistical methods. Burgard and colleagues describe in (Burgard et al., 1999) the autonomous, interactive tour-guide robot *RHINO* which was successfully deployed in a densely populated museum. By using input from a stereo camera, sonar, laser range finder and infrared sensors, the robot solved

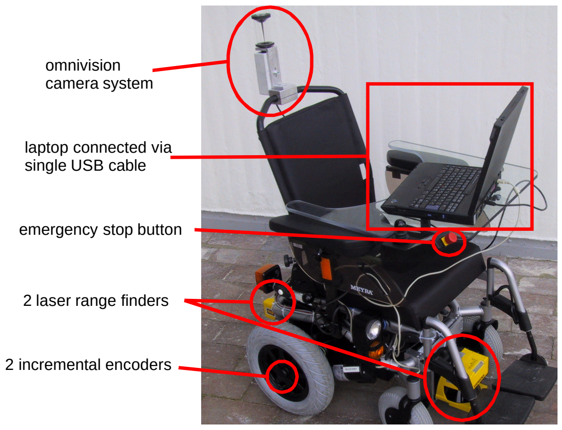


Figure 1: Rolland III with its sensory equipment.

fundamental tasks like self-localisation, mapping and autonomous navigation. A further example of successful employed service robots comes in form of a system able of autonomously exploring and mapping abandoned mines (Thrun et al., 2005).

The above examples focus on exploiting metric occupancy grids to model the environment. Although this approach is very successful in simultaneously localisation and mapping tasks, it does not scale very well because it needs a lot of memory resources. Hence the alternative idea, to use topological descriptions of the environment for robot localisation and mapping, is appealing because of its computational efficiency. They use graph structured representations, where locally distinctive places such as door frames, corners or midpoints of free space are connected by possible passages like hallways. In contrast to metrical approaches like occupancy grids they do not ask for a global frame of reference but lack their overall precision. The *Spatial Semantic Hierarchy* (Kuipers, 2000) describes the topological features of large-scale space in terms of places, paths and regions which are interrelated by connectivity, order, and containment. This *topological level* builds on the so-called *causal level* which consists of causal schemas connecting two distinctive states of the system by a sequence of control laws, i. e. moving the robot from the first distinctive state to the second. The causal level itself resides on the *control level* which defines the isolated distinctive states by the applicability of hill-climbing control laws. Since its origin, the SSH has been extended to fully support topological mapping. By exploiting local perceptual models which are grounded on sensor-based occupancy grids, pre-existing obstacle maps, and computer vision, Beeson and colleagues facilitate the deduction of a consistent global map (Beeson et al., 2003). Current implementations derive the necessary features like gateways and hallways by analysing the so-called medial axis representation.

A major drawback of using medial axis representation is their unstable behaviour under dynamic changes in the environment. Changing topological features like open vs. closed doors, signatures of moving obstacles, as well as problems caused by sensorial limitations are addressed by the use of higher order Voronoi diagrams (Fiegert and De Graeve, 2003). Instead of generating a medial axis by calculating the distance to the closest obstacle points, this approach calculates the distance to the closest obstacles. This calls for preceding segmentation. The resulting higher order medial axis allows the filtering of dynamic objects. Fiegert and colleagues complete their approach by matching the generating points of local obstacles with the distance grid of the global map so far constructed, yielding a reasonable good probability measure for their applied *Monte Carlo Localisation* algorithm.

3. Physical Stimulus *or* What Does the Robot Perceive?

In contrast to the predecessor of Rolland III which was equipped with a ring of sonar sensors and later on a single backward facing laser range finder (Lankenau and Röfer, 2001), our current experimental platform is equipped with two laser scanners mounted at ground level, which allow for scanning beneath the feet of a human operator. Each of the two Siemens LS4 laser scanner has an opening angle of 190^o. Within this field of view the device emits up to 528 rays of light in order to measure the time each beam travelled after having been reflected by an obstacle. From this information the internal electronics calculates a list of distances and angles and sends it to the processing computer. The device operates with a maximal lateral resolution of 0.36^o and a radial resolution of 5 mm within a scope of 50 m at a constant update rate of 25 scans/s.

An additional sensor device for observing the robot's environment is presented by an omni directional vision system mounted on the top of Rolland III (cf. figure 1). Like most catadioptric systems (Benosmann and Kang, 2001), the one applied here comprises a firewire colour camera facing upwards to a hyperbolic mirror above. Omni directional images are restricted in image resolution, but offer the main advantage of giving a complete 360^o visual perception of the surrounding in each time step. Thus, they are widely applied and researched in robot vision tasks, e. g. localisation and navigation. Methods for unwarping distorted omni directional views into both panoramic and perspective views are well elaborated to offer user-friendly visual feedback.

Finally, the system provides two Lenord+Bauer Gel248 incremental encoders which measure the rotational velocity of the two independently actuated wheels.

Please note that the provided information needed for dead reckoning is highly imprecise due to variable wheel diameter and slippage.

4. Working Memory *or* From Scans to Local Maps

In the following section we describe intermediate data structures which are filled by the pre-processed flow of data coming from the sensors, hence carrying little or no semantics.

4.1 Spatial Working Memory

As described in section 3., the basic perceptual components of the robot collect raw data describing distances to nearby obstacles, and velocities of the two actuated wheels. A first formalisation of this data comes with the structured description of the obstacle points measured by a laser scan at time t. This list of metrical points is expressed in the robot's frame of reference:

$$ScanPoints_t = \{(x_{i,t}, y_{i,t}) | i = 1 \ldots numOfScanPoints\} \tag{1}$$

Though this information plotted into a coordinate system is appropriate to get an impression of the environment's shape (cf. endpoints of rays in figure 2), it is too detailed to be processed in realtime for the purpose of feature detection. Therefore, each $ScanPoints_t$ is segmented into a list of line segments which represent straight obstacle lines. The list of scan segments of a single scan at time t is thereby defined as the list of vectors connecting two scan points, with the scan points in between no more distant to the vector to be defined than a given ϵ. Hence a single $ScanSegment_t$ is defined by:

$$v_{t,i,j} ::= (\overrightarrow{ScanPoints_{t,i}, ScanPoints_{t,j}}) | \; dist(v_{t,i,j}, ScanPoints_{t,k \forall i<k<j}) < \epsilon \tag{2}$$

The segmented lines with square endpoints in figure 2 show that the resulting amount of data is quite handy. This allows us to detect a subset of our addressed features with reasonable amount of processing time. In contrast to the above data structures which are build from measurements gathered in a single cycle of computation, we also use local occupancy grids which are acquired by integrating consecutive objects of type $ScanPoints_t$ over the actual odometry pose. The odometry pose $[x, y, \theta]$ describing the robot's position and heading in the plane is thereby defined by the kinematic equation for differential driven vehicles:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos\theta(t) & 0 \\ sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2}(v_r(t) + v_l(t)) \\ \frac{v_r(t) - v_l(t)}{L} \end{bmatrix} \tag{3}$$

In (3), L denotes the width of the main axle, and $v_r(t)$, $v_l(t)$ the rotational speed of the right resp. the left actuated wheel. Figure 3 illustrates an exemplary $EvidenceGrid_t$. In every frame t new measurements of type $ScanPoints_t$ are inserted into the grid by updating the corresponding cell's probability of being occupied. In our example the colour of the grid cells range from white, designating free space, to black, marking occupied space. Note that the map only covers the nearby environment because the utilised technique of integrating the odometry pose would lead to inconsistent global maps. The medial axis representation of the environment completes the list of employed data structures. Derived from the $EvidenceGrid_t$, the $DistanceGrid_t$ is also organised as a rectangular array of grid cells, now containing the distances to the closest obstacle points. The necessary information for every single grid element is computed by a fast sweep line algorithm. The graph like components in figure 3 show a derivate of the $DistanceGrid_t$. This so-called $VoronoiDiagram_t$ is a reduction of the $DistanceGrid_t$, which describes the space with maximal clearance to the surrounding obstacles by a graph like structure. While we show in section 5. that this kind of representation is also suitable for describing the global workspace, we describe in section 6. how we exploit the local $ScanSegments_t$ and $VoronoiDiagram_t$ in order to detect open doors in the robot's workspace i. e. a structured office environment.

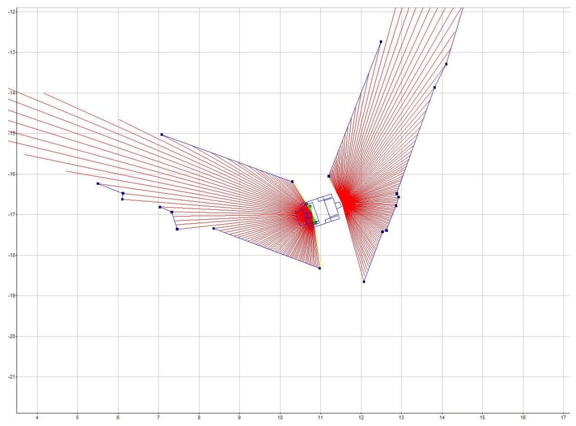


Figure 2: The robot's local frame of reference including $ScanPoints_t$ and $ScanSegments_t$.

4.2 Visual Working Memory

In each cycle, the system retrieves an image taken from the on-board omni directional camera system. The visual percept accordingly is represented as an array of picture elements (pixels). The omni directional camera solution offers several types of view starting from the omni directional round view. For both an intuitive representation for the human viewer and the subsequent task of feature extraction, also panoramic and perspec-

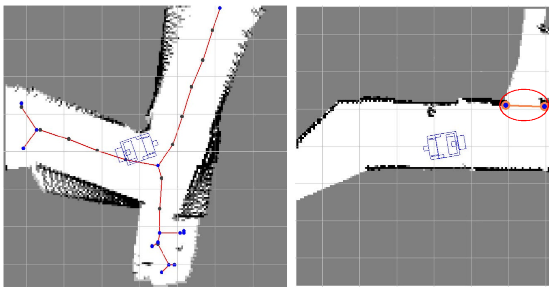


Figure 3: Left: Local $EvidenceGrid_t$ and $VoronoiDiagram_t$; Right: Locally detected door by analysing $ScanSegments_t$ and $VoronoiDiagram_t$.

tive views can be subsequently generated (cf. figure 4).

Visual information is one of the most capable sensor information types available. The data, however, is high-dimensional and pre-processing steps are needed to extract significant features. Vlassis et al. (2001) describe the most important properties of image features as robust, invariant to rotation, occlusion and lighting conditions, and capable of compressing the image data as much as possible while retaining pertinent information.

Approaches to feature extraction can be divided into different fields, where a-priori choice of features is the most intuitive technique, e. g. the analysis of edges, corners or colour in the image data. As the number of different methods for this task is large, the number of different types of extracted features is also. Image representations are made as a summary of points of interest, regions of interest, histograms, etc., but following the aim of compressing the visual information to a most significant part. Based on this "core" of information, it is much more efficient to describe, compare or recognise objects and scenes observed by visual sensors.

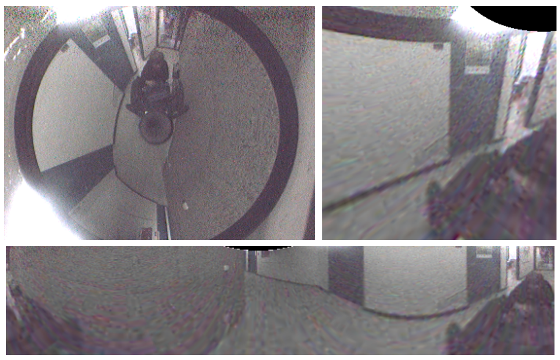


Figure 4: Omni directional camera view (top left); Unwarped perspective view (top right) and panoramic view (bottom). Corresponding to the situation presented in Fig. 3 right.

5. Long-Term Memory *or* What Resides Outside the System's View

The local information, which is used by the robot for purposes of feature detection and immediate reaction like local navigation, has to be integrated into a global data structure in order to solve higher level tasks. Such tasks are global localisation, global route planning or queries to the overall database like *"Where is the secretary's office?"*. In our work we apply a graph structured representation i. e. the *Route Graph* to do this job.

In the route graph taxonomy (Krieg-Brückner et al., 2004, Werner et al., 2000) a *Route Graph* representing the whole workspace of the robot consists of several *Route Graph Layers* each of a different *Kind*. At several *Places* which represent distinguishable robot poses, the route graph layers are connected via *Transitions*, depicting a correspondence between places inside different types of map representation. On the other hand, places can be connected by *Route Segments* in a single route graph layer, stating that the robot can go from one place to the other. Concatenation of these route segments results in *Routes*, standing for a complex navigational description. Beside our implementation of the route graph concept, we specified a XML-based exchange format which allows for a structured communication of an instantiated route graph.

Our ongoing work is mainly focused on the topological layer of the route graph. It describes a network of places and route segments with maximal clearance to the surrounding obstacles, based on Voronoi diagrams. The framework implemented so far is able to compute this description either from local sensor-based grid maps (cf. section 4.) or from a pre-existing global grid map derived from a CAD blueprint. Figure 5 shows an exemplary global route graph. With the aid of a global localisation method described in section 6.2 and our feature detection methods (cf. sections 6.3, 6.4), we enriched the basic Voronoi diagram description with additional places for the recognised doors. Thus it is possible to separate local sub-graphs, which represent a single room from the global description. Using this technique 14 out of 16 rooms adjacent to the driven course where detected automatically with no false classification.

6. Problem Solving *or* Hints of Autonomy

In this section we describe our applied algorithms which solve fundamental tasks like local navigation and global localisation. They all rely on the data structures which are presented in sections 4. and 5..

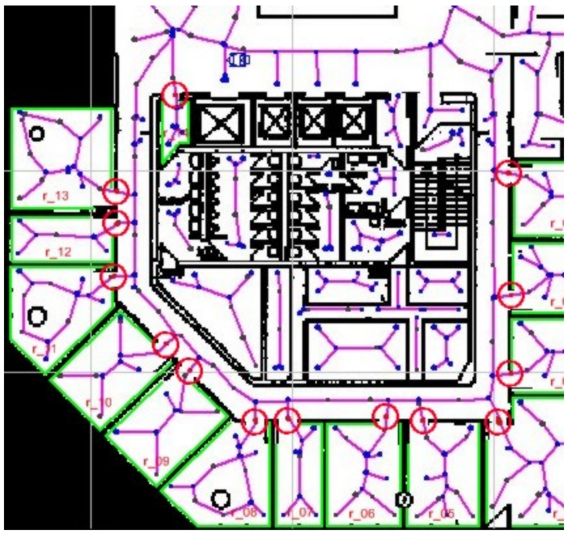


Figure 5: Global route graph with recognised door frames marked by circles.

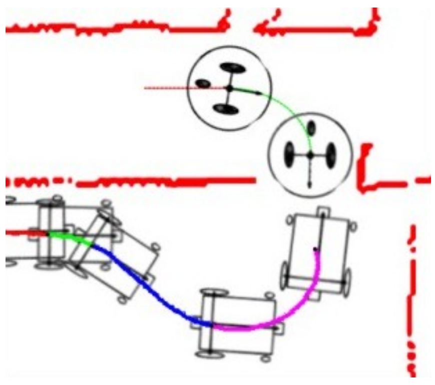


Figure 6: Local trajectories computed by the *Dynamic Window Approach* (top) and our modified method (bottom).

6.1 Local Navigation

Like the widely used *Dynamic Window Approach* (Fox et al., 1995), our local navigation method checks sets of paths for a solution path in order to move the vehicle from its current pose to a given goal-pose. Therefore, the algorithm evaluates a cost function which minimises the distance to the goal and the length of the path to execute, while maximising the distance to the obstacles along the selected path. In contrast to the DWA, candidate paths are composed of clothoids and circular arc segments instead of one individual circular arc. This approach ensures a continuous curvature along a path and can generate the haul-off movement necessary to pass a door frame (cf. figure 6). Furthermore the algorithm improves DWA's collision checking. While the standard DWA assumes a circular shaped robot and therefore gets on by doing simple and fast geometric computation, the developed approach has to operate differently because of our robot's irregular shape, and its centre of rotation which is located in the rear. Thus the proposed algorithm employs a bitmap describing the outer shape of the robot that is matched against the $EvidenceGrid_t$ in order to determine when a collision will occur. Note that our local navigation approach is still under development because several experiments showed problems when entering extremely narrow passages.

6.2 Global Localisation

As an intermediate step before using local $VoronoiDiagram_t$ for localisation, a Monte-Carlo-Localisation method was implemented that is based on the self-locator used by the German RoboCup Team (Röfer and Jüngel, 2003). To establish hypotheses of the current location of the wheelchair, particles are drawn from a pre-computed table that indexes the global environment by the area of the scan perceived from a certain position. In addition, only distance measurements resulting from flat surfaces i. e. from $ScanSegments_t$ are used to determine the current position. Thereby, persons standing around are ignored. The number of particles drawn from observations depends on how good the actual sensor measurements match the expected measurements from the positions of the particles (Lenser and Veloso, 2000).

6.3 Feature Detection using Laser Range Finders

Our basic approach for detecting features in the environment, e. g. doors, relies mainly on analysing the data structures $ScanSegments_t$ and $VoronoiDiagram_t$. In a first step we search the $ScanSegments_t$ correlating to the front and rear scanner for two points which can be connected by a straight line with a length corresponding to the width of the doors in our environment. This yields a list of candidates for detected doors. By employing an angle histogram over the alignment of all elements of $ScanSegments_t$, we are able to filter most false candidates which are not parallel to the main alignment of the scene. Finally the remaining candidate door frames are intersected with the global $VoronoiDiagram_t$. Furthermore false assumptions mainly given by parts of wall segments fall away because they do not intersect an edge of the $VoronoiDiagram_t$.

6.4 Feature Detection using the Omnivision Camera

Another approach to detecting doors in the environment is based on visual perception by the omni directional camera. Recognising doors from the image data requires robust and significant feature extraction. We decided to use symmetry as a natural feature to supplement the scan door recogniser (range information) by a symmetry door recogniser (image information). Huebner (2003) proposed a compact symmetry operator detecting vertical symmetry in image data, which has yet been proved capable of detecting closed doors in a structured office environment (Zhang and Huebner, 2002) by several features.

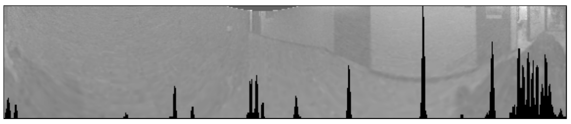


Figure 7: Vertical symmetry histogram for door recognition. Shaded: The panoramic source image from Fig. 4.

While open doors are made accessible and closed doors stay undetected by range information (see Fig. 3 right), the basic symmetry operator particularly extracts vertical symmetry axes out of the corresponding visual information for closed doors (see Fig. 7). Range information can be helpful consulted to reject axes produced by walls or distortion. Each method weakens the sensorial disadvantages of the other, thus both methods complement one another in the task of door recognition.

6.5 Global Navigation and Query Mechanisms

Since the data structure representing the global workspace is organised as a graph with its edges describing the possibility to move the robot along them, our global navigation approach basically builds on an A^* graph search. Additional query mechanisms which enable the user to describe or ask for the location of designated objects e. g. rooms, are implemented by the use of direction based topological relations working on the topological layer of the route graph (cf. figure 8).

7. Safety Issues

Since the setup of the predecessor of our actual experimental platform, safety issues are part of the overall engineering process. Due to the fact that the employed robot has to be classified as a safety critical system, because it shares its workspace at least with its operator, actions have to be taken in order to prevent human beings from any harm. As a result on software side, the so-called *Safety Layer* (Lankenau and Röfer, 2001) analyses any driving command with respect to the current

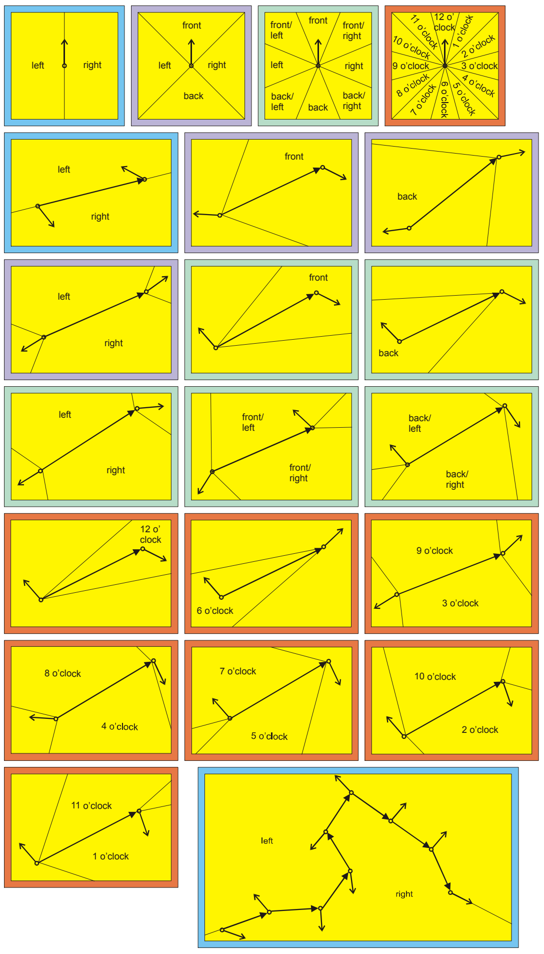


Figure 8: Topological relations interrelating *places*, *route segments* and *routes*.

obstacle situation. It then decides whether the given command is to be forwarded to the actuators or to be replaced by a necessary deceleration manoeuvre. The key concept in the implementation of the *Safety Layer* is the so-called *Virtual Sensor*. For a given initial orientation (θ) of the robot and a pair of translational (v) and rotational (w) speed, it stores the indices of cells of an *EvidenceGrid* that the robot's shape would occupy when initiating an immediate full stop manoeuvre. A set of precomputed *Virtual Sensors* for all combinations of (θ, v, w) now allows to check the safety of any driving command, either instructed by the operator via joystick or by an autonomous navigation process, in real time.

Figure 9 illustrates the application of the *Safety Layer*. It shows the wheelchair located in the center of an $EvidenceGrid_t$ along with the green-coloured *Virtual Sensor*. The screenshot was taken while the robot was driving a left turn with moderate speed, in order to avoid a collision with two people whose signatures can be noticed in the lower right part of the *EvidenceGrid*$_t$. The green area in front of the wheelchair does not intersect with any obstacle point. This clearly indicates that a directly prefaced deceleration manoeuvre would bring the vehicle to a safe stop.

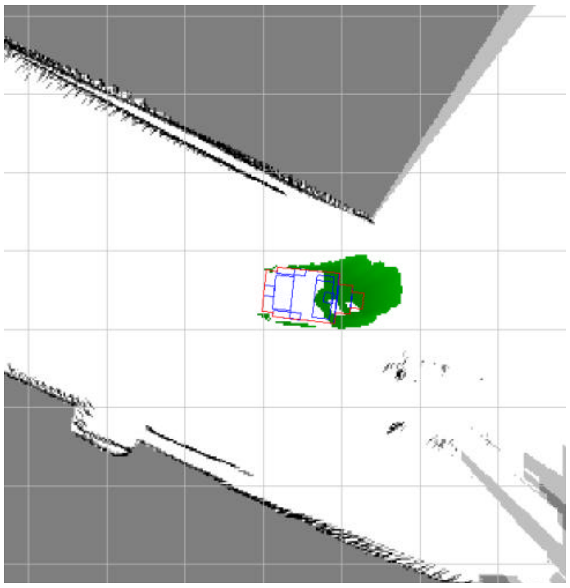


Figure 9: Local *EvidenceGrid*$_t$ and *Virtual Sensor*

8. Language Comprehension *or* The Natural Language Interface

There are several reasons why a natural language interface to robots such as Rolland III are desirable. As the wheelchair is designed to serve as a means of transportation for disabled people, it should be possible to make it move without actually having to move oneself. Second, for most humans oral language is the most natural and expressive way of communication. Being the means (besides gestures) in which humans most likely instruct other humans about navigational issues, natural language interaction is an obvious way of instructing the wheelchair. Finally, Rolland III is deployed as a device for empirical studies about human-computer interaction using natural language (Tenbrink et al., 2002), hence it has to provide this ability in order to be utilised for this purpose.

Currently, Rolland III is equipped with components to interpret language from speech signals to semantic interpretations and to produce spoken feedback according to its internal states. While for speech recognition we rely on off the shelf products, the analysis of syntactic and semantic relations is done by an integrated parser based on work of the SFB 360 “Situated artificial communicators” in Bielefeld. Grounded in Combinatory Categorial Grammar (Steedman, 2000) and already slightly adapted in Bielefeld (Vierhuff et al., 2003), the grammar formalism was further developed to match the necessities found in our domain. It provides a high degree of incrementality and thus is able to come up with intermediate results at early stages of parsing.

Syntax and semantics are represented in a combined formalism processed by an enhanced unification mechanism. For example, the german determiners “einen” and “einem” are difficult to distinguish in continuous speech, especially when followed by words starting with a plosive. Implying a different grammatical case, this could cause unification to fail. By using graded unification (Kim, 1994), we prevent this problem and gain robustness against inconsistencies caused by the speech recogniser as well as irregularities caused by spontaneous speech. The semantic analysis of navigational instructions directly yields an interpretation to be executed by the wheelchair. Spoken feedback about success or failure is then be given back to the user.

The current interaction model between user and robot is simple, but the system is about to be enhanced by a more complete dialogue model within the next few years. This will allow Rolland III to exchange information about landmarks and more complicated navigational instructions given in advance. For instance, places recognised by the robot can be supplied with a name by the user and later referenced in a navigational context, thus supplementing Rolland III's memory with knowledge otherwise not perceivable by the robot.

9. Conclusion and Future Work

We described the autonomous robot Rolland III with its sensorial equipment and the basic types of information which are used for tasks like feature detection and navigation. These techniques allow us to realise basic scenarios, in which the user navigates through an office like environment by talking about places which have been detected by the robot.

Following the paradigm to annotate the global route graph with all information available, a next step is to include information from the operator of the automated vehicle. Spatial statements like “*The vending machine is next to the right.*” will make the system insert a place of kind *HUMAN_ANNOTATION* into the global route graph. This place will be located at the position calculated by the global localiser module and augmented by the laser scans and camera images taken to the time the user gave his statement. Thereupon the local occupancy grids associated with the inserted place will be enriched with a set of samples, describing the region occupied by the referred object in a statistical manner. Beside stating the location of objects w.r.t. an egocentric frame of reference, it will be possible to define object locations w.r.t. entities already available e.g. "*The printer is in room 3050.*" or "*The printer is left to the shelf.*"

Our ongoing work also investigates the possibility to use the medial axis based topological layer of the route graph for still unaddressed problems like simultaneous mapping and localisation.

10. Acknowledgements

This work has partly been sponsored by the Deutsche Forschungsgemeinschaft (DFG) in context of the Sonderforschungsbereich/Transregio 6023 *Spatial Cognition*.

References

Beeson, P., Modayilm, J., Provost, J., Savelli, F., and Kuipers, B. (2003). Exploiting local perceptual models for topological map-building. In *Proceedings of the Eighteens International Joint Conference On Artificial Intelligence*.

Benosmann, R. and Kang, S. (2001). *Panoramic Vision: Sensors, Theory, Applications*. Springer.

Burgard, W., Cremers, A., Fox, D., Hahnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55.

Fiegert, M. and De Graeve, C. (2003). Handling environment dynamics in medial axis based localization. In *Autonome Mobile Systeme*.

Fox, D., Burgard, W., and Thrun, S. (1995). The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13.

Huebner, K. (2003). A 1-dimensional symmetry operator for image feature extraction in robot applications. In *16th International Conference on Vision Interface*.

Kim, A. (1994). Graded unification: A framework for interactive processing. In *32nd Conference on Association for Computational Linguistics*.

Krieg-Brückner, B., Frese, U., Lüttich, K., Mandel, C., Mossakowski, T., and Ross, R. (2004). Specification of an ontology for route graphs. In *Spatial Cognition IV, Lecture Notes in Artificial Intelligence*.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233.

Lankenau, A. and Röfer, T. (2001). A safe and versatile mobility assistant. *IEEE Robotics and Automation Magazine*, 8(1):29–37.

Lenser, S. and Veloso, M. (2000). Sensor resetting localization for poorly modeled mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Röfer, T. and Jüngel, M. (2003). Vision-based fast and reactive monte-carlo localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Steedman, M. (2000). *The Syntactic Process*. MIT Press.

Tenbrink, T., Fischer, K., and Moratz, R. (2002). Spatial strategies in human-robot communication. *KI - Zeitschrift für Knstliche Intelligenz*, 16(4):19–23.

Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hähnel, D., Montemerlo, M., Morris, A., Omohundro, Z., Reverte, C., and Whittaker, W. (2005). Autonomous exploration and mapping of abandoned mines. *IEEE Robotics and Automation Magazine*, 11(4).

Vierhuff, T., Hildebrandt, B., and Eikemeyer, H. (2003). Effiziente verarbeitung deutscher konstituentenstellung mit der combinatorial categorial grammar. *Linguistische Berichte*, 194:213–237.

Vlassis, N., Motomura, Y., Hara, I., Asoh, H., and Matsui, T. (2001). Edge-based features from omnidirectional images for robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Werner, S., Krieg-Brückner, B., and Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In *Spatial Cognition II, Lecture Notes in Artificial Intelligence*.

Zhang, J. and Huebner, K. (2002). Using symmetry as a feature in panoramic images for mobile robot applications. In *Robotik 2002, VDI-Berichte 1679*.

Specification of an Ontology for Route Graphs[*]

Bernd Krieg-Brückner, Udo Frese, Klaus Lüttich, Christian Mandel,
Till Mossakowski, and Robert J. Ross

FB3 Mathematik und Informatik, Universität Bremen,
Postfach 330 440, D-28334 Bremen
bkb@informatik.uni-bremen.de

Abstract. This paper describes the general concept of Route Graphs, to be used for navigation by various agents in a variety of scenarios. We introduce the concept of an ontology and describe the modelling of general graphs as an example. This approach is then applied to define a "light-weight" ontology of Route Graphs in an indoors environment, giving at first just a taxonomy of (sub)classes and relations between them, as well as to other (spatial) ontologies. Finally, we show how to formalise ontologies using a First Order Logic approach, and give an outline of how to develop actual data structures and algorithms for Route Graphs.

1 Introduction

Route Graphs have been introduced as a general concept ([1]), to be used for navigation by various agents in a variety of scenarios such as humans using railway, underground, road or street networks, as travellers, car drivers or pedestrians, resp. Each application scenario or kind of Route Graph will introduce special attributes on the general structure in a hierarchy of refinements.

While the concept was originally introduced to mediate terminology between artificial intelligence and psychology in spatial cognition, scenarios are not restricted to human users. Route Graphs are also intended for interaction between service robots, such as the Bremen autonomous wheelchair Rolland (cf. Fig. 1), and their users, as well as between robots, for example as a compact data structure for on-line communication in an exploration scenario. Moreover, a bare-bones Route Graph representation is easily constructed from pre-existing map-like representations (at least for floor plans of buildings) or by robot exploration, and we hope that cognitively (more) adequate maps can be constructed from it when the semantic interrelation can be taken into account.

As we are dealing with abstract concepts and interrelations between them and want to standardise or mediate between different uses of terminology, we are using an ontology as *the* central definitional approach and data structure for Route Graphs. We intend to show that an ontological approach is suitable

[*] This paper has been supported by the Deutsche Forschungsgemeinschaft (DFG) in the SFB/TR 8 Spatial Cognition.

C. Freksa et al. (Eds.): Spatial Cognition IV, LNAI 3343, pp. 390–412, 2005.

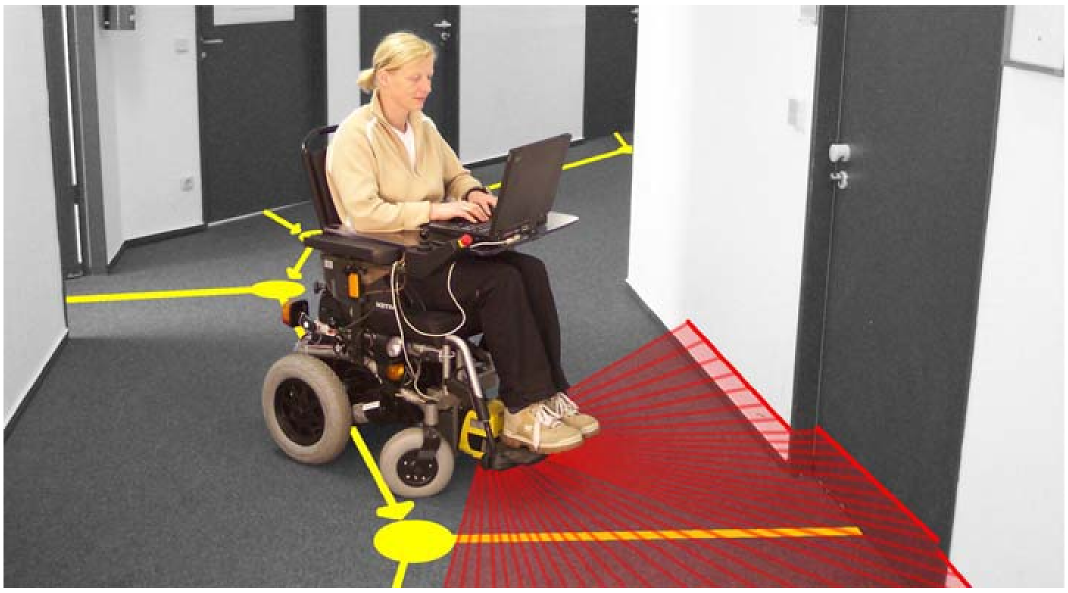


Fig. 1. Rolland following a Route Graph

for both: to define the generic concept of Route Graphs and to instantiate it for a particular scenario in detail, leading eventually to a concrete data structure. Moreover, the example of Route Graphs is suitable for demonstrating that adequate formalisation of ontologies can be introduced step by step in this process.

In this paper, we will show the application to an indoors scenario for Rolland. Thus we have to model detailed information for navigation at the “robot level” as well as more abstract concepts of space at the “user level”, and the relationship between these and “common sense” concepts in the environment, such as rooms, doors or windows as route marks (cf. [2]). For a dialogue between user and wheelchair, we want to relate the concepts of Route Graphs to linguistic ontologies; this is described e.g. in [3, 4].

The paper is structured as follows: We first sketch the general concept of Route Graphs in Sect. 2. In Sect. 3 we introduce the concept of an ontology and describe the modelling of general graphs as an example. This approach is then applied to define a “light-weight” ontology of Route Graphs in an indoors environment, giving at first just a taxonomy of (sub)classes and relations between them, as well as to other (spatial) ontologies, in Sect.4. Finally, we show in Sect.5 how to formalise ontologies using a First Order Logic approach, and give an outline of how to develop actual data structures for Route Graphs.

2 Route Graphs

2.1 Sample Scenarios

Figure 2 shows some sample navigation scenarios. We may distinguish between navigation in systems of passages (e.g. road networks or corridors) or in areas of open space (e.g. on a lake; on a market place, surrounded by buildings; in a hall);

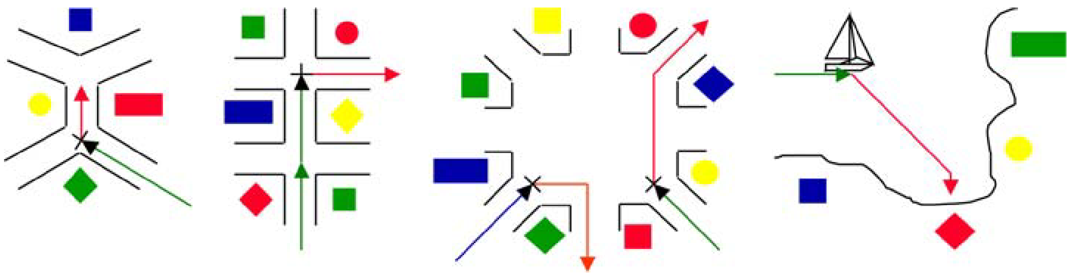


Fig. 2. Examples of routes

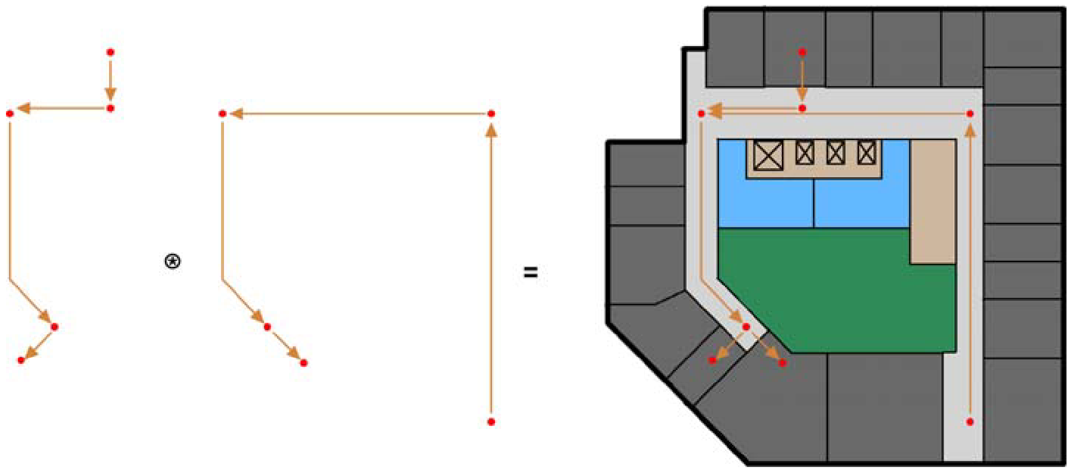


Fig. 3. Two routes and their union

in the former, our course is more or less centred within enclosing borders (e.g. curbstones, walls) and guided by routemarks along the way, in the latter the course is given by a vector to the target and we are guided by global landmarks (e.g. a lighthouse, the sun or a church's spire), cf. [5].

While *Route Graph*s should apply to all such scenarios (cf. [1]), we will concentrate on the indoors scenario here, for Rolland and its user as in Fig. 1, as an example for service robot applications. We briefly introduce the basic concepts of Route Graphs here; more detail will follow in Sect. 3.2 and Sect. 4.

Sample Route: to the Secretary's Office. Consider Fig. 3: two separate routes are united into a simple Route Graph. Let us take the first route as an example. It can be described by directions in natural language as follows:

– Leave Room MZH 8210 into the corridor
– Turn right, go straight ahead to the window
– Turn left, follow the corridor until the last door on the right-hand-side
– Face it, enter Room MZH 8080

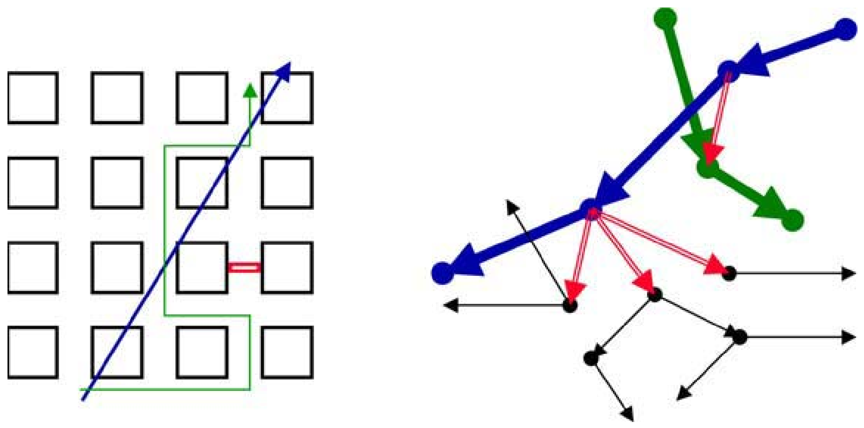


Fig. 4. Layers and transfers

Sample Route Segments. The first two lines can then be translated into the following two route segments (taking additional information about doors, lifts, windows into account, see Sect. 4.4):

Source: room MZH 8210	*Source*: corridor, facing the lift
Entry: turn towards the door	*Entry*: turn right
Course: go through the door	*Course*: follow corridor to the window
Exit: [turn to face the lift]	*Exit*: [turn to face the window]
Target: corridor, facing the lift	*Target*: T-crossing, facing the window

Segments. An edge of a Route Graph is directed from a source node to a target node. We call an edge a *(route) segment*; it always has three additional attributes: an entry, a course and an exit. Exactly what information is associated with these attributes is specially defined for each Route Graph instantiation of a particular kind. For example, an entry to a highway may denote a particular ramp, an exit another, while the course is just characterised by a certain length. Additional information may be added, e.g. that the course has three lanes. As another example, the entry and course for a boat route segment may be given as a vector in geo-coordinates, while the exit into a harbour may specify a particular orientation along a quay.

Places. A node of a Route Graph, called a *place*, has a particular position and orientation, its *origin*. Thus each node has its own "reference system"; it may, but need not, be rooted in a (more) global reference system, such as a 3D geo-system. The origin becomes particularly important in a union of routes or Route Graphs, when place integration is required (see Sect. 4.3).

2.2 Homogeneous Route Graphs, Transfers

Fig. 4 shows, on the right hand side, an example where various Route Graphs have been united to one heterogeneous Route Graph. We say a Route Graph of a particular kind *is homogeneous*, if all its segments are of the same kind. In the

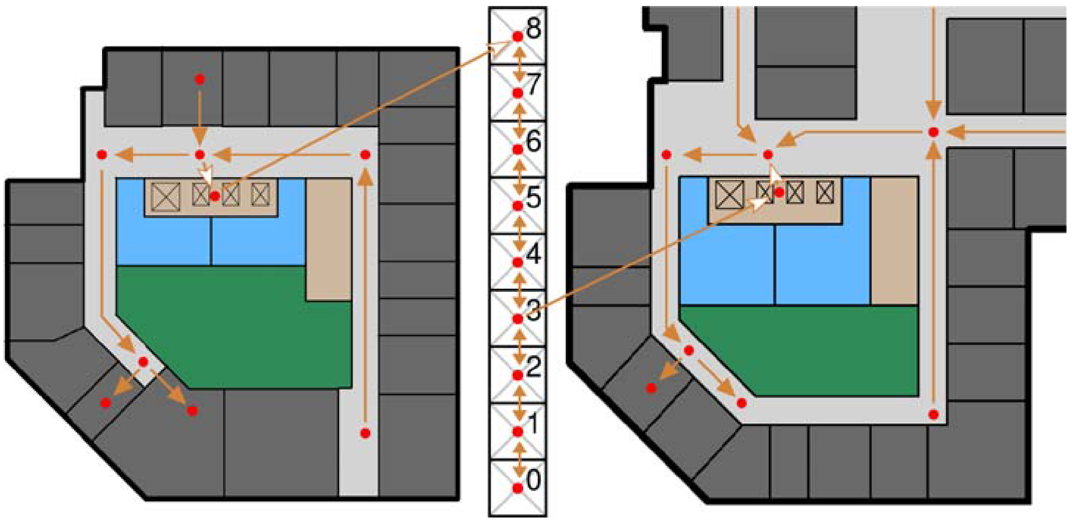


Fig. 5. Two levels and lift

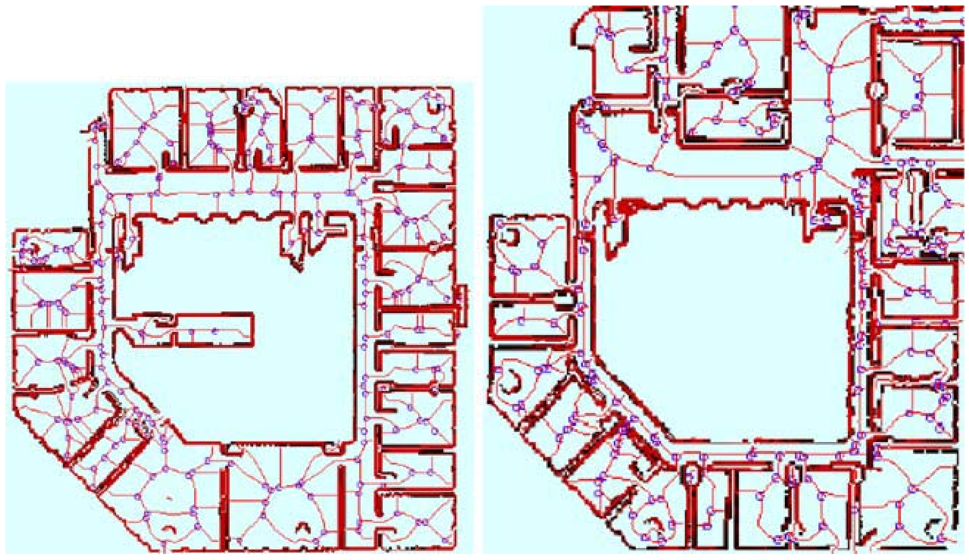


Fig. 6. Voronoi representation for levels 8 and 3

example, the routes depicted by fat arrows might correspond to underground train lines, those with fine arrows to a network of pedestrian passages; both are homogeneous. There is then a need to introduce *transfer segments* for connection, or indeed *transfer Route Graphs* that have transfer segments at their fringes. In the example they would correspond to a pedestrian transfer passage between two underground stations, or several exits from an underground station to the pedestrian network above (note that these might be connected to other exit or entry routes underground). Fig. 5 shows a transfer between a Route Graph at level 8 of our office building, the lift system, and level 3.

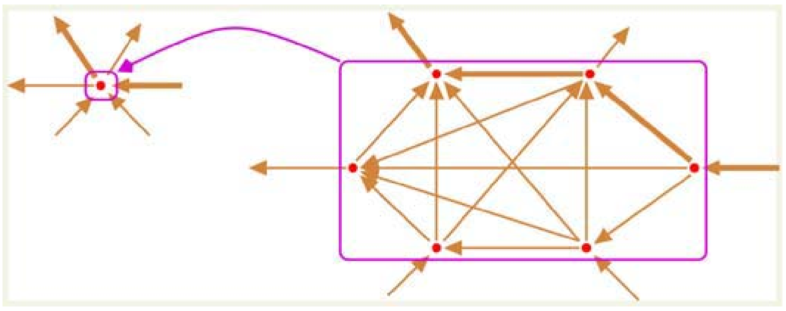


Fig. 7. Abstraction to place

2.3 Layers and Abstraction

We may wish to separate Route Graphs into *layer*s at different levels for conceptual reasons, possibly unconnected. One reason might be that we want to represent route and overview knowledge (cf. [1]). The left hand side of Fig. 4 sketches a route in an urban block scenario, and the target designated by an arrow. When circumnavigating a road blockage, for example, we are likely as humans to use such an arrow as a "sense of direction" while negotiating our way in the block maze – we are using and combining two layers of knowledge at the same time; it is well known that other animals are much better in this respect.

Abstraction. Another reason is abstraction from a more concrete and detailed lower layer to a higher one, as in the abstraction from the robot level to the user level (see also Sect. 4.1). Fig. 5 refers to the user level, while Fig. 6 shows a Voronoi diagram as a representation of space at the robot level (cf. also Sect. 4.2) which corresponds directly to a detailed Route Graph, if we replace the trajectories between places by straight directed edges, in both directions.

What is required here is a relation `abstractsTo` between graphs (in fact a graph morphism), more specifically from a route to a segment, or a graph to a node as in Fig. 7 (cf. also Fig. 16). When a set of nodes SN on the fringe of the graph G at the lower layer is abstracted to a single node N, some obvious conditions must hold, e.g. all incoming/outgoing edges to a node in SN must correspond to incoming/outgoing edges for N; for each pair of incoming and outgoing edges for N there must be a corresponding connecting path in G; etc.

3 Modelling Via Ontologies

Ontologies provide the means for establishing a semantic structure. An ontology is a formal explicit description of concepts in a domain of discourse [6]. Ontologies are becoming increasingly important because they also provide the critical semantic foundations for many rapidly expanding technologies such as software agents, e-commerce, or the "Semantic Web" [7]. In artificial intelligence, ontologies have been used for knowledge representation ("knowledge engineer-

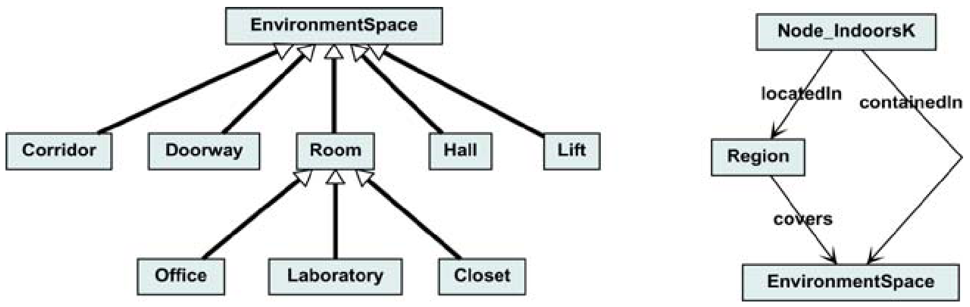


Fig. 8. Subclasses of `EnvironmentSpace` and relations

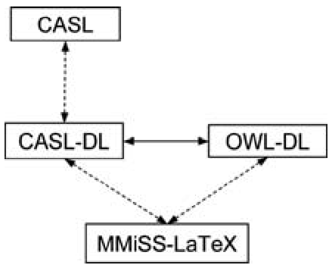


Fig. 9. Various ontology specification formats

ing"). The general idea is to make knowledge explicit by expressing concepts and their relationships formally with the help of mathematical logics.

An *ontology* consists of (a hierarchy of) concepts and relations between these concepts, describing their various features and attributes. Classes and relations are used for defining the abstract semantical level; they provide a vocabulary and properties to characterise the concrete entities corresponding to these concepts. Once the abstract notions are declared in terms of classes, objects can be used to denote entities of semantic concepts. As a simple example for an ontology consider Fig.8. On the left, it depicts a hierarchy of subclasses of the *class* `EnvironmentSpace` – an extract of a taxonomy as it might appear in a "Common Sense Ontology" of space[1], cf. [2]. The relation *is a subclass of* (or "*is a*") is depicted by a fat arrow with a hollow tip. For example, an `Office` is a `Room` which is in turn an `EnvironmentSpace`.

A class represents a set of *objects*; for example `Office8210` is an object (depicted with leading and trailing underscores in the diagrams below) of class `Office`, or `Office8210`: `Office`.

On the right, we see declarations of *relation*s depicted by pointed arrows, with classes as domains and co-domains, to be formalised (see Sect.5) and used eventually for the definition of relations between objects. ***containedIn***, for example, relates the class `Node_IndoorsK` of our RouteGraph ontology (to be de-

[1] We refer to different ontologies separately here although they are in fact parts of one combined ontology; for the structuring of ontologies see Sect.4.8.

fined below) to `EnvironmentSpace` in the "Common Sense Ontology"; moreover, `Node_IndoorsK`s in the RouteGraph ontology are related by ***`locatedIn`*** to (a separate ontology for) a calculus of `Region`s. Thus we should eventually be able to deduce, given
`Place8210: Node_IndoorsK`, `Region8210: Region`, and `Office8210: Office`, then
`Place8210 locatedIn Region8210` and `Region8210 covers Office8210` implies `Place8210 containedIn Office8210`

3.1 Various Ontology Specification Languages

Description Logic in OWL. Ontologies may come in various specification formats, cf. Fig.9 (or [8]). We adhere to the OWL standard [9]. Its description logic DL has been primarily defined to be decidable for the Semantic Web.

Lightweight Ontologies in LaTeX. For lightweight ontologies, just (sub)classes, objects and relations as in the example above, we use a special LaTeX format that can be translated to OWL and vice-versa (a tool [10], based on the generic graph visualisation tool daVinci [11], supports incremental presentation of and navigation in such graphs, as in the examples shown in this paper). It allows the specification of ontologies for documents to enable their semantic interrelation, enabling much more advanced document management facilities ([12, 13]).

First Order Logic in CASL. In contrast, Casl, the Common Algebraic Specification Language approved by IFIP WG1.3 ([14, 15]), covers full First Order Logic, plus predicates, partial and total functions, and subsorting; more-

MMiSSLaTeX
```
\Class{Graph}{graph}{}
      \RelType{hasNode}{Graph}{Node}
      \RelType{hasEdge}{Graph}{Edge}
\Class{Node}{node}{}
\Class{Edge}{edge}{}
      \RelType{hasSource}{Edge}{Node}
      \RelType{hasTarget}{Edge}{Node}

      \Relation{*-*}{hasNode}{has a node}{}
      \Relation{*-*}{hasEdge}{has an edge}{}
      \Relation{->}{hasSource}{has the source}{}
      \Relation{->}{hasTarget}{has the target}{}

\Class{Sequence_Edge}{\Ref{Edge} list}{}
 \Class{Path}{path}{Sequence_Edge}
  \Class{Route}{route}{Path}
```

Fig. 10. Path and route

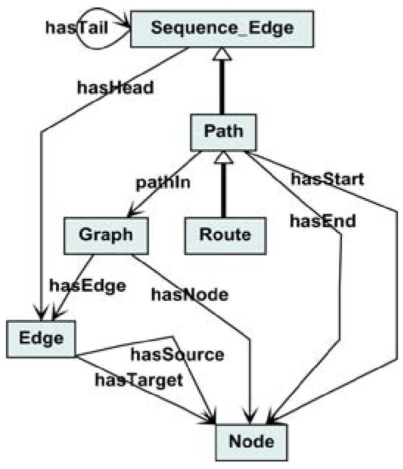


Fig. 11. Simple graph ontology (extract)

over, features for structuring, or “specification in the large”, have been included (cf. Sect.4.8). Thus full formalisation of ontologies becomes possible, (as in the DOLCE framework [16, 17]), needed eventually here, see Sect.5. A sublanguage, CASL-DL, has been defined to correspond to OWL DL [18] such that the mappings/embeddings of Fig.9 between the various representations can be realized.

3.2 Ontology of Graphs

As an example, consider a simple ontology of graphs. Fig.10 shows the core in MMiSSLaTeX: A `Graph` has (one ore more) `Node`s and `Edge`s. The MMiSSLaTeX operation `Class` declares these classes: the first parameter denotes the semantics term, the second a default textual phrase (for use in LaTeX documents), the third the superclass.The operation `Relation` is analogous, but contains as an additional first parameter an indication of the kind of relation, e.g. “`*-*`” to denote “many to many”, “`->`” to denote “onto” (a function), or “`<`” to denote “strict partial order”. With the operation `RelType` relations may be “typed” by source and target classes to allow (static) checking of conformance when objects are related. Thus each `Edge` has exactly one source and one target `Node`.

Note that, at the level of an ontology specification, we do formalise differently than in mathematics or a classical modeling in a specification or programming language: in the latter case, we would introduce a graph as a pair for a set of nodes and edges; here, a `Graph` contains both sets – the abstraction from the pair is perhaps even more natural and definitely adequate as a first go. We show in Sect.5.1 how a more “data structure” oriented design may come back in.

3.3 Paths and Routes

The last lines of Fig.10 show an extension of the basic graph specification. A `Path` is (a subclass of) a `Sequence` of `Edge`s (`Sequence` is a basic concept with additional relations, instantiated here), where the target of the first edge is the source of the second, and so on. Note that a `Path` corresponds to some graph traversal, possibly with cycles. In contrast, a `Route` contains no repetition of `Edges`. Such properties will have to be formally specified by additional axioms on the subclasses, successively refined from `Sequence_Edge` over `Path` to `Route`, cf. Sect.5.1. The visualisation in Fig.11 shows a few more relations, sometimes with multiple relation arrows, e.g. `hasSource` and `hasTarget` from `Edge` to `Node`.

4 A Route Graph Ontology for an Indoors Scenario

4.1 Instantiation to Particular Route Graphs

We assume that the general properties of (labelled) directed graphs are specified along the outline above; similarly, abstract route planning algorithms could be specified at this level without knowing (much) more about `Nodes` or `Edges`.

Indoors Instantiation and Kinds. We instantiate this general graph ontology here for an indoors application of Route Graphs, see Fig.13 (cf. also Sect.4.8 for the instantiation aspect). Thus `Graph` becomes `Graph_IndoorsK`, `Node` becomes `Node_IndoorsK`, and so on. `KindRG` denotes the kind of Route Graph, where `IndoorsK` may be refined later on as shown in Fig.12; other kinds might denote `RailwayK`, `UndergroundK` or `RoadK` Route Graphs, for example.

User Level and Robot Level. Our `IndoorsK` instantiation refers to the (indoors) operating environment of a robot. Note that we do a particular modelling for Route Graphs here which is separate from, but related to, the "Common Sense Ontology" for `EnvironmentSpace` in Fig.8; we have to expect that they are structured differently. As we shall see below, a `Graph_IndoorsK` may refer to a rather detailed description at the level of a robot, e.g. the wheelchair Rolland, or its abstraction at the level of an operator, e.g. a Rolland user. User and robot

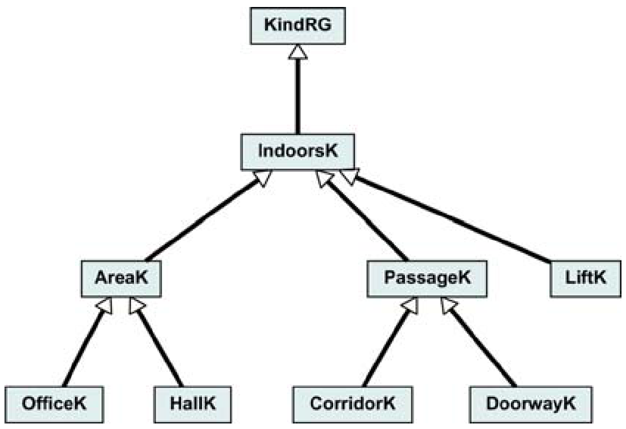


Fig. 12. Ontology of `KindRG`

interact at this abstract level, while robots may interact among themselves at the detailed level, for example during a multi-robot exploration phase.

4.2 Edges and Route Segments

With the transition from graphs to Route Graphs, an `Edge_IndoorsK` (with source and target) is refined to a `Segment_IndoorsK` (see Fig.14), with an additional entry, course and exit, see Fig.15. Each `KindRG` of Route Graph will introduce its special aspects, in particular (relations to) components.

Vectors and Voronoi Diagrams. In the specialisation of `IndoorsK` Route Graphs, a segment is essentially modelled as a vector in polar coordinates: the entry gives the angle from (the origin of) the source place, the course the distance to the target , and the exit the angle to the target, to assume the orientation of (the origin of) the target place (cf. Sect.4.3). In addition, we attach information derived from a Voronoi diagram representation (see also [19]): each point has a maximal circle of available free space around it, represented by its diameter or `Width`. Thus each `Place_IndoorsK` has a `Width`, and the `Width` of a `Course_IndoorsK` denotes the minimal width of the points along its course.

Abstraction of Segments. Fig.16 shows a Voronoi diagram at the bottom corresponding to a detailed Route Graph for the route in the corridor which has been abstracted to a single segment at the top, showing the `CorridorWidth`.

Consider also the example in Fig.17, corresponding to that in Fig.3 and given verbally in Sect.2.1: `Segment1` represents a passage through a doorway, of `DoorWidth`; `SegmentCorridor2` represents (part of) the corridor, of `CorridorWidth`. The entry and exit angles are depicted by fat little (angular) arrows. Fig.15 and Fig.18 show the ontology of segments and the objects related to the particular segment `SegmentCorridor2`. Note that the entry angle and the distance of the course denote the *direct* vector to the target place thus defining a spatial relation between the two nodes; the actual Voronoi trajectory (cf. e.g. `Segment1`) and its length (which may be larger than the distance) are not represented here; they could be added to the course as additional information, but are

MMiSSLaTeX
```
\Class{Graph_IndoorsK}{\Ref{IndoorsK} graph}{}
      \RelType{hasNode}{Graph_IndoorsK}{Node_IndoorsK}
      \RelType{hasEdge}{Graph_IndoorsK}{Edge_IndoorsK}
\Class{Node_IndoorsK}{\Ref{IndoorsK} node}{}
\Class{Edge_IndoorsK}{\Ref{IndoorsK} edge}{}
      \RelType{hasSource}{Edge_IndoorsK}{Node_IndoorsK}
      \RelType{hasTarget}{Edge_IndoorsK}{Node_IndoorsK}
\Class{Sequence_Edge_IndoorsK}{\Ref{Edge_IndoorsK} list}{}
 \Class{Path_IndoorsK}{\Ref{IndoorsK} path}{Sequence_Edge_IndoorsK}
  \Class{Route_IndoorsK}{\Ref{IndoorsK} route}{Path_IndoorsK}
```

Fig. 13. Indoors Graph ontology

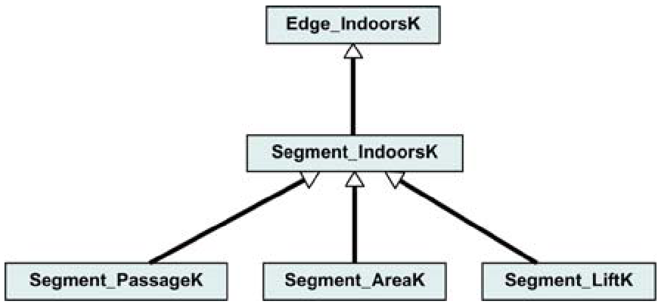


Fig. 14. Ontology of indoors edge subclasses

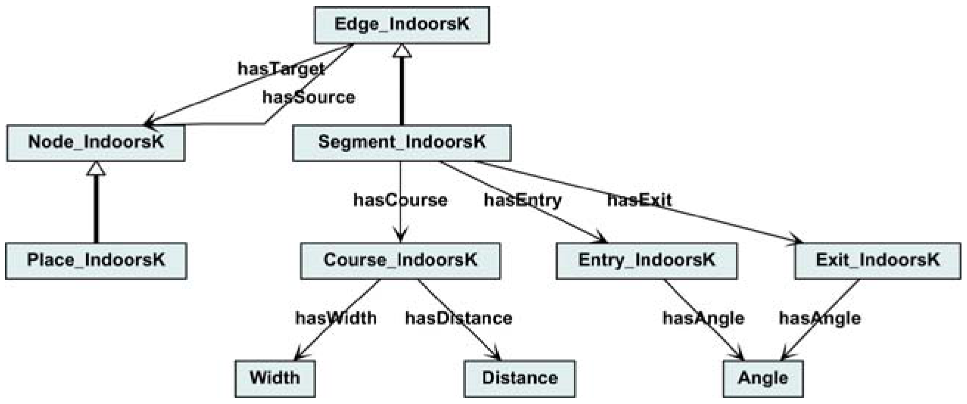


Fig. 15. Ontology of indoors segment subclasses

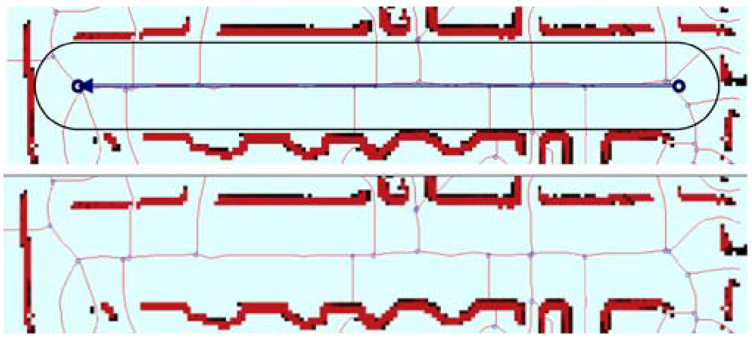


Fig. 16. Abstraction of route in corridor to route segment

probably not necessary since the construction of the actual navigation trajectory from the Route Graph has to take e.g. dynamic obstacles into account.

Modelling Precision. All measurements above (distances, angles, etc.) are intended to be qualitative; in fact no units have been specified so far. Precision can be introduced by an additional attribute, for example an interval of tolerance.

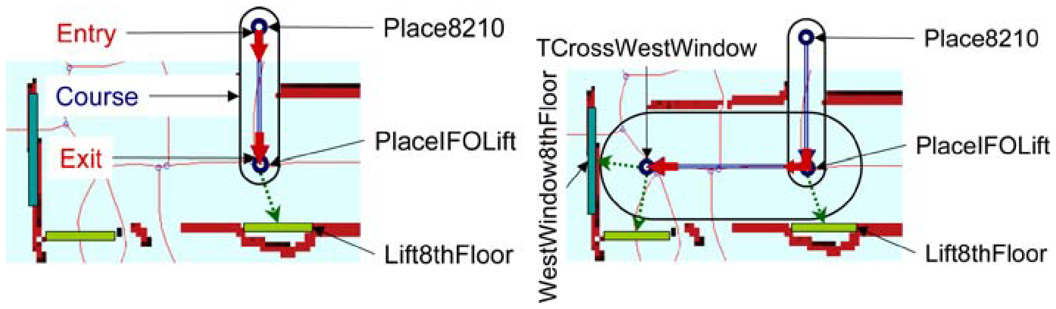


Fig. 17. `Segment1` and `SegmentCorridor2`

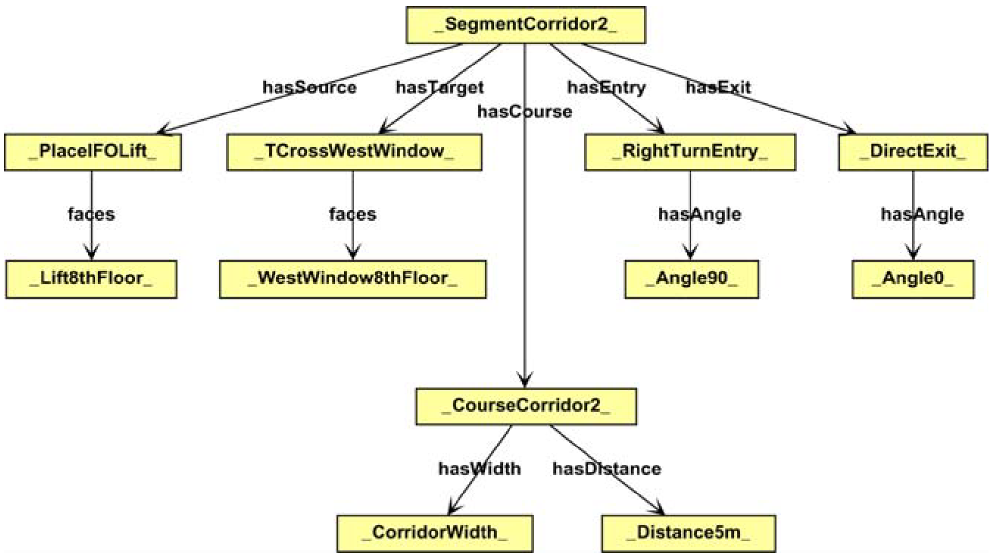


Fig. 18. Ontology of object instantiations representing `SegmentCorridor2`

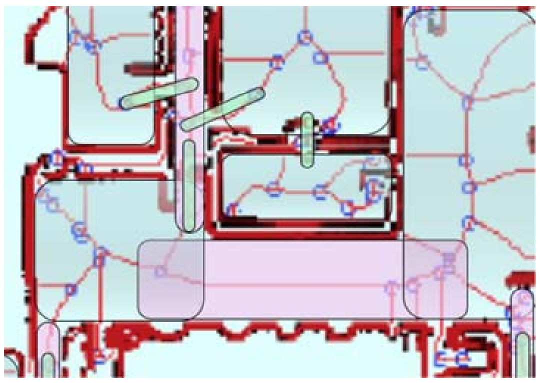


Fig. 19. Overlapping regions

4.3 Nodes and Places

A `Node_IndoorsK` is refined to a `Place_IndoorsK`; each place has its own *origin*, i.e. a well-defined position and orientation with a zero angle. For example (cf.

Fig.18), `Place8210 faces` the office door, `PlaceIFOLift faces` the lift, and `TCrossWestWindow faces` the west window. We might wish to define an origin to coincide with such a `faces` relation, but in fact it does not matter much as it is only important to have a well-defined origin at all.

Place Integration. How is this origin preset? When a node has just one segment attached, its entry angle is zero for an outgoing segment, or its exit angle is zero for an incoming segment.[2] When adding an additional segment, this origin will have to be adhered to. When integrating two routes or in a union of Route Graphs, *place integration* has to be performed for all segments with common places: for any two places to be integrated, a common origin will have to be chosen and the entries/exits of the segments re-computed, if necessary. This is the price to pay for the fact that a place has now become a position of choice for outgoing segments, with different entries in general.

Regions. Places are contained in regions. Recall Fig.8: If a `Place` is `locatedIn` a `Region` and this `Region covers` an `EnvironmentSpace`, say an `Office`, then we may conclude that this `Place` is `containedIn` this `Office`.

Fig.19 shows overlapping and nested regions, where the regions can be classified according to the `KindRG`s in Fig.12.

4.4 Relation to "Common Sense Ontology"

The relations `locatedIn`, `covers` and `containedIn` are examples of relations between different ontologies (or parts of one big joint ontology); this issue is taken up further in [2]. For example, there are various calculi for regions or other spatial configurations (e.g. [20, 21]); the ontology of such calculi, formally specified as suggested in Sect.5, could be associated here.

Facing Windows. is another example. Consider Fig.20 (also the objects in Fig.18): a place can be classified (as a subclass of `Place_IndoorsK`) as a `Place_InFrontOfWindow`, a `Place_InFrontOfLift`, etc.; it may then be related by `faces` to an object in the "Common Sense Ontology", e.g. a `Window` or `Lift`.

Routemarks. In such a situation, we may wish to attach an actual pointer to such an object, i.e. a `Vector_Mark_IndoorsK` having a `Node_IndoorsK`, e.g. a `Place_InFrontOfWindow`, as source and a `Routemark` as target; this `Routemark` is then a `Point` that `marks` a `Window` (e.g. in its centre). As examples, consider the dotted arrows in Fig.17. Routemarks help self-localisation during navigation using such vectors for triangulation.

4.5 Reference Systems

Analogously, a place may be rooted in a global reference system by a vector from its origin; this could also be a 3D Cartesian vector, of course. In fact, it is quite likely that such reference systems correspond to nested regions (Sect. 4.3).

[2] A origin has to be set before a pointer to a routemark is specified, cf. Sect.4.4.

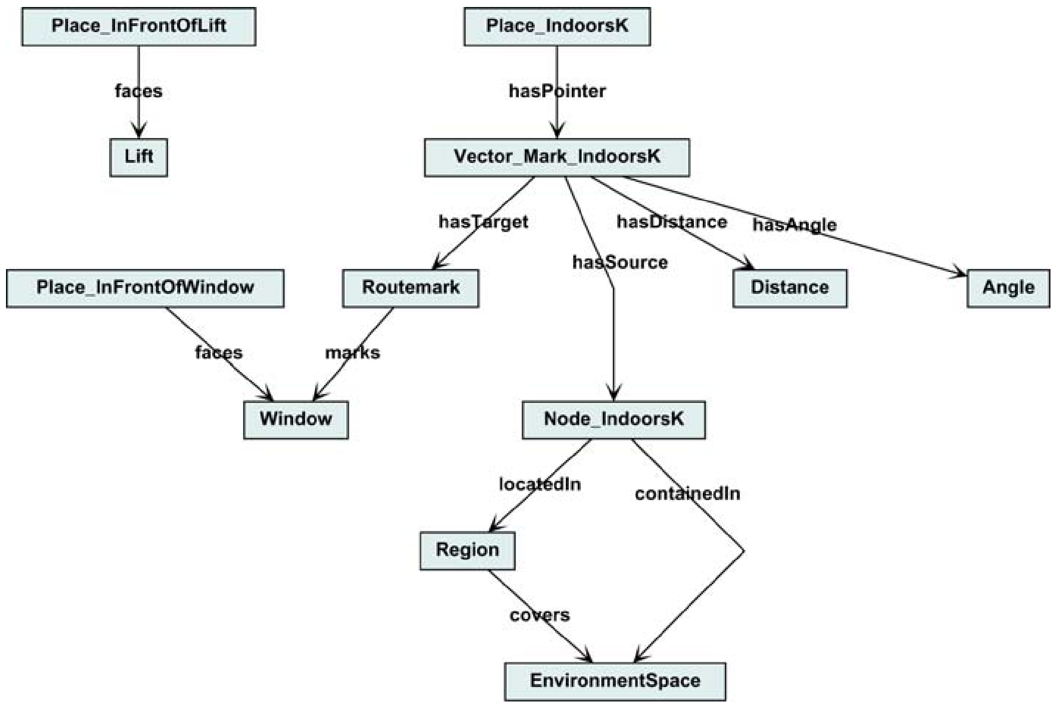


Fig. 20. Places facing a window, and routemarks

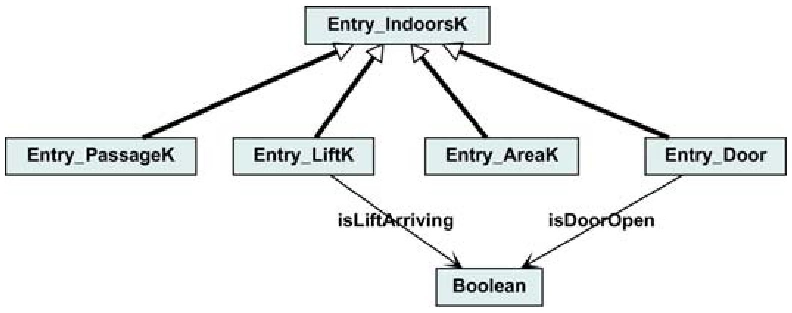


Fig. 21. Entries with dynamic predicates

4.6 Multi-robot Exploration

Multi-Robot Exploration may require additional information to be attached to places, for example a tag marking it as a fringe node for an explored region, to be extended. When various robots co-operate during exploration, place integration becomes of utmost importance, i.e. a criterion for matching two places, to become the same when "closing the loop" (cf. [22, 23, 21]). In such as case, different co-existing Route Graphs denote different possible worlds, with a certain probability attached; identical sub-graphs may of course be shared. We hope that the Route Graph representation will prove to be sufficiently compact for interaction between robots in such situations.

```
MMiSSLaTeX
\NewDecl{GenGraph}[1]  %[KindRG]
{\Class{Graph_#1}{\Ref{#1} graph}{}
      \RelType{hasNode}{Graph_#1}{Node_#1}
      \RelType{hasEdge}{Graph_#1}{Edge_#1}
\Class{Node_#1}{\Ref{#1} node}{}
\Class{Edge_#1}{\Ref{#1} edge}{}
      \RelType{hasSource}{Edge_#1}{Node_#1}
      \RelType{hasTarget}{Edge_#1}{Node_#1}
\GenSequence{Edge_#1}
 \Class{Path_#1}{\Ref{#1} path}{Sequence_Edge_#1}
  \Class{Route_#1}{\Ref{#1} route}{Path_#1}
```

Fig. 22. MMiSSLaTeX source of generic Graph ontology

4.7 Modelling Dynamic Information

So far, all the information associated with Route Graphs was static. For use in navigation, it would be very convenient to be able to model dynamic information as well. Consider Fig.21: entries are guarded by dynamic predicates,[3] to check whether a door is open or a lift is arriving. This way, Rolland may for example choose among segments to several lifts, depending on which is about to arrive. Thus status information is modelled; our graph can be regarded as a kind of state transition graph.

From a puristic point of view, this modelling is adequate for a data structure, but violates the idea of an ontology. However, the ontology (and the "data base" of its associated objects) should be the central source of information, the basic knowledge representation. It will yet have to be seen how these two views can be reconciled better, perhaps by representing dynamic information in a completely separate part of the ontology.

4.8 Structuring Ontologies

Finally, a word about structuring ontologies. Ontologies, as all descriptions or formal specifications, may become quite large and unwieldy; structuring becomes a necessity. We have learned a lot from structuring formal specifications or theories in the context of Casl, cf. [14, 15]: composite specifications may e.g. be constructed by (conservative) extension or union, items may be re-named, morphisms ("views") may be applied, (remote) libraries may be organised as nested folders.

We suggest to apply similar structuring mechanisms to ontologies. One way to do this is the approach of the MMiSS project, where general documents are structured along these lines, and change management supports sustainable development (cf. [12, 13]).

[3] Boolean valued binary relations to emphasise the dynamic query rather than unary predicates represented e.g. as subclasses.

Generic Ontologies. A particularly important structuring mechanism is parameterisation, or abstraction to generic modules, generic (sub)ontologies here. Fig.22 shows a generic graph ontology, as a generalisation of Fig.10: the (only) parameter "#1" may then be instantiated to different specialisations of `KindRG`, e.g. `IndoorsK`, `RailwayK` or `RoadK`. Note that, in its body, `GenSequence` is instantiated, a similarly generalised version of sequences.

5 Formalisation in Casl

This section contains the formalisation of the Route Graph ontology and the RouteGraph data type specification. There are several advantages for using Casl for the entire development of a data structure, together with its functions, from an ontology (see [14, 15, 16, 17], cf. Sect. 3.1, Fig. 9):

Casl covers full First Order Logic, predicates, partial and total functions, and subsorts. Tools in Hets (see below) are available for strong (sub-)type checking, overloading resolution, and interfaces to verification tools.

Casl-DL is a sublanguage of Casl, restricted to the Description Logic underlying OWL DL. It offers precise concepts and definitions together with a translation from and to OWL DL, cf. Fig. 9 and [18]. Thus a variety of tools developed for OWL DL become available for Casl-DL. The LaTeX ontology representation of **??**FirstOrderLogic can be translated to Casl-DL (or Casl) to provide a bare-bones specification.

```
spec GenGraph [sort Kind] =
     sorts Graph[Kind], Node[Kind], Edge[Kind]
     preds hasNode : Graph[Kind] × Node[Kind];
           hasEdge : Graph[Kind] × Edge[Kind]
     ops   hasSource, hasTarget : Edge[Kind] → Node[Kind]
     ∀ e: Edge[Kind]; s, t: Node[Kind]; g: Graph[Kind]
     • hasEdge(g, e) ∧ hasSource(e) = s ∧ hasTarget(e) = t ⇒
       hasNode(g, s) ∧ hasNode(g, t)
then
     GenSequence [sort Edge[Kind]] with hasHead, hasTail, EmptySeq, freq
then ...
     op    sources : Sequence[Edge[Kind]] → Sequence[Node[Kind]]
     ...
     pred  connected(l: Sequence[Edge[Kind]]) ⇔
           (∀ e1, e2: Edge[Kind]
            • hasHead(l) = e1 ∧ hasHead(hasTail(l)) = e2 ⇔
              hasTarget(e1) = hasSource(e2)) ∧ connected(hasTail(l))
     sorts Path[Kind] = {l: Sequence[Edge[Kind]] • connected(l)};
           Route[Kind] = {p:Path[Kind]
                            • (∀ n: Node[Kind] • freq(sources(p), n) ≤ 1) ∧
                              (∀ e: Edge[Kind] • freq(p, e) ≤ 1)}
end
```

Fig. 23. Generic Graph ontology in Casl

```
from BASIC/STRUCTUREDDATATYPES get LIST
from BASIC/NUMBERS get NAT
from BASIC/GRAPHS get NONUNIQUEEDGESGRAPH

spec GRAPH2 [sort Kind] given NAT =
     sorts Node, Edge
     ops   source, target : Edge → Node
then
     NONUNIQUEEDGESGRAPH [sort Node] [sort Edge] and
     LIST2 [sort Edge] with [], __::__, atMostOnce
then
     pred  connected : List[Edge]
     ∀ rs1, rs2: Edge; rsl: List[Edge]
     • connected([])                                         %(connected_empty)%
     • connected([rs1])                                      %(connected_singleton)%
     • connected(rs1 :: rs2 :: rsl) ⇔
          target(rs1) = source(rs2) ∧ connected(rs2 :: rsl)  %(connected_rec)%
then ...
     op    sources : List[Edge] → List[Node]
     ...
     sorts Path = {l: List[Edge] • connected(l)};            %(def_path)%
           Route = {l: Path • atMostOnce(sources(l)) ∧ atMostOnce(l)}
                                                             %(def_route)%
end
```

Fig. 24. Design specification in CASL

HETS – the Heterogeneous Tool Set – provides support for heterogeneous specification in various different logics (e.g. ModalCASL) and associated tools, most notably the interactive verification system Isabelle [24] and an increasing number of (semi-)automatic theorem provers, e.g. SPASS [25]. Translators to some target programming languages such as HASKELL or JAVA are under development. Moreover, HETS will support re-use of proofs with the Development Graph Manager MAYA [26].

We illustrate the development process proposed here with the example of generic (Route) Graphs.[4] It starts with an ontology stating only classes, relations and their axioms in a very loose fashion. This gives a top-level overview of the intended concepts and their interrelation.

5.1 Generic Graph Ontology in CASL

Fig. 23 shows the ontology of Route Graphs of a specific kind, given as parameter; cf. also Sect. 3.2 and Fig. 22. It starts with the declarations of the basic sorts and relations, and an axiom stating that no dangling edges are allowed.

[4] The formal specification of other parts of the ontology such as parts of the "Common Sense Ontology" or various spatial calculi (cf. Sect. 4.4) would also be quite interesting from the point of view of spatial cognition; we defer this to another paper.

```
Haskell
class Edge a where
    source :: a -> Node
    target :: a -> Node

data Path = forall a . (Eq a,Edge a) => ConPath [a]
data Route = forall a . (Eq a,Edge a) => ConRoute [a]

connected :: Edge a => [a] -> Bool
connected [] = True
connected ([_]) = True
connected (s1:s2:l) = target (s1) == source (s2) && connected (s2:l)

noBranches :: Edge a => [a] -> Bool
noBranches (l) = let sources = map source in atMostOnce (sources l)

isPath :: Edge a => [a] -> Bool
isPath = connected

isRoute :: (Eq a,Edge a) => [a] -> Bool
isRoute l = isPath l && noBranches(l) && atMostOnce(l)
```

Fig. 25. Haskell implementation of `Edge`

Sequences of Edges. For the relation of a graph to sequences of edges, the parameterized specification GenSequence is instantiated. It provides some basic relations (omitted here) such as *hasHead* and *hasTail* and some auxiliary functions for specification purposes, such as *sources*, yielding the sources of all edges in a graph as a sequence, or *freq* for counting the number of elements in a sequence. Note that parameterized specifications often use compound names which are to be instantiated, e.g. *Edge*[*Kind*] or *Sequence*[*Edge*[*Kind*]].

Paths and Routes. *Path*[*Kind*] is defined as a subsort of *Sequence*[*Edge*[*Kind*]] with the auxiliary predicate *connected*, then *Route*[*Kind*] as a subsort of *Path*[*Kind*] without duplicate branches or edges (this implies no cycles).

5.2 Design Specification in Casl

A Casl specification of Graph2 based on predefined basic data types such as List and NonUniqueEdgesGraph is presented in Fig. 24. It povides the same sorts as the ontological definition of GenGraph in Fig. 23 and is the first "design specification" in a software engineering sense. We can prove rather straightforwardly that this specification satisfies the requirements set out in Fig. 23.

5.3 Implementation of Inheritance in Haskell

We now turn to an operational implementation in the functional programming language Haskell and demonstrate some of the subtleties in the translation.

Fig. 25 shows the polymorphic implementation of the predicates `connected` and `noBranches` based on the type class `Edge`. The implementation of `connected` uses the same recursive definition as in Fig. 24. The existential data types `Path` and `Route` require type classes `Eq` and `Edge` as minimal context. All this, including constructor functions for `Path` and `Route` (not shown), can be implemented without knowledge about the actual structure of later instantiations.

The Route Graph specialisation of class `Edge` to class `Segment` is shown in Fig. 26. Note that providing special information for `entry`, `course` and `exit` is optional and yields a safe `Nothing` when e.g. an instance does not need an exit.

Haskell

```haskell
class Edge a => Segment a where
    entry   :: a -> Maybe Angle
    entry _ = Nothing::Maybe Angle
    course   :: a -> Maybe Course
    course _ = Nothing::Maybe Course
    exit   :: a -> Maybe Angle
    exit _ = Nothing::Maybe Angle

data SegmentIndoorD = SegInd {segI_source :: Node,
                              segI_entry  :: Angle,
                              segI_course :: Course,
                              segI_exit   :: Angle,
                              segI_target :: Node}

instance Edge SegmentIndoorD where
    source = segI_source
    target = segI_target

instance Segment SegmentIndoorD where
    entry  = Just . segI_entry
    course = Just . segI_course
    exit   = Just . segI_exit
```

Fig. 26. Haskell implementation of `SegmentIndoor`

For the instantiation to indoors navigation, the data type `SegmentIndoorsD` is defined with the constructor `SegInd` (with special components/selector functions); it instantiates the classes `Edge` and `Segment`. The subsorting relation of SEGMENT and EDGE in CASL is translated to a class hierarchy in Haskell, where the data type includes all the information needed for all classes of the hierarchy.

Functions in class `Segment` are only accessible for data stored as `Path` or `Route` if we add the context `Segment` to the existential data types. Thus, we have to use the lowest class in the hierarchy that provides access to all information we need when we introduce existential types to hold heterogenous data.

6 Conclusion

We have presented the generic concept of Route Graphs, modelled by an ontology, and its step by step formalisation. We hope that the introduction of a "light-weight" ontology first, restricted to (sub)classes and relations only, will appeal to those who are not so interested in formalisation, and provides a general overview to those who are. Such an ontology specification can be done rather quickly in the special LaTeX format, and ontology visualisation tools (e.g. [10]) can already be applied. Moreover, such an ontology is already suitable for the semantic interrelation of documents (cf. [13]), as e.g. in the repository for a joint interdisciplinary research project such as the SFB/TR8 "Spatial Cognition".

The subsequent formalisation of the Route Graph ontology in Casl will appeal to those who emphasise the need for complete formalisation of ontologies, as in the DOLCE approach [16, 17]; here the structuring concepts of Casl and the availability of verification tools are perhaps the primary advantage.

The development of an actual data structure from a top-level ontology in Casl is a new experiment in Software Engineering since this first (loose) requirement specification is tailored to ontologies and not to software. Nevertheless, we believe that the approach works quite well, and the process will hopefully be further automated as much as possible. In the context of cooperation among and integration between a large interdisciplinary diversity of research and application areas (such as AI, Cognitive Science, Linguistics, and Psychology) we see it as a definite advantage to have a common "language", viz. semantic description formalism, to start from, *the* central (joint) ontology. We hope that those who are only interested in the result, e.g. a data structure for interaction in the robotics domain, will be patient with this "interdisciplinary definitional overhead", and happy with the outcome. We expect tools for the (formalised) ontology and interfaces to the data structure, with appropriate operations to simulate access "as an ontology", to coexist peacefully in the running system.

Finally, we hope to have made a contribution to clarifying concepts and providing a data structure for human–robot and robot–robot interaction for the indoors scenario. In particular, the (indoors) Route Graph ontology shall serve as a basis for linguistic augmentation to enable a dialogue between user and wheelchair, as outlined in [3, 4].

The approach presented in this paper will have to be extended, at the "user level", by a relation to spatial calculi, e.g. for treating "right" and "left", and nested overlapping regions. Formalisation of these calculi in Casl, at the ontology level, should be rather straightforward. The challenge will be the choice and combination of suitable calculi, and the transition between them.

We are looking forward to other instantiations of the generic Route Graph ontology (and data structures) in application domains such as geo-information systems or location-based service scenarios.

Acknowledgement. We thank the other members of the Route Graph and Ontology Working Groups of the SFB/TR 8, particularly John Bateman, Scott Farrar, Achim Mahnke, Reinhard Moratz, Hui Shi, and Tilman Vierhuff.

References

1. Werner, S., Krieg-Brückner, B., Herrmann, T.: Modelling navigational knowledge by route graphs. In Freksa, C., Habel, C., Wender, K., eds.: Spatial Cognition II. Number 1849 in LNAI. Springer (2000) 295–317
2. Bateman, J., Farrar, S.: Modelling models of robot navigation using formal spatial ontology. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., Barkowsky, T., eds.: Spatial Cognition IV – Reasoning, Action, Interaction. Number 3343 in LNAI. Springer (2004) 366–389
3. Krieg-Brückner, B., Shi, H., Ross, R.J.: A safe and robust approach to shared control via dialogue. Journal of Software **15(12)** (2004) 1744–1755
4. Ross, R., Shi, H., Vierhuff, T., Krieg-Brückner, B., Bateman, J.: Towards dialogue-based shared control of navigating robots. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., Barkowsky, T., eds.: Spatial Cognition IV – Reasoning, Action, Interaction. Number 3343 in LNAI. Springer (2004) 478–499
5. Krieg-Brückner, B., Röfer, T., Carmesin, H.O., Müller, R.: A taxonomy of spatial knowledge for navigation and its application to the bremen autonomous wheelchair. In Freksa, C., Habel, C., Wender, K., eds.: Spatial Cognition. Number 1404 in LNAI. Springer (1998) 373–397
6. Uschold, M., Grüninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review **11** (1996) 93–155
7. Gómez-Pérez, A., Benjamins, V.R., eds.: Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings. In Gómez-Pérez, A., Benjamins, V.R., eds.: EKAW. Volume 2473 of Lecture Notes in Computer Science., Springer (2002)
8. Corcho, O., Gómez-Pérez, A.: A roadmap to ontology specification languages. In: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, Springer (2000) 80–96
9. Bechhofer, S., van Hermelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: W3C: OWL Web Ontology Language – Reference. W3C Recommendation `http://www.w3.org/TR/owl-ref/` (2004)
10. (`http://www.informatik.uni-bremen.de/agbkb/publikationen/softlibrary_e.htm`)
11. (`http://www.informatik.uni-bremen.de/~davinci/`)
12. Krieg-Brückner, B., Hutter, D., Lindow, A., Lüth, C., Mahnke, A., Melis, E., Meier, P., Poetzsch-Heffter, A., Roggenbach, M., Russell, G., Smaus, J.G., Wirsing, M.: Multimedia instruction in safe and secure systems. In: Recent Trends in Algebraic Development Techniques. Volume 2755 of Lecture Notes in Computer Science., Springer (2003) 82–117
13. Krieg-Brückner, B., Lindow, A., Lüth, C., Mahnke, A., Russell, G.: Semantic interrelation of documents via an ontology. In Engels, G., Seehusen, S., eds.: DeLFI 2004, Tagungsband der 2. e-Learning Fachtagung Informatik. Volume P-52 of Lecture Notes in Informatics., Springer (2004) 271–282
14. Astesiano, E., Bidoit, M., Krieg-Brückner, B., Kirchner, H., Mosses, P.D., Sannella, D., Tarlecki, A.: CASL – the common algebraic specification language. Theoretical Computer Science **286** (2002) 153–196 `www.cofi.info`.
15. Mosses, P.D., ed.: CASL Reference Manual. Volume 2960 of Lecture Notes in Computer Science. Springer (2004)
16. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with DOLCE. [7] 166–181

17. Lüttich, K., Mossakowski, T.: Specification of ontologies in CASL. In Varzi, A.C., Vieu, L., eds.: Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004). Volume 114 of Frontiers in Artificial Intelligence and Applications., IOS-Press (2004) 140–150
18. Lüttich, K., Krieg-Brückner, B., Mossakowski, T.: Ontologies for the semantic web in CASL. 17th Int. Workshop on Algebraic Development Techniques (2004) (to appear).
19. Wallgrün, J.O.: Autonomous construction of hierarchical voronoi-based route graph representations. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., Barkowsky, T., eds.: Spatial Cognition IV – Reasoning, Action, Interaction. Number 3343 in LNAI. Springer (2004) 413–433
20. Randell, D., Cui, Z., Cohn, A.: A spatial logic based on regions and connection. In: Proceedings KR-92, San Mateo, Morgan Kaufmann (1992) 165–176
21. Moratz, R., Wallgrün, J.O.: Spatial reasoning about relative orientation and distance for robot exploration. In Kuhn, W., Worboys, M., Timpf, S., eds.: Spatial Information Theory: Foundations of Geographic Information Science. Conference on Spatial Information Theory (COSIT). Lecture Notes in Computer Science, Springer (2003) 61–74
22. Frese, U., Larsson, P., Duckett, T.: A multigrid algorithm for simultaneous localization and mapping. IEEE Transactions on Robotics (2004) (to appear).
23. Frese, U.: A discussion of simultaneous localization and mapping. Autonomous Robots (2004) (submitted).
24. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic. Springer Verlag (2002)
25. Weidenbach, C., Brahm, U., Hillenbrand, T., Keen, E., Theobalt, C., Topic, D.: SPASS version 2.0. In Voronkov, A., ed.: Automated Deduction – CADE-18. Volume 2392 of Lecture Notes in Computer Science., Springer (2002) 275–279
26. Autexier, S., Hutter, D., Mossakowski, T., Schairer, A.: The development graph manager MAYA (system description). In Kirchner, H., Reingeissen, C., eds.: Algebraic Methodology and Software Technology, 2002. Volume 2422 of Lecture Notes in Computer Science. Springer (2002) 495–502

Robot Navigation based on the Mapping of Coarse Qualitative Route Descriptions to Route Graphs

Christian Mandel*, Udo Frese†, Thomas Röfer‡
Department of Computer Science
Universität Bremen
P.O.Box 330 440
Germany - 28334 Bremen
Email: {*cman, †ufrese, ‡roefer}@informatik.uni-bremen.de

***Abstract*— This paper describes the use of natural language route descriptions in the mobile robot navigation domain. Guided by corpus analysis and earlier work on coarse qualitative route descriptions, we decompose instructions given by humans into sequences of imprecise route segment descriptions. By applying fuzzy rules for the involved spatial relations and actions, we construct a search tree that can be searched in a depth-first branch-and-bound manner for the most probable goal configuration w.r.t. the global workspace knowledge of the robot. The applicability of our approach is shown by a real-world experiment where an operator instructs his automated wheelchair to navigate in an office-like environment.**

I. Introduction

Since robots have found their way from sealed work-stations in factories to populated places such as museum halls, railway stations, or hospitals, it is clear that sharing an environment with human beings requires suitable means of communication between these two groups. Apart from safety issues, appropriate communication becomes even more important if man and machine collaborate in order to fulfil a task. Service robots serve as a good example for these needs.

In our case, the autonomous wheelchair "Rolland" [1] is intended to assist its operator in various navigation tasks. The most natural application scenario allows the wheelchair-bound person to ask the device to go to a desired goal location e.g. "Please bring me to the kitchen.". In the case that the system's map of the environment has never been annotated with the symbol "kitchen", a clarifying utterance such as "Go down the corridor and take the second door to the left." could recover from potential failure. The work presented in this paper applies to task descriptions of the latter form.

We begin in section II with a brief overview of approaches to communicate qualitative task descriptions to a robot. Section III continues with the description of our experimental platform and accounts for the utilized prerequisites from the robotics domain, e.g. data structures for global world knowledge, self localization, and the local navigation method applied. In section IV we present the formalization of our so-called *Coarse Route Descriptions* (CRD). As an extension to the work in [2], they are based on the analysis of corpora which were aquired during an experiment where about 27 subjects had to give route descriptions to a designated goal, while operating the autonomous wheelchair Rolland [3]. In section V we present our approach to mapping CRDs to the global environment map. Commencing with the global pose of the system, we apply fuzzy rules for the spatial relations that describe the involved sub-goals, as well as subsequent actions which mostly comprise reorientations along the route to the goal. In section VI we conclude with the presentation of experimental results that where acquired by the execution of an a priori given coarse qualitative route description.

II. Related Work

Current research in human-robot interaction (HRI) comprises a vast spectrum of applied interface techniques. Thrun gives in [4] a broad survey of the state of the art and divides current approaches into direct and indirect interaction. Whereas direct communication means that the robot and its operator are able of initiating a dialogue act, indirect communication solely allows the operator to give commands, possibly followed by a response from the robot. Another distinction of HRI is given in [5], where the authors contrast the so-called "Front-End" approach that completely specifies the task to be performed by the robot against an incremental approach. The latter one is based on the decomposition of task descriptions into elementary actions, thus allowing for dynamic changes in the execution phase.

From the perspective of instructed robot navigation, different modalities in HRI have been established. Kyriacou et.al. describe in [6] a verbally instructed robot that executes a given route description in an artificial miniature town. Encouraged by corpora analysis they transfer a given route description into a list of action chunks accompanied by spatial relations and landmarks. During the execution phase the system builds on computer vision, e.g. for the recognition of referenced road features. We can anticipate that this approach differs from our work in that it solely relies on local sensor maps instead of exploiting prior available global knowledge. Hence, the comparison of both approaches is a good example for the contrasting incremental, resp. front-end approaches.

Unlike natural language driven interfaces, Chronis et.al. use sketched route maps for communicating with a mobile robot [7]. Hand-drawn landmarks that are derived from closed polygons are enriched by spatial relations that indicate their position w.r.t. the sketched route. The resulting *Qualitative*

Landmark States (QLS) are finally matched against the sensorial input in order to keep the robot on track.

III. Experimental Platform and Prerequisites from the Robotics Domain

Our experimental platform Rolland III is the battery-powered wheelchair Meyra Champ 1.594. In contrast to its predecessor that was equipped with a ring of sonar sensors, and later on with a single backward facing laser range finder [8], Rolland III is equipped with two laser scanners mounted at ground level, which allow for scanning beneath the feet of a human operator. As an additional sensor device, the system provides two incremental encoders which measure the rotational velocity of two independently actuated wheels. Please note that the provided information needed for dead reckoning is highly imprecise due to variable wheel diameter and slippage.

A. Local Navigation and Obstacle Avoidance

The cruicial factor for the applicability of a mobile robot system in a real world scenario is its ability to deal with dynamic and unforeseen obstacles. To do so, almost all approaches use periodic sensor measurements in order to maintain a map of the vicinity of the robot. Using this local map, they either compute geometric paths or apply reactive behaviours in order to reach a locally defined goal while avoiding perceived obstacles. Common examples are the Dynamic Window Approach (DWA) [9], the Virtual Force Field Method (VFFM) [10], Nearness Diagram Navigation (NDN) [11] and many others.

The basic DWA, as an example for path planning algorithms, turned out to be not suitable for our needs, because it only checks single arcs with fixed curvature in every cycle of computation. Although this approach is sufficient for circular robots being able to turn around their midpoint while entering a narrow passage, it does not model a necessary haul-off movement needed by vehicles turning around the center of their rear axle. A similar problem arises when employing a purely behaviour-based approach like the VFFM. Here the robot is modeled as a particle influenced by the sum of repulsive forces from the obstacle points and an attractive force starting from the desired goal. It is obvious that one cannot achieve shunting behaviours using this technique. Minguez et al. proposed a method for incorporating the kinematics and the shape of a robot in behaviour-based techniques [12]. The authors exemplified their approach by applying the VFFM and NDN in their so-called Ego-KinoDynamic Space, and thereby explicitly taking account for the contour and the kinematic properties of the used robot. Despite this refinement, the overall algorithm still looks only one curve ahead, leaving the problem of haul-off movements untreated.

Starting from these insights we decided to employ a geometric path planner using cubic Bezier curves, since they are able to connect two given points while accounting for a desired curve progression and for directional requirements in the start point and end point. Considering the work of Hwang et al. [13] which gives a broad overview on approaches using this type of curve, we will now sketch our basic algorithm. Given the current pose of the wheelchair $startPose = (x_s, y_s, \theta_s)$ and the desired target $goalPose = (x_g, y_g, \theta_g)$, we search the space of cubic Bezier curves for paths that

i) connect $\vec{p}_0 = (x_s, y_s)$ with $\vec{p}_3 = (x_g, y_g)$,
ii) are smoothly aligned with θ_s in $\vec{p}_0$ and with θ_g in $\vec{p}_3$,
iii) are obstacle free in the sense that a contour of the robot shifted tangentially along the path does not intersect with any obstacle point from a given occupancy grid OG.

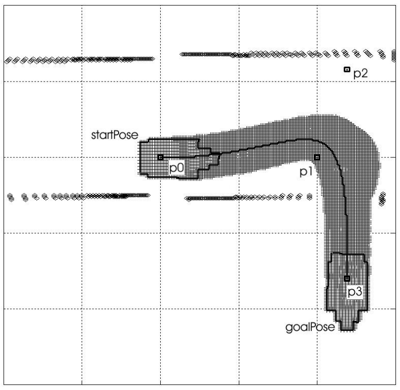


Fig. 1. Occupancy grid including cubic Bezier curve-based path, connecting startPose and goalPose. Control points $\vec{p}_1$ and $\vec{p}_2$ are located on straight lines that i) pass through $\vec{p}_0$ resp. $\vec{p}_3$ and ii) are aligned to the orientation of startPose resp. goalPose.

Equation (1) describes a cubic Bezier curve, connecting the points $\vec{p}_0$ and $\vec{p}_3$, at a given arc length $t \in [0..1]$. In order to unequivocally determine the characteristics of the curve, we still have to chose the control points $\vec{p}_1$ and $\vec{p}_2$ such that we fulfil requirements ii) and iii).

$$\begin{aligned} \vec{p}(t) &= \vec{a}t^3 + \vec{b}t^2 + \vec{c}t + \vec{p}_0,\ t \in [0..1] \\ \text{with } \vec{c} &= 3(\vec{p}_1 - \vec{p}_0), \\ \vec{b} &= 3(\vec{p}_2 - \vec{p}_1) - \vec{c}, \\ \vec{a} &= \vec{p}_3 - \vec{p}_0 - \vec{b} - \vec{c} \end{aligned} \tag{1}$$

The computation of the free parameters $\vec{p}_1$ and $\vec{p}_2$, as can be seen in (2), spans the search space over the cubic Bezier curves, whose solution is intended to solve our path planing problem.

$$\begin{aligned} \vec{p}_1(l_1) &= \vec{p}_0 + l_1\overrightarrow{(\cos(\theta_s), \sin(\theta_s))},\ l_1max > l_1 > 0 \\ \vec{p}_2(l_2) &= \vec{p}_3 - l_2\overrightarrow{(\cos(\theta_g), \sin(\theta_g))},\ l_2max > l_2 > 0 \end{aligned} \tag{2}$$

Fig. 1 illustrates the result of a single path planning cycle. It shows the occupancy grid including the integral of former sensor measurements along with the current state of the robot $startPose$ and the desired target $goalPose$. The solid drawn cubic Bezier curve has been chosen as a solution in fulfillment

TABLE I

EBNF FORMALIZATION OF COARSE ROUTE DESCRIPTIONS. FOR THE SAKE OF COMPACTNESS WE OMIT THE DEFINITION OF CURSIVE PRINTED NON-TERMINALS.

<CoarseRouteDescription>	::=	{ <CoarseRouteSegmentDescription> }
<CoarseRouteSegmentDescription>	::=	(<ControllerOnRouteSegment>, <RouterOnRouteSegment>, <ActionAtEndOfRouteSegment>)
<ControllerOnRouteSegment>	::=	{ (<SpatialRelation> <AttributedPlace> | <DistancePredicate>) }
<RouterOnRouteSegment>	::=	{ (<SpatialRelation> <AttributedPlace> }
<ActionAtEndOfRouteSegment>	::=	<TurnAction> | "Stop"
<SpatialRelation>	::=	"After"^{3} | "Along"^{2} | "Between"^{2} | <NaturalLanguageDirection>$^{2\|3}$ | "OutOf"^{3} | "Through"^{3} | "Until"^{2}
<NaturalLanguageDirection>	::=	<NatLangDir::2Clock> | <NatLangDir::4Clock> | <NatLangDir::8Clock> | <NatLangdir::12Clock>
<NatLangDir::2Clock>	::=	"Left" | "Right"
<NatLangDir::4Clock>	::=	"Left" | "Front" | "Right" | "Back"
<NatLangDir::8Clock>	::=	"Left" | "FrontLeft" | "Front" | "FrontRight" | "Right" | "BackRight" | "Back" | "BackLeft"
<NatLangDir::12Clock>	::=	"9oClock" | "10oClock" | "11oClock" | "12oClock" | "1oClock" | "2oClock" | "3oClock" | "4oClock" "5oClock" | "6oClock" | "7oClock" | "8oClock"
<AttributedPlace>	::=	(<*Place*>, <*AttributeType*>, <*AttributeValue*>)
<DistancePredicate>	::=	<*NumericalValue*> (<*LengthUnit*> | <*TimeUnit*>)
<TurnAction>	::=	<NaturalLanguageDirection>

of requirements i) - iii) that minimizes the time of travel. The upper velocity-bound of the robot at point $\vec{p}(t)$ is therefore determined by the minimal distance between the robot-contour tangentially located at $\vec{p}(t)$ to any obstacle-point, and the curvature $c(t)$.

B. Global Localization

As an intermediate step before using already available topological information for localization, a Monte-Carlo-Localization method was implemented that is based on the self-locator used by the German RoboCup Team [14]. To establish hypotheses of the current location of the wheelchair, particles are drawn from a pre-computed table that indexes the global environment by the area of the scan perceived from a certain position. In addition, only distance measurements resulting from flat surfaces i.e. from segmented lines are used to determine the current position. Thereby, persons standing around are ignored. The number of particles drawn from observations depends on how good the actual sensor measurements match the expected measurements from the positions of particles [15].

C. RouteGraph as Data Structure for Global World Knowledge

In order to solve high level tasks such as the interpretation of coarse qualitative route descriptions, one needs an appropriate global data structure that meets a list of requirements. Beside the adequate representation of navigable space, it has to be extendible by annotations that describe landmarks of interest. These annotations could be given either by humans or by automatic feature detectors. After all, it has to provide an interface that supplies basic spatial queries as a foundation for the evaluation of more elaborated spatial tasks. This interface is described in section V.

In our work we apply a graph structured representation, i.e. the *Route Graph*, to do the job sketched above. In the route graph taxonomy, broadly described in [16] and [17], a route graph representing the whole workspace of the robot consists of several *Route Graph Layers* each of a different *Kind*. At several *Places* which represent distinguishable robot poses, the route graph layers are connected via *Transitions*, depicting a correspondance between places inside different kinds of map representation. On the other hand, places can be connected by *Route Segments* in a single route graph layer, stating that the robot can go from one place to the other. Concatenation of these route segments results in *Routes*, describing complex navigation tasks.

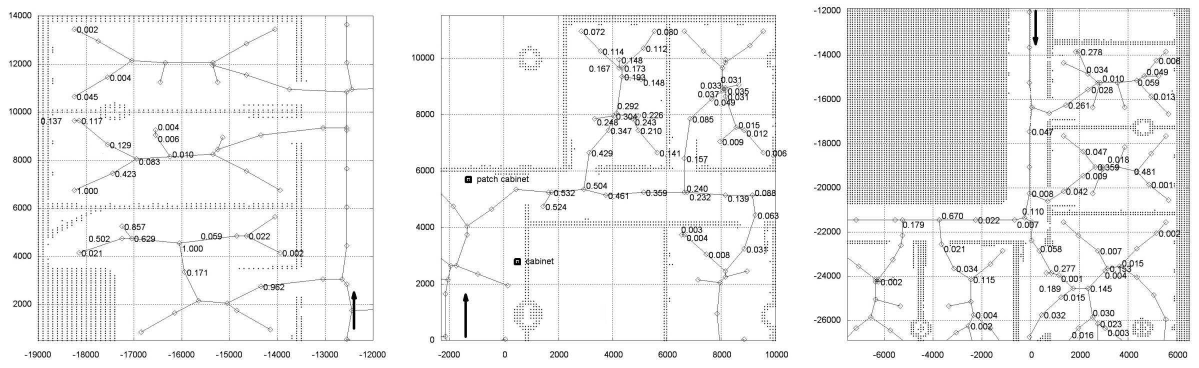
Our ongoing work is mainly focused on the topological layer of the route graph e.g. the analysis of its use in self localization tasks. The topological layer describes a network of places and route segments with maximal clearance to the surrounding obstacles, based on Voronoi diagrams. Our framework implemented so far is able to compute this description either from local sensor-based grid maps, or from a pre-existing global grid map derived from a CAD blueprint.

IV. FORMALIZATION OF COARSE ROUTE DESCRIPTIONS GIVEN VIA NATURAL LANGUAGE

In this section we lay out the concept of interpreting coarse qualitative route descriptions, usually given by humans to other agents. The main idea, borrowed from [2], states that the concerned route descriptions can be segmented into four categories: starting-point, path/progression, reorientation and goal. Following this insight, the authors develop a route description as a sequence of tupels of the following form:

$$< \ [\, \{ \text{ controlmarks } \} \ \textbf{router} \ ] \ \textit{reorientation} \ > \qquad (3)$$

Starting at the current pose of the agent, controlmarks describe distinctive landmarks along a route segment without directional changes. By reaching a router that depicts a place where a directional change may occur, a reorientation stands for the directional instruction that aligns the agent to the new route segment. The following route segments are encoded in the same way with the only difference that the the final router represents the goal of the whole route description. Before we describe miscellaneous enhancements of this formalization, we take a brief look at some corpus examples in order to justify the overall approach. The corpus results from an empirical study, where about 27 subjects had to give in-advance route descriptions towards a designated goal, while operating the



(a) The places to the *front left*. (b) Pass *through* the cabinet and the patch cabinet. (c) Go *until* the 4th. junction to the left.

Fig. 2. Interpretation of spatial relations “NaturalLanguageDirection”, “Through” and “Until” via fuzzy functions. The current pose of the agent i.e. the ego is depicted by a bold arrow. Prominent places acting as relatum are labeled by their fuzzy ratings.

autonomous wheelchair Rolland. See [3] and [18] for a brief discussion of this study. In both of the following examples that are translated from German to English, the subjects describe the way from their current pose to the so-called *Stugaraum.*

i) *...I have to turn around...once...180 degrees*[1]*...***drive through the door***...turn right...***and pass through the connecting door***...then turn right once again...*then a quite long time straight ahead...pass the elevators...**pass the main stairs***...and then we arrive at the Stugaraum...*
ii) *...we turn once...180 degrees...*leave the room...**on the corridor** *we turn 90 degrees to the left...*follow the corridor...through the glass door...**after the glass door we take the first junction***...turn 90 dregrees to the left...*continue straight ahead...through a glass door...down the corridor...through a second glass door...**and on the left side there is the Stugraum**...

Considering the *italic* printed reorientation blocks, the **bold** printed routers and normal typeset controlmarks, one can verify the basic form of definition 3 against the given examples. In spite of this achievement, the inner structure of each single building block remains undeveloped. For this reason we have extended the basic formalization of coarse route descriptions. A first look on the resulting EBNF in table I reveals the overall structure.

As in definition 3, a CoarseRouteDescription is still defined as an ordered sequence of CoarseRouteSegmentDescription, each of which now comprises a ControllerOnRouteSegment, a RouterOnRouteSegment and an ActionAtEndOfRouteSegment. So that the controller and the router are capable to represent a landmark along, respectively at the end of the route segment, they are now defined via a SpatialRelation and an AttributedPlace. Whereas the controller can characterize a route segment also via a DistancePredicate e.g. “a quite long time straight ahead”, the expressiveness of controllers and routers basically lies in the combination of spatial relations and attributed places. A simple example for such a combination is “through a glass door”, in which the spatial relation “through” is a three-valued relation. As the first argument it takes the current pose of the agent called *ego*. The attributed place “glass door” forms the second argument and is called *relatum* in an abstract way. The last argument *referent* is the hypothetical pose that is reached when moving from ego “through” the relatum. The EBNF in table I gives an open list of spatial relations whose order is denoted by an attached superscript.

A special spatial relation is given by NaturalLanguageDirection. This frequently appearing relation can either be used as a binary relation e.g. “the door to my left”, or as a ternary relation such as “the door left to the elevator”. To take account for the fact that humans use directions in different levels of granularity, four kinds of NaturalLanguageDirection have been defined, each of a different resolution. They range from NatLangDir::2Clock which solely differentiates Left and Right to NatLangDir::12Clock in analogy to the directions pointed to by a watch-hand at clock hours.

Completing the formalization of coarse route descriptions, an attributed place represents a landmark that is referenced via a spatial relation. To stay general, the type and the domain of an attribute are left unspecified. A typical example that is used in our implementation (cf. sec. V) is “the third junction to the left”, representing a place that has the property of being the third junction with a possible veer to the left w.r.t. a given ego.

V. Processing of Coarse Route Descriptions

This section starts with a specification of fuzzy functions that correspond to the spatial relations that most frequently appeared in the corpora. We then proceed with the development of an algorithm that inputs a formalized *CRD* (cf. sec. IV) representing the current task description, a global route graph (cf. sec. III-C) which holds the available world knowledge, and the global pose of the system. By evaluating

[1]Note that this example is somewhat unusual, due to the precisely given turning angles. The translation of these numerical values into the proper spatial relations, e.g. 180 degrees $\rightsquigarrow$ NatLangDir::Back, is part of the semantic speech analysis and outside the scope of this work.

each fuzzy function that corresponds to the consecutively appearing spatial relations in the CRD, our algorithm builds up a search tree that is searched in a depth-first branch-and-bound manner to determine the most probable goal pose of the route description.

A. Spatial Relations as Fuzzy Functions

Our approach introduced here treats the task of evaluating spatial relations as a judgement of how well at least two given points out of the global workspace description can be correlated via the spatial relation to be analyzed. We chose fuzzy functions mapping to the domain $[0..1]$ because of the intrinsic coarse nature of natural language information.

1) Natural Language Direction: When humans use directional prepositions in natural language, they use them either as binary or ternary spatial relation. In this example we describe the binary case, in which the agent references a relatum from its current pose ego. As described in section IV, one uses natural language directions within different levels of granularity. Our illustration in Fig. 2(a) addresses the direction "FrontLeft" out of an eight-valued set of directions i.e. <NatLangDir::8Clock> (cf. table I). We start by computing the angle α between i) the straight line that connects ego with relatum and ii) the vector that is based in the position of ego and aligned to the current heading of ego. Taking the most compatible angle β for the given direction "FrontLeft", which is $pi/4$ w.r.t. the eight-valued level of granularity, and the normalizing constant c, we now compute the fuzzy rating of the spatial relation "FrontLeft" as can be seen in (4).

$$FR_{NatLangDir}(ego, relatum) = e^{-\frac{1}{2}\left(\frac{|\alpha-\beta|}{c}\right)^2} \tag{4}$$

2) Passing Through Given Landmarks: The ternary spatial relation "Through" takes as its first argument the current pose of the agent called ego. In our example the second argument is given by the two landmarks $lm_1 = cabinet$ and $lm_2 = patch\ cabinet$, both labeled as relatum. The third argument referent is the global place we want to reach by passing from ego through relatum. This situation is illustrated in Fig. 2(b). Starting from ego depicted by the bold arrow, we choose an arbitrary relatum, that is to be evaluated by the spatial relation, in the neighborhood of ego. In a first step we apply a standard graph search algorithm to compute the route ρ from the closest place nearby ego to relatum. If ρ does not exist, we judge relatum with the lowest possible value 0. Otherwise we continue with the computation of a circular region χ with diameter $d_\chi = \|\vec{lm_2} - \vec{lm_1}\|$ such that the centers of the landmarks lie on χ's circumference. The mentioned circle χ now segments ρ into the parts before, inside and after χ. Taking the length λ of ρ after χ, and the normalizing constant c, we now compute the fuzzy rating of the spatial relation "Through" as can be seen in (5).

$$FR_{Through}(ego, referent, relatum) = e^{-\frac{\lambda}{c}} \tag{5}$$

The fact that fuzzy ratings decrease for places that are farther away from χ models the typical nature of human route descriptions in that single parts apply to the direct neighborhood of referenced landmarks.

3) Going Until Indexed Junction: The binary spatial relation "Until" takes as its first argument the current pose of the agent called ego. In our example the second argument relatum is given by the place until which we want to go from ego, i.e. the $i = 4_{th}$ place with a junction to the left. In order to look for an indexed place with a given attribute, we first have to define the fuzzy rating for the existence of that attribute in a given place $\emptyset$. Therefore we model a junction to a given natural language direction in analogy to the binary direction relation in section V-A.1, however with a different computation of the angle α. In this case the angle α is computed as the angle between two route segments sharing the common place $\emptyset$. We now start with the computation of the route ρ from the closest place nearby ego to relatum. Interpreting ρ as an ordered set of places $\{p_0, p_1, ..., p_n\}$, we compute the ordered set of fuzzy ratings $FR(\rho) = \{fr_1, fr_2, ..., fr_n\}$, which represent the judgements that there exists a junction to the left in the corresponding places p_i. Considering the $j = \binom{n}{i}$ possible combinations to find in p_n the i_{th} place with a junction to the left, we can now compute $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_j\}$ as the $\binom{n}{i}$-subsets of $FR(\rho)$ (cf. [19]), yielding the overall fuzzy rating for the spatial relation "Until" as can be seen in (6).

$$FR_{Until}(ego, relatum) = \sum_{k=1}^{j}\prod_{l=1}^{n}\left\{\begin{array}{r} fr_l{:}fr_l \in \mathcal{A}_k \\ 1 - fr_l{:}fr_l \notin \mathcal{A}_k \end{array}\right. \tag{6}$$

B. Evaluation of Coarse Route Descriptions via Search Trees

For the evaluation of coarse route descriptions, i.e. the calculation of the most likely target pose, we utilize a search tree the nodes of which represent fuzzy rated places that result from the evaluation of single CRD-elements. Fig. 3 illustrates the construction of the search tree. Considering a single node as a compound of a pose and its fuzzy rating or score, the root $p1$ is given by the initial pose of the agent along with the highest possible score 1. Taking the first spatial relation SR_1 that appears in the ordered list of coarse route segment descriptions, one chooses the first candidate pose $p2$

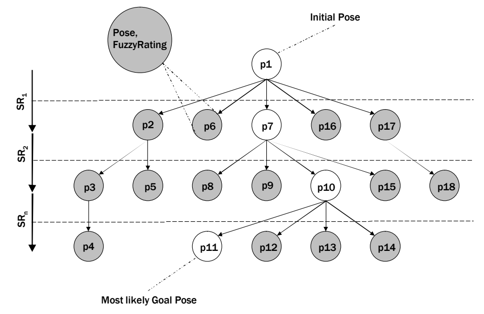


Fig. 3. Evaluation of a coarse route description via depth-first, branch-and-bound search.

that acts as the relatum for the fuzzy function associated with SR_1. After calculating $p2$'s own score, it is multiplied by the score of the parent node $p1$. This mechanism propagates the uncertainty of former fuzzy ratings down to lower levels of the search tree. The algorithm now recursively continues in a depth-first branch-and-bound manner with the evaluation of the following spatial relations. In order to keep the size of the search tree small, we apply two different bound-criteria. First we do not construct nodes from the evaluation of a single spatial relation if the resulting score is lower than a given threshold c_1. Second we stop the evaluation at a given node if its cumulative score is already lower than the highest score of a leaf resulting from the evaluation of the final spatial relation. In addition to these rules, we only address candidates for the relatum of the spatial relation to be analyzed that are closer to the current ego than a given distance c_2.

VI. Conclusion

In this work we have addressed the formalization of coarse qualitative route description in order to apply them to the mobile robot navigation domain. It has been shown that the involved spatial relations can be regarded as the key-elements in understanding route descriptions. By interpreting them with the help of fuzzy functions, we were able to derive the most probable target pose from a given route description. As an example of our real world experiments (cf. Fig. 4 for an illustration), the following instruction has been formalized, evaluated and executed in an office-like environment.

> ...go to the elevators area and continue until the first junction to the left...turn left there and pass through the cabinet and the patch cabinet...then continue until the second door to the left...turn left, go through the door and stop...

The findings presented so far bridge a gap between low-level speech recognition and ontological language understanding on the one hand and path planning for mobile robots on the other hand. Upcoming work has to focus on the investigation of further spatial relations in order to interpret the wide choice of natural language route descriptions.

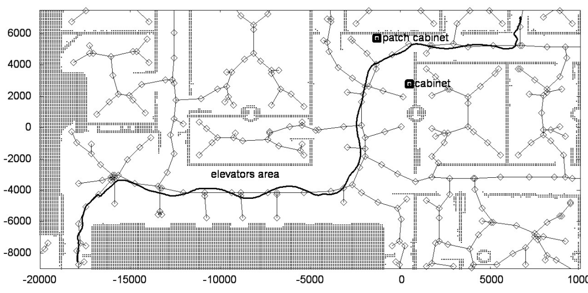


Fig. 4. Test run as estimated by the Monte Carlo Localizer: Starting at the corridor to the lower left, going via the elevators area, passing through the cabinets and finishing in the office to the upper right. Note that the depicted trajectory results from a real world experiment that has been conducted with an autonmomously moving robot, instructed by spoken route descriptions (cf. [20] for an illustrative video).

Acknowledgment

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) in context of the Sonderforschungsbereich/Transregio 8 *Spatial Cognition.*

References

[1] C. Mandel, K. Hübner, and T. Vierhuff, "Towards an autonomous wheelchair: Cognitive aspects in service robotics," in *Proceedings of Towards Autonomous Robotic Systems (TAROS)*, 2005.
[2] R. Müller, A. Lankenau, A. Musto, K. Stein, and A. Eisenkolb, "Coarse qualitative descriptions in robot navigation," in *Spatial Cognition II, Lecture Notes in Artificial Intelligence*, 2000.
[3] I1-[OntoSpace]. (2004) Human-robot interaction. SFB/TR8 Spatial Cognition, University of Bremen. [Online]. Available: http://134.102.58.154/human-robot.html
[4] S. Thrun, "Towards a framework for human-robot interaction," *Human Computer Interaction*, 2003.
[5] A. Knoll, B. Hildebrandt, and J. Zhang, "Instructing cooperating assembly robots through situated dialogues in natural language," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997.
[6] T. Kyriacou, G. Bugmann, and S. Lauria, "Vision-based urban navigation procedures for verbally instructed robots." [Online]. Available: citeseer.ist.psu.edu/651267.html
[7] G. Chronis and M. Skubic, "Robot navigation using qualitative landmark states from sketched route maps," in *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2004.
[8] A. Lankenau and T. Röfer, "A safe and versatile mobility assistant," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 29–37, 2001.
[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance, Tech. Rep. IAI-TR-95-13, 1 1995. [Online]. Available: citeseer.ist.psu.edu/fox97dynamic.html
[10] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, - 1989. [Online]. Available: citeseer.ist.psu.edu/article/borenstein90realtime.html
[11] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, 02 2004.
[12] ——, "The ego-kinodynamic space (ekds): Shape, kinematics and dynamics of the vehicle," in *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
[13] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
[14] T. Röfer and M. Jüngel, "Vision-based fast and reactive monte-carlo localization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.
[15] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modeled mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.
[16] B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, and R. Ross, "Specification of an ontology for route graphs," in *Spatial Cognition IV, Lecture Notes in Artificial Intelligence*, 2004.
[17] S. Werner, B. Krieg-Brückner, and T. Herrmann, "Modelling navigational knowledge by route graphs," in *Spatial Cognition II, Lecture Notes in Artificial Intelligence*, 2000.
[18] S. Hui and T. Tenbrink, "Telling rolland where to go: HRI dialogues on route navigation." in *Proc. Workshop on Spatial Language and Dialogue (5th Workshop on Language and Space)*, 2005.
[19] A. Nijenhuis and H. Wilf, *Combinatorial Algorithms.* Academic Press, 1978.
[20] A1-[RoboMap] and I3-[SharC]. (2006) Interpretation and execution of verbal route descriptions by the autonomous wheelchair rolland. SFB/TR8 Spatial Cognition, University of Bremen. [Online]. Available: http://www.informatik.uni-bremen.de/rolland/videos/Interpretation_of_Coarse_Route_Description_08_06_06.mpg

Interpreting Route Instructions as Qualitative Spatial Actions

Hui Shi, Christian Mandel, Robert J. Ross

Universität Bremen, Germany
{shi,cman,robertr}@informatik.uni-bremen.de

Abstract. In this paper we motivate the use of *qualitative spatial actions* as the fundamental unit in processing user route instructions. The spatial action model has been motivated by an analysis of empirical studies in human-robot interaction on the navigation task, and can be interpreted as a conceptual representation of the spatial action to be performed by the agent in their navigation space. Furthermore, we sketch out two distinct models of interpretation for these actions in cognitive robotics. In the first, the actions are related to a formalized *conceptual user modeling* of navigation space, while in the second the actions are interpreted as *fuzzy operations* on a vornoi graph. Moreover, we show how this action model allows us to better capture the points at which user route instructions become out of alignment with a robot's knowledge of the environment through a number of examples.

1 Introduction

Route navigation instructions allow one agent to instruct another to a particular location within their shared environment. While robotic agents in the near future may have access to extremely detailed environmental descriptions through technological application, e.g., GPS, a-priori mappings etc, the need for an artificial agent to be capable of processing route instructions remains an interesting research question for a number of reasons. Firstly, and arguably most importantly, cooperation with *naive* users in natural rather technical interactions, places onus on the agent to handle natural instructions to a location, e.g., route instructions, rather than forced goal selection through other means, e.g., hierarchical list selection. Furthermore, disparities between the robot's spatial representation and the ever-changing world are always a possibility, thus opening up the possibility of adding route instructions to goals which were not already known to the robotic agent.

The modeling and interpretation of route instructions has been addressed from a number of different research perspectives. In the spatial cognition and cognitive modeling communities, a great deal of effort has been applied to the analysis of route instructions as a reflection of a speaker's cognitive map (e.g., [26, 8, 28, 10]). While such approaches are certainly interesting in terms of providing insight into the nature of the spatial models that motivate route instructions as verbalised by users, they are often somewhat abstracted from the detail required for on-line computational analysis. On the other hand, some in the robotics community have attempted to process route instructions as procedural information without reference to any explicit spatial representation [13]. In practice however, concrete robotic systems use very detailed spatial representations and reasoning processes that operate at a finer level of granularity than those models proposed by the cognitive modeling community [19, 3, 24], but yet little work has been done to date on unifying such approaches with the cognitive spatial representation and language processing efforts.

In this paper we propose an abstraction model for the interpretation of route instructions so as to unify a number of approaches to route instruction modeling and reasoning that have already been considered in the literature. We begin in Section 2 by reviewing a number of empirical studies involving the presentation of route instructions to artificial communicating partners. Such studies, showing that route instructions are most frequently, but certainly not always, presented by a human partner as a series of actions to be performed by an agent in a navigation space, lead us to propose a *Qualitative Spatial Actions* (QSA) as the fundamental modeling unit of verbal route instructions as intended by users. We introduce the elements of this QSA model in Section 3, detailing its coverage and relationship to linguistic and conceptual knowledge within an interpretation model. While the categories and coverage of the QSA model are driven by empirical analysis, computational systems must provide suitable interpretations of qualitative spatial actions in context. In Section 4 and 5 we present two distinct models for the interpretation of qualitative spatial actions. In the first, in Section 4, QSAs are interpreted as constructive operators within a formalized Conceptual User Model based on a *Route Graph* [27, 9] augmented with spatial relations drawn from the *Double Cross Calculus* [7, 8]. In the second, in Section 5, QSAs are interpreted using *fuzzy functions* applied against a vornoi graph representation as already applied in a concrete robotics application. We explicitly compare and contrast the two interpretation models in Section 6, showing that while both have their particular advantages, both approaches are required to facilitate a range of querying and reasoning tasks on route instruction. Section 7 continues on this theme by identifying a number of areas where we see *interpretation disparities* between the two interpretation models discussed. Before concluding in Section 9, we relate the approaches proposed in this paper to other work on the interpretation of route instructions in Section 8.

2 Empirical Results of Route Description Studies

As indicated above, the proposed QSA model is based on an analysis of concrete route instruction examples as given by users to mobile robots. We have built upon the results of two separate corpora of empirical results for this analysis. The first of these corpora was that due to the Instruction Based Learning (IBL) project [13], while the second was collected from our own project investigations of interaction between users and a semi-autonomous wheelchair [18].

1. okay take your first right
2. and continue down the street
3. past Derry's
4. past Safeway
5. and your parking lot the car park will be on your right

Fig. 1. Short Route Instruction from the IBL Corpus

2.1 Corpus 1: Instruction Based Learning

The Instruction Based Learning (IBL) project [13, 2] attempted to construct robotic systems in which verbal commands including route instructions were interpreted into internal program code for execution by mobile robots. In designing the system, empirical studies were conducted to allow: (a) the construction of a program primitive set; and (b) the derivation of a domain model to allow construction of a language analysis grammar and the training of speech recognition software. Those studies, detailed by Bugmann in [2], are summarized here to elucidate the contents of the corpus.

In the studies, 24 subjects were instructed to give route instructions which were to be processed by a small remote-controlled robot situated within a *toy town*. The subjects were divided into three equi-sized groups. The first two of these groups were informed that their route instructions would actually be post-processed by a remote human operator who observes the environment through a camera situated on the *head* of the robot. Subjects, who were also told to use previously defined routes where possible, had to describe six distinct routes consisting of three short and three long routes. Whereas the first two groups of subjects produced unconstrained monologic speech, the third group was forced into simplified dialogues with an operator to produce smaller chunked route instructions – albeit to describe the same underlying route instructions as the first group.

The resulting corpus contains a total of 144 routes instructions. Figure 1 shows one of the short monologue style route description segmented by hand into major or minor clauses.

2.2 Corpus 2: Shared-Control Wheelchair Studies

The second corpus which we base our analysis on was obtained in a series of experiments which were conducted to investigate the nature of language used by humans in communicating with robots about spatial concepts, such as route navigation, spatial relations between objects [22, 6, 21, 18]. As with the IBL experiments, the reader is directed to the above sources for a detailed description of the experimental setup and results – here we simply identify the salient points of those studies for illustrative purposes. In particular it should be noted that while the experiments did involve a number of different robotic platforms, e.g., Sony Aibo dogs, pioneer robots, wheelchair, and interaction scenarios, e.g., scene descriptions, route instructions, map annotations, we focus here on the results

1. dann äh , muss ich mich jetzt umdrehen
   I must turn around
2. aus der Tür fahren
   drive out the door
3. äh , dann nach rechts mich drehen
   then turn to the right
4. dann , - ziemlich lange geradeaus
   straight ahead for a relatively long way
5. äh , rechts vorbei an den Fahrstühlen
   pass by the lifts
6. vorbei an den Haupttreppen
   pass by the stairs
7. ähm, und dann, – sind wir eigentlich schon am Stugaraum
   and then we should be at the student-union room

Fig. 2. Example Route Instruction from the Rolland Corpus

concerning the presentation of route instruction and directed movement commands to the semi-autonomous wheelchair.

The route navigation studies were performed using *Rolland*, a semi-autonomous wheelchair, which was situated in an office environment within a university. Over 40 subjects (a mix of native English and German speakers) participated in the two-phase experiments. In the first phase, users were situated in the wheelchair and asked to move around the partially known environment, *telling* the wheelchair about the locations of particular rooms within the environment. Having become more familiar with the environment and landmarks present in the environment, the second phase of the experiment then begins. In the second phase, users are asked to navigate Rolland to a destination through a verbal route description which makes use of some of the previously identified landmarks in the shared environment. Subsequent route instructions were then recorded by a positioned recorder on Rolland, and were later transcribed for analysis.

Figure 2 shows one sample route description given by a native German speaker to Rolland (with a descriptive rather than literal translation).

2.3 Discussion

From a quick scan of both route instruction examples we see that the route description is, to the most part, conveyed as a set of actions to be performed by the agent to achieve the goal of moving them from a start location to the goal. Thus, unsurprisingly, the route instructions are construed as something more akin to a plan rather than a static spatial description such as a scene description.

That said, the route instructions found in the corpus do of course move beyond well-behaved sequences of qualitatively or goal directed actions. In particular, we find the predictable occurrences of finer grained functional instructions, as well as a range of corrections, sanity checks, and other constraints, e.g.:

(1) a. turn about ninety degrees right – IBL: `U20_GB_EH_19`
b. oh sorry erm from the hospital – IBL: `U9_GC_HG_3`
c. if you go to the grand hotel you have gone too far – IBL: `U17_GB_MD_5`

Despite the presence of such utterances which may not at first seem applicable within an action driven modeling, it is our hypothesis that route instructions remain best modeled as actions with constraints which manipulate the location and movement of an agent within a navigation space. Furthermore, if this is the case we argue that a suitable and consistent approach to the modeling of route instructions as such actions should naturally lend itself to interpretations against distinct spatial representations. Working from these assumptions, we will sketch out the Qualitative Spatial Action model in the next section.

3 The Qualitative Spatial Actions Model

In the last section we noted that in empirical studies route descriptions manifest themselves in language as sequences of conditionals and declarative instructions where the primary process present seem to map quite neatly to spatial movements and manipulations. In this section we propose the *Qualitative Spatial Action Model* as one means of capturing such route instructions.

3.1 Model Details

Informally we can define the *Qualitative Spatial Action Model* as a sequence of abstract spatial actions. To formally define the model, we must first introduce the sets of elementary types which model spatial orientations and places:

– $\mathcal{O}$ defines the set of orientation relations. There are different levels of qualitative direction granularity [17]. In this work we combine the front/back and the left/right dichotomy and distinguish 8 meaningful disjoint orientation relations: *front*, *rightFront*, *right*, *rightBack*, *back*, *leftBack*, *left*, *leftFront*. o, o', o_1, etc. are used for orientations.
– $\mathcal{M}$ consists of all landmarks in the spatial environment considered. Generally, a landmark is some view a human can perceive in the environment. m, m', m_1, etc. are used for landmarks.

A Qualitative Spatial Action Model instance is then a sequence of spatial actions drawn from the set:

– *turn*(o): turn to a given orientation o of $\mathcal{O}$.
– *travel*(c): move to a place such that condition c holds.
– *stop*: stop any movement.
– *annotate*(m, c): relate landmark m according to condition c.

where condition c consists of a set of ego-centric relations defined by

1	*turn*(*back*)
2	*travel*({*via*(door})
3	*turn*(*right*)
4	
5	*travel*({*passBy*(lifts)})
6	*travel*({*passBy*(mainStairway)})
7	*annotate*(stugaRaum, {*ego_ori*(stugaRaum, *front*)})

Fig. 3. The representations of the route description

ego_ori : $\mathcal{M} \times \mathcal{O}$	the orientation between a *landmark* and the current position
via : $\mathcal{M}$	a *landmark* is on a travel path
passBy : $\mathcal{M}$	a *landmark* is on the left or on the right hand side of a travel path
passOn : $\mathcal{M} \times \mathcal{O}$	a specific case of the *passBy* relation, where the orientation of the *landmark* to the travel path is given.
until : $\mathcal{M}$	a landmark is at the end of a travel path.

Applying the model, the route description from the Rolland Corpus depicted in Figure 2 can be modeled as the series of actions in Table 3. Some important features to note here. (a) utterance 4 is not captured by our model since we do not yet have an approach to capturing extremely subjective features such as “a long way”, “not far”, and so forth. (b) Although most route instructions in both the IBL corpus and the Rolland corpus are given at the granularity level as shown in the examples, there are some exceptions. For example, a user may employ exclusively directions such as “right”, “left”, “straightforward”. Our action model is not suitable for this kind of route descriptions. (c) It is possible to extend the set of spatial relations with new ones, such as “along”, but in this paper we restrict the QSAM model to the above introduced relations, in order to discuss its interpretations.

4 Interpreting QSAM: Conceptual User Navigation Space

Knowledge that mental construals of space are schematized (cf. [25]) along with a systematic simplification of metric information, has led to the description of cognitively inspired mental space models, such as the mental space of navigation consisting of objects such as landmarks and paths (e.g., [20, 5]). Such models in turn have given rise to cognitively inspired formalised models of space which often attempt to make use of qualitative descriptions of navigation space [1, 4, 28, 11, 27]. Such models are interesting since they are expressive enough to represent most knowledge humans used to carry out specific navigation tasks, while, on the other hand, supporting an efficient way to predicate and explain navigation and spatial behaviors.

In the following we review one such model of users conceptual navigation space and show how the QSAM actions can be directly constructed into such a model for reasoning purposes.

4.1 The Double-Cross Calculus and the Route Grpah

This conceptual model combines the Double-Cross Calculus [7, 8, 28], with the Route Graph [27, 9] to provide an abstract topological representation of navigation with well defined spatial relations. The Double-Cross Calculus, put forward by Freksa, uses orientation information for qualitative spatial representation and reasoning. In this calculus, the concept *orientation grid* is introduced to represent qualitative orientation information. The grid is aligned to the orientation determined by two points in 2-dimensional space, the start point and the end point of a movement. Combining the front/back and the left/right dichotomy the Double-Cross Calculus can distinguish 15 meaningful disjoint orientation relations, see Fig. 4.1. There are six areas, seven on the lines, and two points. Moreover, the concepts with respect to route segments, i.e., *entry*, *exit* and *course*, are used to represent some positions, like *atEntry*, *onCourse*, *rightAtEntry*, etc. Note that we name all these positions, while in the original Double-Cross calculus each position is associated with a pair of numbers between 0 and 7.

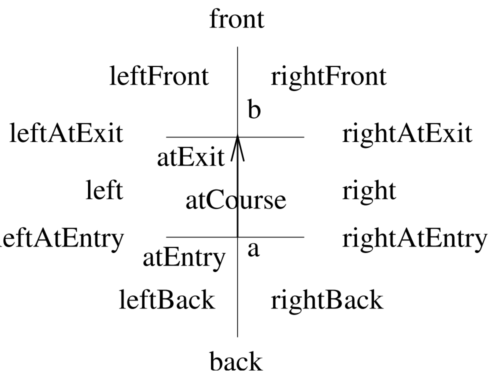


Fig. 4. Double-Cross orientation grids with 15 different positions

Route Graphs [27, 9] have been introduced as a general concept for navigation by various agents in a variety of scenarios. They can be used as metrical maps with sensory input and additional metrical computation models to control the navigation of robotic agents in the environment, or be used at the cognitive level to model abstractly humans' topological knowledge while they act in the space. The most important character of Route Graphs is that different routes can be integrated into a graph-like structure, in which the information concerning these routes are composed. The most important concepts of Route Graphs are *route* and *route segment*. A route is a sequence of route segments, while a route segment consists of an *entry*, an *exit* and a *course*. For example, an entry or an exit in a route graph at the metrical level can be an angle in degree with respect to a global 2-D geometric system, the course is just characterized by metrical data for length and width; while an entry/exit at the cognitive level may contain qualitative orientation information (e.g., to the left/right), the course is then the path between two reorientations.

4.2 The Conceptual User Model

Topological maps like Route Graphs are suitable for human navigation using landmarks and passages, but the representation of orientations (e.g, turn left or right) and spatial relations (e.g., on the left or right) remain unspecified. On the other hand, most approaches to representing qualitative spatial knowledge consider the relationship between spatially extended objects, without the supplement of the topological information (e.g., the best route, the same location). Thus, our solution has been to combine the Double-Cross Calculus and the Route Graph into a conceptual model of navigation space. The foundation of the model can be found in [10].

Formally then, the Conceptual User Model is represented as a triple of types, functions and relations. The set of types consists of:

- $\mathcal{O}$ defines the set of orientation relations. In this model we use 15 disjoint orientations defined by the Double-Cross Calculus (see Fig. 4.1).
- $\mathcal{M}$ consists of all landmarks in the spatial environment considered, the same as defined in the Qualitative Spatial Action Model.
- $\mathcal{P}$ consists of all places in the spatial environment considered. A place can be marked with a landmark, such as a room. a door. p, p', p_1, etc. are used for places.
- $\mathcal{V}$ defines the set of all directed connections between different places, denoted as vectors. We will use **ab** for the vector between a and b. A vector can be a corridor or a street (or part there of). A subset of $\mathcal{V}$ is specified as $\mathcal{S}$ route segments, and will be denoted using s, s', s_1, etc.

The most important functions are *entry*, *exit* and *course* which takes a route segment and defines the entry, the exit (as an orientation) and the course (as a vector) of the route segment.

The two elementary relations are *at* and *ori*. The relation $at : \mathcal{M} \times \mathcal{P}$ marks a place with a landmark. The relation $ori : \mathcal{V} \times \mathcal{P} \times \mathcal{O}$defines the orientation of a place with respect to a vector, such as $ori(\mathbf{bc}, p, rightFront)$ means p is on the right front of **bc**. Based on these two relations a set of auxiliary relations can be defined, for example,

on :	$\mathcal{V} \times \mathcal{P}$	a *place* is on a *vector*
left_of :	$\mathcal{V} \times \mathcal{P}$	a *place* is on the left of a vector
right_of :	$\mathcal{V} \times \mathcal{P}$	a *place* is on the right of a vector
begin_of :	$\mathcal{V} \times \mathcal{P}$	a *place* is the start place of a vector
end_of :	$\mathcal{V} \times \mathcal{P}$	a *place* is the end place of a vector

The following axioms assert that these relations can be defined through the two elementary relations.

$$\mathit{on}(\mathbf{ab}, p) \Leftrightarrow \mathit{ori}(\mathbf{ab}, p, \mathit{atCourse}) \tag{1}$$

$$\mathit{left\_of}(\mathbf{ab}, p) \Leftrightarrow \mathit{ori}(\mathbf{ab}, p, \mathit{left}) \tag{2}$$

$$\mathit{right\_of}(\mathbf{ab}, p) \Leftrightarrow \mathit{ori}(\mathbf{ab}, p, \mathit{right}) \tag{3}$$

$$\mathit{begin\_of}(\mathbf{ab}, p) \Leftrightarrow \mathit{ori}(\mathbf{ab}, p, \mathit{atEntry}) \tag{4}$$

$$\mathit{end\_of}(\mathbf{ab}, p) \Leftrightarrow \mathit{ori}(\mathbf{ab}, p, \mathit{atExit}) \tag{5}$$

4.3 Building Qualitative Route Representations

Given the above modeling of user navigation space, QSAM actions can be interpreted as constructive operators in building such Qualitative Route Represenations. Specifically, we interpret each *travel* action as a new route segment with appropriate relations as condition. Furthermore, each *turn* action then defines the entry of a route segment, while an *annotate* action is simply interpreted as defining a set of spatial relations on some route segment.

To illustrate, let us consider again the example route description which was presented as natural language in Figure 2, before being re-modeled in the QSAM in Figure 3. Figure 5 shows the resultant qualitative route represenation following the interpretation of QSAM primitives as constructive operators.

1	$\mathit{entry}(s_0) = \mathit{back}$
2	$\mathit{at}(\text{door}, x),\ \mathit{on}(\mathit{course}(s_0), x)$
3	$\mathit{entry}(s_1) = \mathit{right}$
4	
5	$\mathit{at}(\text{lifts}, y),\ \mathit{left\_of}(\mathit{course}(s_1), y) \vee \mathit{right\_of}(\mathit{course}(s_1), y)$
6	$\mathit{entry}(s_2) = \mathit{front}$
	$\mathit{at}(\text{mainStairs}, z),\ \mathit{left\_of}(\mathit{course}(s_2), z) \vee \mathit{right\_of}(\mathit{course}(s_2), z)$
7	$\mathit{at}(\text{stugaRaum}, u),\ \mathit{ori}(\mathit{course}(s_2), u, \mathit{front})$

Fig. 5. The representation of the sample route description

As defined, a route is a sequence of route segments. Suppose $\langle s_0, s_1, \cdots, s_n \rangle$ is a route, then following constrains should be satisfied, where $0 \leq i \leq n$ and $0 \leq j \leq n-1$:

- $\mathit{entry}(s_i),\ \mathit{exit}(s_i) \in \mathcal{O}$
- $\exists \mathbf{pq} \in \mathcal{V}.\mathit{course}(s_i) = \mathbf{pq}$
- $\mathit{course}(s_j) = \mathbf{p_1 p_2} \wedge \mathit{course}(s_{j+1}) = \mathbf{p_3 p_4} \rightarrow p_2 = p_3$

5 Interpreting QSAM: Fuzzy Operations

In this section we introduce an approach which enables a mobile robot to directly interpret and execute a given route description modeled as a sequence of qualitative spatial actions. In order to make such an interpretation against a global environment map, we develop and apply a number of fuzzy functions which provide interpretations of spatial relations against a vornoi graph based structure. These functions then allow us to evaluate interpretation hypotheses for individual sub-goals.

5.1 Spatial Relations as Fuzzy Functions

We begin by establishing a set of fuzzy functions in order to make the spatial relations established earlier algorithmicly useful. We do so by treating the task of evaluating spatial relations as a judgment of how well at least two given points out of the global workspace description (cf. Route Graphs in section 4.1) can be correlated via the spatial relation to be analyzed. In the following, we define a selection of the ego-centric relations used in the defintion of Qualitative Spatial Actions (See Section 3), i.e., *ego_ori*, *via* and *until*. While this does not cover all possible relations, we believe that those definitions below can provide a basis for a number of other relations.

Ego-centric orientation. When humans use directional prepositions in natural language, they often use them from their current point of view. In this example we describe the binary relation *ego_ori*, in which the agent references a relatum (a landmark in our case) from their current pose ego. We now address the binary direction *rightFront* and start by computing the angle α between (a) the straight line that connects ego with relatum and (b) the vector that is based in the position of ego and aligned to the current heading of ego. Taking the most compatible angle β for the given direction *rightFront*, which is $-\pi/4$ w.r.t. the eight-valued level of granularity, and the normalizing constant c, we now compute the fuzzy rating of the spatial relation *rightFront* as can be seen in equation (6).

$$FR_{ego\_ori}(ego, relatum) = e^{-\frac{1}{2}\left(\frac{|\alpha-\beta|}{c}\right)^2} \tag{6}$$

Passing via given landmarks. The fuzzy function corresponding to the spatial relation *via* takes as its first argument the current pose of the agent, i.e. the ego. The second argument is given by the two landmarks m_1 and m_2, both labeled as relatum. The third argument *referent* is the global place we wish to reach by passing from ego via relatum. Starting from ego we choose an arbitrary relatum that is to be evaluated by the spatial relation, in the neighborhood of ego. In a first step we apply a standard graph search algorithm to compute the route ρ from the closest place nearby the ego to relatum. If ρ does not exist, we judge that relatum with the lowest possible value 0. Otherwise we continue with the computation of a circular region χ with diameter $d_\chi = \|\boldsymbol{m_2} - \boldsymbol{m_1}\|$ such that the centers of the landmarks lie on χ's circumference. The mentioned circle χ now segments ρ into the parts before, inside and after χ. Taking the length λ of ρ after χ, and the normalizing constant c, we now compute the fuzzy rating of the spatial relation *via* as can be seen in equation (7).

$$FR_{via}(ego, referent, relatum) = e^{-\frac{\lambda}{c}} \tag{7}$$

The fact that fuzzy ratings decrease for places that are farther away from χ, captures the intuitive nature of human route descriptions in that elements typically apply to the direct neighborhood of referenced landmarks.

Going until indexed junction. In order to interpret the spatial relation *until* two parameters are necessary: (a) the current pose of the agent, and (b) a relatum given by the place which we wish to travel to from the ego, e.g., the fourth junction to the left. In order to look for an indexed place with a given attribute, we first have to define the fuzzy rating for the existence of that attribute in a given place $\emptyset$. Therefore we model a junction to a given direction in analogy to the binary direction relation as described above, however with a different computation of the angle α. In this case the angle α is computed as the angle between two route segments sharing the common place $\emptyset$. We now start with the computation of the route ρ from the closest place nearby ego to relatum. Interpreting ρ as an ordered set of places $\{p_0, p_1, ..., p_n\}$, we compute the ordered set of fuzzy ratings $FR(\rho) = \{fr_1, fr_2, ..., fr_n\}$, which represent the judgments that there exists a junction to the left in the corresponding places p_i. Considering the $j = \binom{n}{i}$ possible combinations to find in p_n the i_{th} place with a junction to the left, we can now compute $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_j\}$ as the $\binom{n}{i}$-subsets of $FR(\rho)$ (cf. [15]), yielding the overall fuzzy rating for the spatial relation "until" as can be seen in (8).

$$FR_{until}(ego, relatum) = \sum_{k=1}^{j} \prod_{l=1}^{n} \begin{cases} fr_l{:}fr_l \in \mathcal{A}_k \\ 1 - fr_l{:}fr_l \notin \mathcal{A}_k \end{cases} \tag{8}$$

5.2 Interpretation of Qualitative Spatial Actions

After introducing fuzzy functions as the necessary primitives for interpreting spatial relations, we now present an algorithm that pushes us towards goal-oriented robot movement by calculating the most likely target pose given the current pose of the agent and the sequence of actions that represents the user's route description. We therefore utilize a search tree $\mathcal{S}$, the nodes of which represent fuzzy rated places that result from the evaluation of the involved spatial relations. Considering a single node of $\mathcal{S}$ as a compound of a pose and its fuzzy rating or score, the root $p1$ is given by the initial pose of the agent along with the highest possible score 1. Taking the first spatial action SA_1 that appears in the action sequence, one chooses the first candidate pose $p2$ that acts as the relatum for the fuzzy function associated with SA_1. After calculating $p2$'s own score, it is multiplied by the score of the parent node $p1$. This mechanism propagates the uncertainty of former fuzzy ratings down to lower levels of the search tree. The algorithm now recursively continues in a depth-first branch-and-bound manner with the evaluation of the following spatial relations.

In order to keep the size of the search tree small, we apply two distinct bound-criteria. First, we do not construct nodes from the evaluation of a single spatial relation if the resulting score is lower than a given threshold c_1. In our implementation we chose $c_1 = 0.25$. Second, we stop the evaluation at a given node if its cumulative score is already lower than the highest score of a leaf resulting from the evaluation of the final spatial relation. In addition to these rules, we only address candidates for the relatum of the spatial relation to be analyzed that are closer to the current ego than a given distance c_2. In our implementation we chose $c_2 = 15m$. Experiments showed that the selection of relata within a given radius c_2 around the referent is the computational most expensive operation during the construction of the search tree. We therefore implemented a quad-tree representation of the global route graph search-space in order to do fast spatial queries.

The example in Figure 6 shows the executed path of our automated wheelchair according to the proceeding interpretation of an action sequence that matches the given coarse route description.

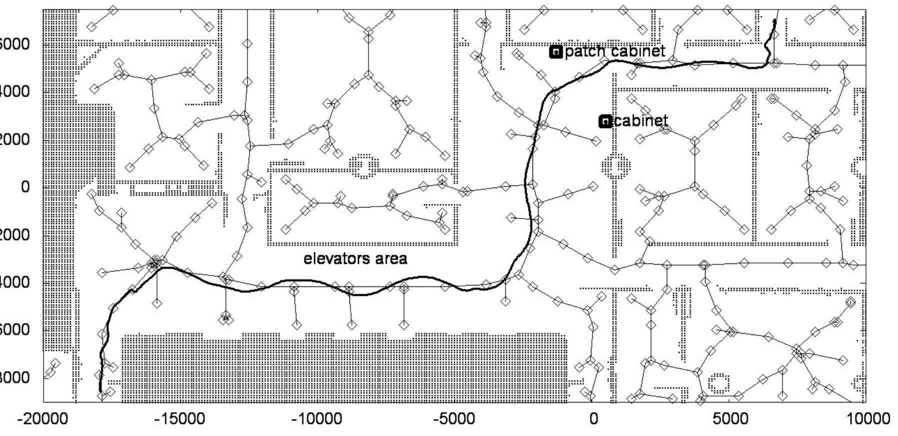


Fig. 6. Formalized and executed route according to the following route description: "...go to the elevators area and continue until the first junction to the left...turn left there and pass through the cabinet and the patch cabinet...then continue until the second door to the left...turn left, go through the door and stop...".

6 Comparing Interpretation Approaches

We have presented the Qualitative Spatial Action Model as a formalized way of representing coarse route descriptions given by people. In section 4 and section 5 we subsequently described two distinct interpretation mechanisms that directly work on the introduced spatial relations in QSAM. This section is intended to compare the logic-oriented reasoning on QSAM used by the conceptual user model and the fuzzy function based computational interpretation of the spatial relations.

6.1 Objectives

Generally speaking, qualitative spatial calculi are explicit formalisms of a human's spatial reasoning models, which aim to explain a person's behaviour in a space without precise quantitative descriptions. The Conceptual User Model presented in Section 4, as an example of such a modeling, provides an efficient way to reason about human's navigation behaviour using orientation information. On the other hand it defines the semantics of the Qualitative Spatial Actions by interpreting the spatial relations occurring in actions by the well-founded orientation relations of the Double-Cross Calculus. Moreover, combining the Double-Cross Calculus and the Route Graph makes it possible to integrate different routes into a graph like structure, to apply spatial relations between routes, and to construct new routes from existing information. Fig. 7 shows one such example, where after identifying a common place *p* and *q*, the operations provided by the Route Graph [10] allow both routes (including distinct entires and exits) to be merged into a single structure.

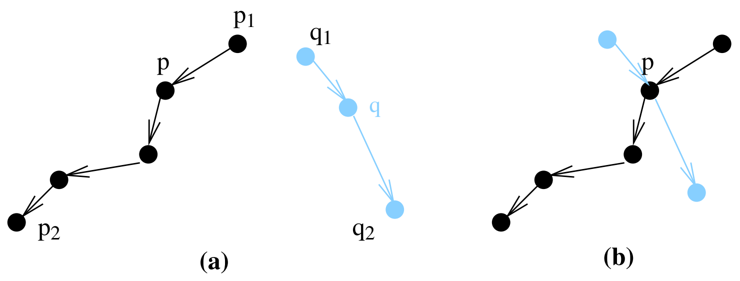


Fig. 7. Integration of two separate routes (a) into one route graph (b)

On the other hand, the interpretation of QSAM through the evaluation of fuzzy functions is motivated by the need to translate an available qualitative task description from the navigation domain into a precise target position w.r.t. a given environmental description. While we used the developed approach to infer a navigable objective for an artificial agent, i.e. an autonomous wheelchair in an office-like environment, the gained ability could also be used for different scenarios. For example one could imagine a traveler that arrived at an foreign station, asking a local for the way to a place of interest on the level of road map navigation. Recording the description of the helpful person, the stranger's PDA could calculate its own solution and assist during the way to the sightseeing.

6.2 Keynotes of Approaches

Qualitative route descriptions composed of QSAM actions may be reasoned on against an already existing construal of the users mental image of the environment where the navigation takes place. Suppose such an environment can be represented, in the Conceptual User Model, by a set of places possibly associated with landmarks, a set of routes, and a set of orientation relations, which are supposed to be known by the robotic agent. Before a new route representation in the Conceptual User Model can be added to the environment, its consistence with respect to the environment is going to be proved, which means that the spatial relations in the route representation should be consistent with those holding in the environment. Furthermore, the new introduced conditions will be simplified according to the environment – if that is possible.

To illustrate this procedure, let us return again to the sample route description in Table 5. Suppose that the stored environment contains the relation $\mathit{right\_of}(\mathbf{pp_1}, p_2)$, where p is the start place of the route description, p_1 is the door, and p_2 the lifts, i.e., the lifts on the right hand side of the vector between the start place and the door. Suppose $\mathit{course}(s_0) = \mathbf{pb}$ and $\mathit{course}(s_1) = \mathbf{bc}$, then from $\mathit{entry}(s_1) = \mathit{right}$ we know $\mathit{ori}(\mathbf{pb}, \mathbf{bc}, \mathit{right})$. Now, according to the composition table of the Double-Cross calculus, we can conclude that

$$\mathit{ori}(\mathbf{bc}, p_2, \mathit{rightFront}) \vee \mathit{ori}(\mathbf{bc}, p_2, \mathit{right}) \vee \mathit{ori}(\mathbf{bc}, p_2, \mathit{rightBack})$$

Thus, the lifts (at the place p_2) can only be passed on the right hand side. The alternation of "on the left" (i.e., $\mathit{left\_of}(\mathit{course}(s_1), p_2)$) is inconsistent.

By applying fuzzy functions that map places to the co-domain of [0..1] we take account for the coarse qualitative nature of spatial relations that reside in natural language route descriptions. In contrast to the qualitative conceptual approach, the search for a solution of a given navigation task explicitly allows for ambiguous situations by the assignment of equal probability values for similar configurations. Taking the *passBy* example from above, we interpret the spatial relation *passBy* occurring in the action $\mathit{travel}(\{\mathit{passBy}(\text{lifts})\})$ by using the appropriate fuzzy function, where places at both the left and right hand side should be judged with respect to the relatum "lifts". According to the resultant probabilities, we might conclude that the "lifts" can only be passed on the right hand side. Thus, the two interpretations produce the same result in this example, although the processes applied to achieve the result are totally different.

One of the main benefits of interpreting QSAM via the evaluation of fuzzy functions leads directly to its main disadvantage. Because situations with small probabilities may be taken into account as part of the solution (as long as they are judged to be more probable than a given threshold, cf. section 5.2), the search through the whole search space is a computationally expensive operation. Another shortcoming is the need to provide an implemented fuzzy function for each spatial relation that may appear in the corpus of the selected domain. While the logic-oriented reasoning on QSAM gets by with the definition of a normal form of spatial relations e.g. *on*, *left_of*, *right_of*, *begin_of* and *end_of*, fuzzy functions come along as very specialized implementations of the corresponding spatial relation, e.g. *through*. Up to now it remains unclear how to translate specific spatial relations to the set of relations from the normal form in an automatic way.

7 Interpretation Disparities

In this section, we are going to discuss situations, in which both interpretations deliver different results, i.e., *interpretation disparities*.

7.1 Different Levels of Abstractions

The Voronoi Graph describes a network of places and connections between those places with maximal clearance to surrounding obstacles. At the conceptual level on the other hand, the main interest is in qualitative spatial information and abstract places. For example, a room in the Voronoi Graph is represented by a set of places, whereas connections between places at the conceptual level are vectors containing no metric information. Thus, an environment in the Conceptual User Model is in fact an abstract view of the Voronoi Graph.

Since the action model can be interpreted w.r.t. two different models within different abstractions, disparities may occur in interpreting the same action. Consider for example an area including a group of lifts and the region before them (a situation in our office building scenario). Within the Voronoi Graph the region and each lift are represented by a set of places, that are connected by edges. While at the conceptual level it may be abstracted into two places, one for the region and one for the group of lifts, which are connected with two vectors with opposite directions. in order to interpret a “turn” action occurring in this area that intends to take a lift, there is at most one possible interpretation at the conceptual level. The fuzzy interpretation may result several possible solutions, since more than one lift can be taken.

7.2 Errors in Route Descriptions

Navigation tasks belong to high-level cognitive processes that involve the judgment of environment information, the localization of spatial objects, and the deciding of spatial relations between these objects. Describing a route to a robot is a special case of navigation, and may be subject to *knowledge-based mistakes* ([16]) made by users.

Unsolvable spatial information. Among the corpus examples we collected, a considerable number of utterances contained references to landmarks, such as doors, rooms, etc. An example is: “Drive past the copier room and the mailbox room.”. Here, the robot’s knowledge of the intended entities, e.g., the “copier room” and the “mailbox room”, is presupposed. Suppose that the robot has no knowledge about them. In this case the fuzzy function based approach will search for all possible places before eventually detecting missing information. The conceptual approach will first add new places with related landmarks into the environment, and then use the place identification function to find out whether the place with the given landmark already exists in the environment or not.

Spatial relation mismatches. In case where a user relates spatial objects incorrectly, taking the example “Drive past the copier room and the mailbox room.” again, and now we suppose that the mailbox room should be passed first, with the copier room after that. The Conceptual User Model might detect this mismatch, if there exist relations in the conceptual environment, such that the correct spatial relation between the copier room and the mailbox room can be inferred. During integration of the new route description, an inconsistent situation will be detected using logical reasoning. While, to interpret this action the search algorithm in the fuzzy function based approach might first reach the place for mailbox room without memorising the landmark, and then find the place for the copier room. After finding the place marked with the “copier room”, it continues to search for a place marked with the “mailbox room”. A zigzag route would be found in such a case.

Orientation mismatches. Moreover, mistakes can also be caused by using incorrect orientation, a special case of deciding spatial relations. Now consider again the scenario described in Section 6.2) where the lifts may only be passed on the right given the intended route from the user. Now on the other hand, suppose the utterance “pass by the lifts on your left” is given by the user. At the conceptual level this inconsistency can easily be detected using logical reasoning on spatial relations. While, the search algorithm of the fuzzy interpretation will search all possible places on the left without any success.

Localization mismatches. Now we consider a situation where a user believes themselves to be at a location, which is different from the one localized by the robot’s sensor system, thus the route description may contain landmarks or directional orientations which do match the robot’s internal spatial representation. If this situation occurs, both the conceptual and the fuzzy interpretations will detect the irresolvable landmark or the impossible orientation. While, sometimes a route description can be “correctly” interpreted, though the locations do not match with each other. In this case, the mismatch remains undetected.

8 Relation to Other Work

Over the past ten years there has been considerable interest in verbal HRI within the spatial domain. For reasons of space, we limit ourselves to those efforts specifically concerning the processing and modeling of route instructions.

In particular, there have been a number of proposed models which attempt to capture route instructions as spatial models [28, 11, 14]. The Route Graph, referenced in Section 4 and detailed in [27, 9], is one prominent example. The route graph is essentially a graph-based abstract representation of space which can be instantiated to a number of different *kinds* including vornoi-like low-level representations to more abstract representations such as the conceptual user model descried in Section 4. While Route Graphs are very interesting as a method for representing complete navigation spaces, little work has been performed to date on showing how individual route graphs can be composed from verbal descriptions, or on how individual route descriptions can be used against complete route graphs in computational systems. We would hope that this paper meets, at least in part, such questions.

Another approach to the interpretation of route instructions has come more directly from the robotics community [12, 23]. In the IBL project [13, 2] a number of *functional primitives* were extracted from the IBL corpus to be implemented as behaviors directly upon a robot which had a local-only rather than a global conceptualization of space. The IBL functional primitives are in some ways quite similar to the actions defined within the QSAM model. However, we have attempted to give them a grounding as actions which may be applied in a range of navigation spaces, thus allowing us to concretely relate them to the forms of spatial representation actually used in hybrid rather then only behaviour based robots. Godot on the other hand [23] made use of an internal map and used landmark locations within that map to aid the route interpretation process.

9 Summary, Conclusion, & Future Work

In this paper we have attempted to re-affirm actions as the fundamental unit of verbal route descriptions given by a user to an artificial agent. We have based our approach on existing route-instruction corpora, and proposed Qualitative Spatial Actions as actions which move an agent within the appropriate navigation space. Moreover, we have concretely linked qualitative spatial actions to two existing approaches to the representation of navigation space, thus facilitating the *interpretation* of route instructions by artificial agents.

We feel that linking these different approaches to the modeling and interpretation of routes is valuable since it allows a more widespread understanding of the practical issues involved in interpreting and reasoning on route descriptions in robotic systems. Furthermore, we also believe that the qualitative spatial actions defined are useful beyond the interpretation of full route instructions, since they can also serve as a basis of conceptual spatial instructions which should be implemented by a robot under verbal human control.

In future work we would like to extend the QSAM model to cover more of both the IBL and Rolland corpa. In particular this would involve the accommodation of quantitative or at least pseudo-quantitative movement instructions, and region related spatial relations. Having established the QSAM model, we also wish to apply the abstractions as an interface to on-line route instructions, e.g., users issuing instructions to the wheelchair as they move down the corridor.

Acknowledgments We gratefully acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Center SFB/TR 8 Spatial Cognition - Subprojects I3-SharC and A1-RoboMap. We would also like to thank Prof. John Bateman for useful discussions on some of the issues presented in this paper.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11):832–843, 1983.
2. G. Bugmann, E. Klein, S. Lauria, and T. Kyriacou. Corpus-Based Robotics: A Route Instruction Example. In *In Proceedings of IAS-8*, 2004.
3. G. Chronis and M. Skubic. Robot navigation using qualitative landmark states from sketched route maps. In *Proceeding of the IEEE 2004 International Conference on Robotics and Automation*, pages 1530 – 1535, 2004.
4. A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *Ceoinformatics*, 1:1–44, 1997.
5. M. Denis. The description of routes: A cognitive approach to the production of spatial discourse. *Cahiers de Psychologie Cognitive*, 16:409–458, 1997.
6. K. Fischer. *What Computer Talk Is and Is not: Human-Computer Conversation as Intercultural Communication*, volume Vol 17. Computational Linguistics, 2006.
7. C. Freksa. Qualitative spatial reasoning. In D. M. Mark and A. U. Frank, editors, *Cognitive and Linguistic Aspects of Geographic Space*. Kluwer, 1991.
8. C. Freksa. Using orientation information for qualitative spatial reasoning. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *LNCS*, pages 162–178. Springer-Verlag, 1992.
9. B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, and R. J. Ross. Specification of route graphs via an ontology. In C. Freksa, B. Nebel, B. Krieg-Brückner, M. Knauff, and T. Barkowsky, editors, *In Proceedings of Spatial Cognition IV, Chiemsee, Germany*, volume 3343 of *Lecture Notes in Artificial Intelligence*, pages 989–995. Springer, 2004.
10. B. Krieg-Brückner and H. Shi. Orientation calculi and route graphs: Towards semantic representations for route descriptions. In *In Proceedings of GIScience 2006, Münster, Germany*, volume To appear of *Lecture Notes in Artificial Intelligence*, 2006.
11. B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.
12. S. Lauria, G. Bugmann, and T. Kyriacou. Training personal robots using natural language instruction. *IEEE Intelligent Systems*, 16(3):38–45, 2001.
13. S. Lauria, T. Kyriacou, G. Bugmann, J. Bos, and E. Klein. Converting natural language route instructions into robot executable procedures. In *Proceedings of the 2002 IEEE International Workshop on Human and Robot Interactive Communication*, pages 223–228, 2002.
14. M. MacMahon. A framework for unterstanding verbal route instructions. In *Proceedings of AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence*, 2004.
15. A. Nijenhuis and H. Wilf. *Combinatorial Algorithms.* Academic Press, 1978.
16. J. Reason. *Human Error.* Cambridge University Press, 1990.

17. J. Renz and D. Mitra. Qualitative direction calculi with arbitrary granularity. In *Proceedings PRICAI 2004 (LNAI 3157)*, pages 65–74, Auckland, New Zealand, Sept. 2004. Springer.
18. H. Shi and T. Tenbrink. Telling rolland where to go: Hri dialogues on route navigation. In *WoSLaD Workshop on Spatial Language and Dialogue, October 23-25*, 2005.
19. M. Skubic, P. Matasakis, B. Forrester, and G. Chronis. Extracting navigation states from a hand-drawn map. In *Proceeding of the IEEE 2001 International Conference on Robotics and Automation*, 2001.
20. L. Talmy. How language structures space. In H. L. Pick and L. P. Acredolo, editors, *Spatial Orientation: Theory, Research and Application.* Plenum, NY, 1983.
21. T. Tenbrink. Identifying objects in english and german: Empirical investigations of spatial contrastive reference. In *WoSLaD Workshop on Spatial Language and Dialogue, October 23-25, 2005*, 2005.
22. T. Tenbrink. Identifying objects on the basis of spatial contrast: an empirical study. In C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, and T. Barkowsky, editors, *Spatial Cognition IV: Reasoning, Action, Interaction. International Conference Spatial Cognition 2004, Frauenchiemsee, Germany, October 2004, Proceedings*, pages 124–146, Berlin, Heidelberg, 2005. Springer.
23. C. Theobalt, J. Bos, T. Chapman, and A. Espinosa-Romero. Talking to godot: Dialogue with a mobile robot. In *Proceedings of the 2002 IEEE International Conference on Intelligent Robots & Systems*, pages 1338–1343, 2002.
24. S. Thrun. Robotics mapping a survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.
25. B. Tversky. Structures of mental spaces – how people think about space. *Environment and Behavior*, Vol.35, No.1:66–80, 2003.
26. B. Tversky and P. Lee. How space structures language. In C. Freksa, C. Habel, and K. Wender, editors, *Spatial Cognition: An interdisciplinary Approach to Representation and Processing of Spatial Knowledge*, volume 1404 of *Lecture Notes in Artificial Intelligence*, pages 157–175. Springer-Verlag, 1998.
27. S. Werner, B. Krieg-Brückner, and T. Hermann. Modelling navigational knowledge by route graphs. In C. Freksa, W. Brauer, C. habel, and K. Wender, editors, *Spatial Cognition II: Integrating Abstract Theories, Empirical Studies, Formal Methods, and Pratical Applications*, volume 1849 of *Lecture Notes in Artificial Intelligence*, pages 259–316. Springer-Verlag, 2000.
28. K. Zimmermann and C. Freksa. Qualitative spatial reasoning using orientation, distance, and path knowledge. *Applied Intelligence*, 6:49–58, 1996.

Applying a 3DOF Orientation Tracker as a Human-Robot Interface for Autonomous Wheelchairs

Christian Mandel[*] and Thomas Röfer[†] and Udo Frese[‡]

Abstract

Within this work, we demonstrate the applicability of a three degrees of freedom orientation tracker as suitable controlling equipment for an autonomous wheelchair. It is shown that a small-size sensor device, like the XSens MTx, can either serve as a joystick replacement, or as an interface device for a newly developed navigation algorithm. Specifically, the sensor device is mounted at the back of its operator's head, where it permanently measures posture values that are converted into adequate steering commands. Comprehensive experiments in a real world office scenario with untrained participants are used for evaluation purposes.

1. INTRODUCTION

Disabled people often find the use of electrical wheelchairs an indispensable aid for their rehabilitation or daily life. Since the most common way of operating such a device is given by a control loop that consists of a technical system e.g. a joystick connected to actuating elements, and the human operator, engineers have historically focused on improving aspects of these kinds of systems. The gathered results range from safety-layers that act in-between the control loop and prevent the wheelchair-bound person from colliding with any obstacles, to autonomous navigation modules that sometimes require different kinds of user interfaces than a joystick. While the work that is presented in this paper is still a technical one, its motivation clearly grounds on the user's requirements in an intelligent wheelchair.

[*]C. Mandel is with Department of Computer Science, University of Bremen, P.O.Box 330440, 28334 Bremen, Germany `cmandel@uni-bremen.de`

[†]T. Röfer is with DFKI Lab Bremen, Safe and Secure Cognitive Systems, Robert-Hooke-Straße 5, 28359 Bremen, Germany `roefer@informatik.uni-bremen.de`

[‡]U. Frese is with is with DFKI Lab Bremen, Safe and Secure Cognitive Systems, Robert-Hooke-Straße 5, 28359 Bremen, Germany `ufrese@informatik.uni-bremen.de`

People with high level quadriplegia suffer from paralysis of all four limbs while still being able to move their heads. Thus a common wheelchair-interface like a joystick moved by hand is obviously inappropriate. For this kind of handicap, we propose a head-mounted three degrees of freedom orientation tracker[1] to allow the operator of an automated wheelchair to steer his/her vehicle by intuitive head movements. In the following we will develop two methods with which a small-size IMU can be exploited for this purpose by forwarding head-posture dependent joystick-like signals, or by interfacing a fully-fledged autonomous navigation module respectively. The later one computes a nearby target-pose by intersecting the operator's line of view with a two-dimensional local obstacle map, so that a geometric path planner can subsequently compute an optimal trajectory within that map.

We begin in section 2 with a brief overview of approaches that deal with a variety of human-robot interfaces, in particular focusing on control mechanisms for automated wheelchairs. In section 3 we continue with an introduction of our experimental platform *Rolland III*, followed by a detailed view on the *XSens MTx* IMU, as well as a short overview of the software-framework applied for this work. Section 4 proceeds with a detailed account on the implementation of an IMU-based head-joystick, before we then describe in section 5 our developed geometric path planner that applies cubic Bezier curves and is interfaced with the IMU. After section 6 shows the results of an experimental evaluation that comprised both of the presented IMU-controllers, we conclude in section 7 with an assessment of the gathered results.

2. Related Work

For many years, human-robot interaction (HRI) has been an active research field that studies methods of interfacing with (semi-)autonomous robot systems.

[1]In the following we will abbreviate *three degrees of freedom orientation tracker* or rather *inertial measurement unit* by *IMU*.

Within the domain of operating electrical wheelchairs, scientists began developing techniques that should enable paralysed people to steer their vehicle.

Jaffe proposed in [1] a head position interface that applies two ultrasonic sensors, mounted at the wheelchair's head-rest in order to sense the user's head position. Despite the contactless and therewith smart fixation of the hardware components, this approach implicates the drawback that the user's head position can only be measured in two-dimensional space. Thus only basic head movements within a plane parallel to the ground can be evaluated. Within a clinical trial [2], an evaluated version of Jaffe's work has been rated by 17 participants, all suffering from spinal cord injury dysfunction. Due to the fact that the assessed equipment was nearly identical in construction to its predecessor, virtually one third of all subjects reported poor performance in commanding turns, while about 40% judged straight-line driving poor or very poor.

More recent work of Chen and colleagues [3] proposes a tilt sensor module that is attached to the back of the user's head. With the help of two integrated inertial sensors, the overall device interprets two-dimensional head movements and subsequently triggers an appropriate translational or rotational motion of the vehicle. Due to the drifting output of the sensor, the overall system can only determine movements of the head but no globally correct posture angles. Hence, only discrete steering commands can be generated, e.g. $70cm/s$ translational speed when the operator moves his/her head forward once, or $100cm/s$ translational speed wheen the operator moves his/her head forward twice.

A completely different approach in the development of an advanced wheelchair user interface is presented in the work of Canzler and Kraiss [4]. The authors describe a system that extracts and analyses facial features like head posture, gaze direction, and lip movement with the help of computer vision. While we want to abstract away from actual implementation details that comprise the adaption of detailed geometry and texture information, it should be noted that their system has been tested under real world conditions. Here it was able to recognize at least four gesture dependend commands like *go*, *stop*, *left*, and *right*.

3. System Overview

In this section we overview the hard- and software components that provide the prerequisites for the implementation of an IMU-based interface to electrical wheelchairs.

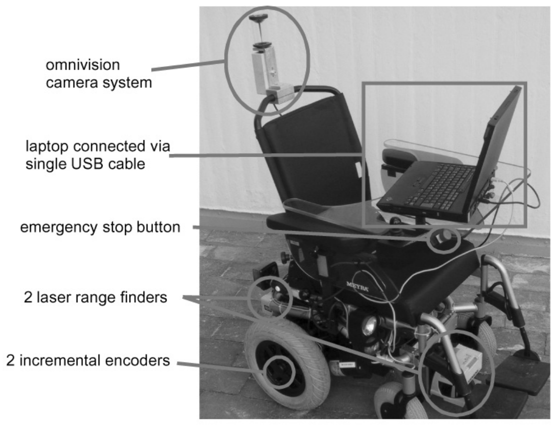


Figure 1. The autonomous wheelchair *Rolland* along with its sensorial equipment and a processing laptop.

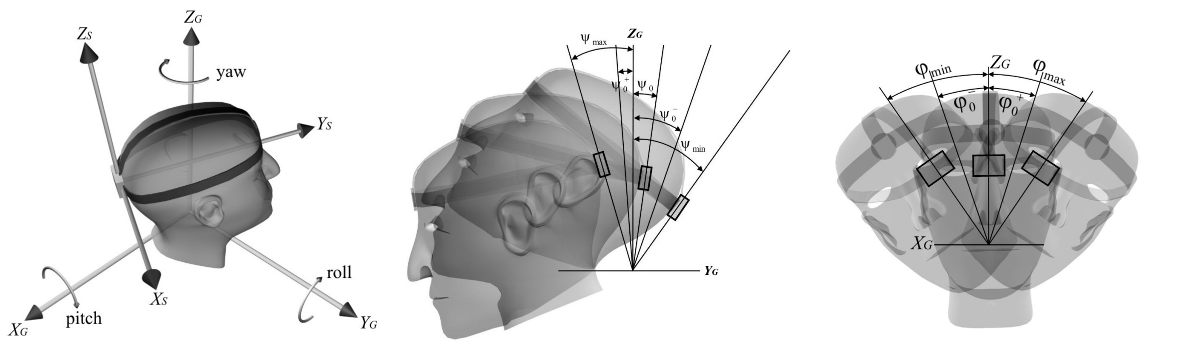
3.1. Hardware

For several years, the autonomous wheelchair *Rolland* has been used in our laboratory as an experimental platform for the investigation of questions related to the field of service- and rehabilitation-robotics, cf. [5, 6, 7]. The current model is based on the electrical wheelchair *Champ 1.594*, produced by the German company *Meyra*. It is equipped with two laser range finders, mounted beneath the operator's feet, that sense distances to nearby obstacles. Further hardware includes two incremental encoders that measure wheel-rotation for dead reckoning, an omnivision camera system, and a processing laptop, cf. Fig.1.

The *XSens MTx* IMU is a small-scale[2] electronical device that provides inertial information, namely 3D acceleration, 3D rate of turn, and 3D earth-magnetic field [8]. By the combination of the output of the device's internal accelerometers, gyroscopes, and magnetometers the IMU outputs also drift-free absolute 3D orientation data. In order to use this device for measuring the posture of a person's head, we have mounted the IMU at the head's back with the help of a small-sized and easy to wear frontlet, cf. Fig.2(a). Throughout this work we have set the configuration of the IMU to output Euler angles that describe the orientation of the IMU's local coordinate system S with respect to the fixed global coordinate system G. We refer to a single IMU reading by the triple $\mathscr{I} = (\psi, \varphi, \theta)$, as defined in (1).

$$\begin{aligned} \psi &= pitch = rotation\ around\ X_G \in [-90°...90°] \\ \varphi &= roll = rotation\ around\ Y_G \in [-180°...180°] \quad (1) \\ \theta &= yaw = rotation\ around\ Z_G \in [-180°...180°] \end{aligned}$$

[2]outline dimensions: $53*38*21\,mm^3$, weight: $30\,g$



(a) Schematic view of the inertial measurement unit mounted at the back of the user's head, including the global coordinate system G and the sensor's local coordinate system S.

(b) *left:* Maximal pitch deflection ψ_{max}, minimal pitch deflection ψ_{min}, and mean pitch deflection ψ_0, as adopted during calibration phase of the IMU. ψ_0^+ and ψ_0^- characterise the pitch dead zone, i.e. only pitch angles exceeding these values will be accepted as control commands. *right:* Roll angles $\varphi_{max}, \varphi_0^+, \varphi_0, \varphi_0^-, \varphi_{min}$ are defined in analogy.

Figure 2. Illustration of an inertial measurement unit that is attached to the user's head with the help of an easy to wear frontlet. The depicted head posture angles are calculated during the calibration phase of the IMU and given in the global coordinate system G.

3.2. Software

The hardware described above, which is used within the context of this work, is driven by an adaption of a software architecture that was originally developed for the GermanTeam, the world champion 2004 and 2005 in the Four-Legged League in RoboCup [9]. The framework structures the code into *modules*, *representations*, and *processes*. A module solves a specific task and is encapsulated by a well-defined interface consisting of representations. For each module, several solutions may exist, between which one can switch at runtime, and a module may also be deactivated completely. Processes run concurrently and group modules together. They define the flow of information (the representations) between the modules, whether modules are part of the same process or of different ones. For the purpose of applying a head-mounted IMU as a joystick replacement, there exist at least two important modules. The *Drive Controller* gathers the raw data coming from the IMU, and converts this information into desired translational and rotational speeds for the vehicle. See sec. 4 for an in-depth discussion of this module. Before these values are forwarded to the executing actuators, they are assessed by a crucial second module, the so-called *Safety Layer*. The key concept in the implementation of the safety layer is the *Virtual Sensor*. For a given initial orientation (θ) of the robot and a pair of translational (v) and rotational (w) speeds, it stores the indices of cells of a local obstacle map that the robot's shape would occupy when initiating an immediate full stop manoeuvre. A set of precomputed virtual sensors for all combinations of (θ, v, w) then allow us to check the safety of any driving command issued by the Drive Controller.

In the case of utilising the IMU as an interface device for a geometric path planning algorithm, there are three major modules involved. At first, the *TargetRequestGenerator* tries to convert the actual head posture information, i.e. the current IMU reading $\mathscr{I}$, into a desired *TargetRequest*, cf. sec. 5.1. Afterwards, the *LocalPathPlanner* searches for a suitable trajectory towards the given target request. See section 5.2 for a discussion of this module. Beside the subsequent execution of a path- and a velocity control module, whose discussion is left out for the sake of compactness, it should be noted that within this scenario the safety layer is also applied. Therefore the overall system gains a second level of safety mechanism, this in addition to the local path planner module that itself only generates obstacle free paths.

4. IMU-based Head-Joystick

This section presents implementation details for a Drive Controller that interprets IMU-readings as head-joystick signals. In the targeted application scenario, the operator controls his/her vehicle by pitching his/her head forwards and backwards in order to control translational velocity. In analogy, left and right roll movements of the users's head control rotational velocity.

Before we can use the IMU as a joystick replacement, that is in use by a certain operator, we have to calibrate the device in the sense that we want to adopt the minimal, the mean, and the maximal deflection of the person's head w.r.t. each axis in use. For this reason, let P_{min} and P_{max} be two sets of IMU-readings that have been taken while the user has pitched his/her head with maximal deflection forwards and backwards respectively. In analogy, let R_{min} and R_{max} be two sets of IMU-readings that characterise the minimal and the maximal roll deflection of the user's head. By computing the arithmetic mean of the ψ and φ components of each of the four sets, we get a good approximation for the user's minimal and maximal head deflections, i.e. ψ_{max}, ψ_{min}, φ_{max}, and φ_{min}. Furthermore we describe the rest position of the person's head by $\psi_0 = \frac{\psi_{max}+\psi_{min}}{2}$ and $\varphi_0 = \frac{\varphi_{max}+\varphi_{min}}{2}$.

In order to allow the wheelchair bound person to move his or her head a bit without causing unintended motion, we now define a dead zone around ψ_0 and φ_0 by introducing ψ_0^+, ψ_0^-, φ_0^+, and φ_0^-, cf. Fig.2(b) for an illustration of the defined angles. A valid head-joystick command (v,w) is now solely defined for input values $(\psi_{valid}, \varphi_{valid})$ that satisfy (2), and computed as in (3). Constants c_v and c_w are used to map v and w onto the velocity-domain of a particular vehicle.

$$\begin{aligned} \psi_{valid} &\in \left[\psi_{max} \cdots \psi_0^+\right] \cup \left[\psi_0^- \cdots \psi_{min}\right] \\ \varphi_{valid} &\in \left[\varphi_{max} \cdots \varphi_0^+\right] \cup \left[\varphi_0^- \cdots \varphi_{min}\right] \end{aligned} \tag{2}$$

$$\begin{aligned} v &= c_v \frac{\psi_{valid} - \begin{cases} \psi_0^+ & : \ \psi_{valid} > \psi_0^+ \\ \psi_0^- & : \ \psi_{valid} < \psi_0^- \end{cases}}{\begin{matrix} \psi_{max} - \psi_0^+ & : & \psi_{valid} > \psi_0^+ \\ -\psi_{min} + \psi_0^+ & : & \psi_{valid} < \psi_0^- \end{matrix}} \\ w &= c_w \frac{\varphi_{valid} - \begin{cases} \varphi_0^+ & : \ \varphi_{valid} > \varphi_0^+ \\ \varphi_0^- & : \ \varphi_{valid} < \varphi_0^- \end{cases}}{\begin{matrix} \varphi_{max} - \varphi_0^+ & : & \varphi_{valid} > \varphi_0^+ \\ -\varphi_{min} + \varphi_0^+ & : & \varphi_{valid} < \varphi_0^- \end{matrix}} \end{aligned} \tag{3}$$

During early experiments with the proposed head-joystick, we observed strong feedback effects between the translational acceleration v' of the wheelchair, and the user's head pitch angle ψ. In order to achieve a smoothed acceleration behaviour, we implemented a basic damping mechanism that replaces a velocity command v_t at a given point in time by the arithmetic mean of former velocity commands $\dot{v_t}$.

$$\dot{v_t} = \frac{1}{n} \sum_{n=0}^{n_{max}} v_{t-n} \tag{4}$$

Note that the computation of $\dot{v_t}$ as can be seen in (4), still preserves the safety issue of setting the translational velocity to zero if the actual pitch angle of the user's head lies within the defined dead zone, i.e. ψ_t does not hold to (2).

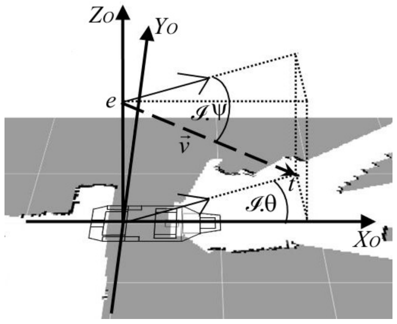
5. IMU-based Interface for Local Path Planning Algorithms

The application scenario in mind, when using the IMU as an interface-device for a geometric path planning algorithm, does no more put the operator in a continuous control loop like the implementation of a head-joystick in section 4 did. Instead, the user issues discrete driving commands by facing his/her head to a desired target-position once, and lets the system autonomously execute the given task. Within this section we first describe the deduction of a target-position by intersecting the line of the pilot's view with a local obstacle map. Furthermore we show that a suitable goal-orientation is given by a tangent lying at the target-position within a given distance map that stores the distances to the closest obstacle for each cell. After that, our geometric path planning approach is discussed. It applies cubic Bezier curves for the computation of navigable paths that connect the actual pose of the vehicle with the target-pose commanded by the operator.

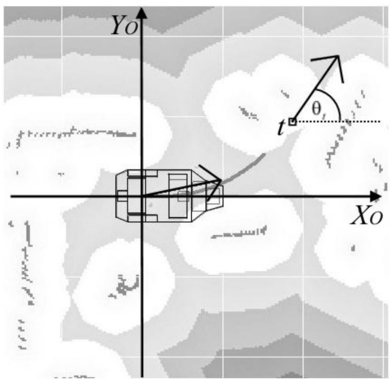
5.1. Computation of Target-Pose from IMU-Reading

The first step in the computation of a target-pose from a single IMU-reading $\mathscr{I}$ is given by the computation of the target's location. For this purpose we assume a vector $\vec{v}$ based in-between the eyes of the pilot e, initially aligned with the wheelchair's heading and parallel to the ground. We now rotate $\vec{v}$ by the directed difference between the wheelchair's heading θ_{wc} and the global yaw angle of the pilot's head $\mathscr{I}.\theta$ around Z_G, or around Z_O respectively. This approach requires that the odometry coordinate system O of the wheelchair is always consistent with the global coordinate system G of the IMU. To overcome drifting errors within O, caused by wheel slippage and imprecisely modeled wheel diameter, we currently adapt $\theta_{wc} - \mathscr{I}.\theta$ manually if the user looks forward and sets up a synchronization command. Next, $\vec{v}$ is rotated by the user's head pitch angle $\mathscr{I}.\psi$ around X_G, or Y_O respectively. The resulting vector $\vec{v}$, that represents the direction of the user's view within O, is now intersected with an obstacle map. Each cell of that map describes whether a given cell in the vicinity of the robot is free or occupied by an obstacle. Under the assumption that the user hasn't looked in parallel to the ground, nor pitched his/her head upwards, the overall procedure yields an intersection point t, that serves as the aimed target position. For an illustration of this step, confer Fig.3(a).

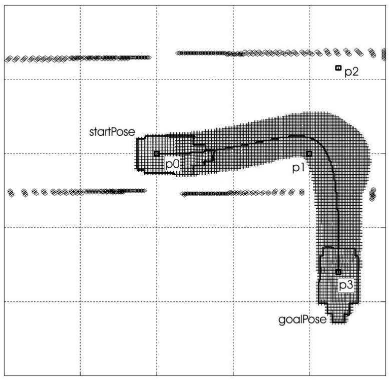
In the second step, we augment the target position t by a desired target heading θ_t. In order to reduce the de-



(a) Obstacle map including the shape of the wheelchair, heading to the right ($\theta_{wc} = 0$). The direction of the operator's view $\vec{v}$ determines the selected target position t. The odometry coordinate system O is consistent with the IMU's global coordinate system G.



(b) Distance map including the shape of the wheelchair, heading to the right. The direction of the pilot's view is shown by the arrow based in the center of the map and rotated somewhat to the left. The target-pose (t, θ_t) is depicted by the arrow to the upper right.



(c) Obstacle map including cubic Bezier curve-based path, connecting startPose and goalPose. Control points $\vec{p}_1$ and $\vec{p}_2$ are located on straight lines that i) pass through $\vec{p}_0$ resp. $\vec{p}_3$ and ii) are aligned to the orientation of *startPose* resp. *goalPose*.

Figure 3. The pipeline of processes within the scenario of using the IMU as an interface device for a geometric path planning algorithm ranges from Fig.3(a), the derivation of a suitable target position t, over Fig.3(b), the augmentation of t with an appropriate target heading θ_t, up to Fig.3(c), the computation of an obstacle free path from the current pose of the vehicle to (t, θ_t).

mands on the user in handling his/her interface device, this step is fully automated. The idea is, that an appropriate orientation of the vehicle while driving around is almost in paralell to the surrounding obstacles. Thus we compute θ_t as the angle between the tangent to the iso-distance lines in a given distance map at target position t, and the x-axis of the odometry coordinate system O. For an illustration of this step, confer Fig.3(b). It should be noted, that the current implementation selects θ_t in such a way that the target heading always points away from the current pose of the vehicle. Future implementations that should not only allow corridor navigation, but also more complicated shunting behaviours, have to differentiate at this point.

5.2. Geometric Path Planner

Almost all wheeled mobile robots that are designed to autonomously navigate in a populated and unstructured environment, use periodic sensor measurements in order to perceive their actual surrounding. Static as well as dynamic obstacles are then inserted into a local obstacle map that is big enough to cover imminent movements of the vehicle. Within that map, common navigation approaches either apply reactive behaviours like landmark tracing, or plan complete trajectories to the desired target. Prominent examples for the first class of navigation approaches are the Virtual Force Field Method (VFFM)[10], Nearness Diagramm Navigation (NDN)[11], Reactive Navigation in the Ego-KinoDynamic Space (EKDS)[12] applied with the Potential Field Method (PFM)[13]. Instances of the class of geometric path planners are given by the Dynamic Window Approach (DWA)[14], an integrated approach to goal-directed obstacle avoidance under dynamic constraints in dynamic environments (GDOA)[15], and many others.

Despite this vast spectrum of available navigation techniques that each works well within its own application scenario, we observed the need for an own development due to the kinematic restrictions of our experimental platform. *Rolland's* actuating system is given by a common differential drive at which the two actuated wheels are located at the back of the vehicle. Thus a turn on the spot command will lead to a rotation around the midpoint of the rear-axle. Considering now the task of entering a small door from within a narrow corridor, cf. Fig.3(c), it is obvious that a simple circular path that approaches *goalPose* would lead to a collision with the door's outer durn. Instead we have to model a sufficient haul-off movement that early enough brings the vehicle into a pose that is orthogonal to the passage to be crossed. From the class of behaviouristic navigation methods, VFFM, NDN and PFM would force the wheelchair in the scenario above to pass along the right-hand wall until the door frame. However, the following right turn beahaviour would lead to a collision because of the problem stated above. Similar problems arise when employing representatives out of the class of geometric path planners. The DWA for example is well suited for circular shaped robots because it only plans one circular arc ahead when searching for a path that leads to the target. However the problem of necessary haul-off movements is left untreated.

5.2.1. Basic Spline Search-Space. Motivated by the insights above, we decided to employ a geometric path planner using cubic Bezier curves, since they are able to connect two given points while accounting for a desired curve progression and for directional requirements in the start point and end point. Considering the work of Hwang et al. [16] which gives a broad overview on approaches using this type of curve, we will now sketch our basic algorithm. Given the current pose of the wheelchair $startPose = (x_s, y_s, \theta_s)$ and the desired target $goalPose = (x_g, y_g, \theta_g)$, we search the space of cubic Bezier curves for paths that

i) connect $\vec{p}_0 = (x_s, y_s)$ with $\vec{p}_3 = (x_g, y_g)$,
ii) are smoothly aligned with θ_s in $\vec{p}_0$ and with θ_g in $\vec{p}_3$,
iii) are obstacle free in the sense that a contour of the robot shifted tangentially along the path does not intersect with any obstacle point from a given obstacle map.

Equation (5) describes a cubic Bezier curve, connecting the points $\vec{p}_0$ and $\vec{p}_3$, at a given arc length $t \in [0..1]$. In order to unequivocally determine the characteristics of the curve, we still have to chose the control points $\vec{p}_1$ and $\vec{p}_2$ such that we fulfil requirements ii) and iii).

$$\begin{aligned} \vec{p}(t) &= \vec{a}t^3 + \vec{b}t^2 + \vec{c}t + \vec{p}_0,\ t \in [0..1] \\ \text{with } \vec{c} &= 3(\vec{p}_1 - \vec{p}_0), \\ \vec{b} &= 3(\vec{p}_2 - \vec{p}_1) - \vec{c}, \\ \vec{a} &= \vec{p}_3 - \vec{p}_0 - \vec{b} - \vec{c} \end{aligned} \tag{5}$$

The computation of the free parameters $\vec{p}_1$ and $\vec{p}_2$, as can be seen in (6), spans the basic search space over the cubic Bezier curves, whose solution is intended to solve our path planing problem.

$$\begin{aligned} \vec{p}_1(l_1) &= \vec{p}_0 + l_1 \begin{pmatrix} \cos(\theta_s) \\ \sin(\theta_s) \end{pmatrix},\ l_1max > l_1 > 0 \\ \vec{p}_2(l_2) &= \vec{p}_3 - l_2 \begin{pmatrix} \cos(\theta_g) \\ \sin(\theta_g) \end{pmatrix},\ l_2max > l_2 > 0 \end{aligned} \tag{6}$$

Fig.3(c) illustrates the result of a basic path planning cycle. It shows the obstacle map including the integral of former sensor measurements along with the current state of the robot *startPose* and the desired target *goalPose*. The solid drawn cubic Bezier curve has been chosen as a solution in fulfillment of requirements i) - iii) that minimizes the time of travel. The upper velocity-bound of the robot at point $\vec{p}(t)$ is therefore determined by the minimal distance between the robot-contour tangentially located at $\vec{p}(t)$ to any obstacle-point, and the curvature $c(t)$.

Table 1. The comparison includes experimental data of 15 subjects that each drove an approximately $25m$ long path by using a common joystick and the IMU-based head-joystick respectively. For a discussion of the recorded data confer sec. 6.

Criterion	Common Joystick	IMU as Head-Joystick
∅ time of travel	30.73 s	55.03 s
∅ length of travel	22.45 m	25.03 m
∅ average speed	0.76 m/s	0.50 m/s
∅ safety layer interventions	111.04 ms	445.76 ms

5.2.2. Extended Spline Search-Space. Even though the computation of the basic spline search space is adequate for most of the real world navigation situations, there exist special cases that ask for a different treatment. Imagine the case where *startPose* and *goalPose* are both located on a straight line, and that further holds $\theta_s = \theta_g$. If there is now an obstacle located on the direct connection between both poses, the selection of p_1, p_2, p_3 and p_4 according to section 5.2.1 does not yield any navigable spline-based path. To overcome this situation, we introduce an extended search space, that temporarily replaces $\vec{p}_3$ by $\vec{p_3'}$ as in (7), and computes p_1 and p_2 as in (6).

$$\vec{p_3'}(l_3) = \vec{p}_3 \pm l_3 \begin{pmatrix} \cos(\theta_g + \frac{\pi}{2}) \\ \sin(\theta_g + \frac{\pi}{2}) \end{pmatrix}, l_3max > l_3 > 0 \tag{7}$$

This heuristic rule translates the target's position on a line that is orthogonal to the vector from p_0 to p_3.

6. Experimental Evaluation

Both wheelchair interfaces that were presented throughout this work have been tested in a twofold experimental evaluation. By means of a first empirical test, we compared the performance of 15 untrained participants to steer *Rolland* in an unstructured and populated office-like environment. Thereby all participants used a common joystick first, and afterwards the IMU-based head-joystick. Fig.5 gives a first look on the collected data. Both parts show the navigated paths in drifting odometry coordinates, whereby we can explain the strong deviations culminating in the aimed target at the upper right part of the plots. Nevertheless, it is easy to see that the curves resulting from paths driven by the IMU-based head-joystick show oscillations particularly in sections of straight ahead movement. This indicates a problem of most of the participants, in that they couldn'd steer the vehicle on an almost straight line while moving with high translational speed. We expect this effect to be weakened by an upcoming redesign of the dead zone that currently disregards head-joystick commands from within a static intervall around the head's rest position. A dynamic roll dead zone that increases for high pitch angles, promises to neglect slight roll movements of the user's head in situations of unrestricted straight line movement.

Further data on the comparative experiment can be taken from Table 1. It is shown that the average time of travel by using an IMU-based head-joystick is about one third longer compared to the same path driven with a common joystick. The drawbacks of the head-joystick become even more evident when we interpret the average number of safety layer interventions for both types of interfaces. In section 3.2 we described the safety layer as a software module that permanently monitors the commanded translational and rotational velocity. In a situation where a combination of the desired speeds wouldn't allow for a safe breaking manoeuvre, the safety layer initiates a hard full stop. Considering the number of safety layer interventions as a valid safety metric, and the modules frequency of execution which is at least $30Hz$, Table 1 reveals that the application of the head-joystick is about four times more unsafe than the use of a common joystick.

Following the empirical evaluation that is described above, we want to prove our concept of interfacing with a geometric path planner by interpreting the user's head posture. We have chosen the format for an experimental test run instead of a comparative test series, and justify this approach because of the still too complex overall user interface. For example, the operator of the autonomous wheelchair had to give 32 verbal go-commands[3], while each time facing his/her head to the desired goal, in order to complete the path that is depicted in Fig.4. The reason for this undesired high amount of simple instructions is that a valid target-pose can currently only be located within the local obstacle map, that is $7 * 7m^2$ wide.

[3]For recognizing simple utterances like *go* for the confirmation of a selected target pose or *sync* for the alignment of the drifting odometry coordinate system with the IMU's globally correct coordinate system, we employ the of-the-shelf speech recognizer *Vocon* [17].

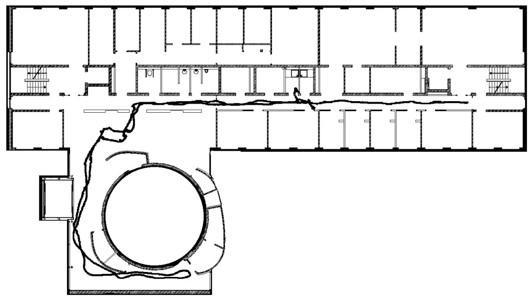


Figure 4. The depicted path has an overall length of approximately $127m$, and was estimated by a Monte-Carlo-Localisation method that is based on an implementation of the German RoboCup Team [18].

.

7. Conclusion

Although that we have shown the applicability of a small-size IMU as suitable controlling equipment for a (semi-)autonomous wheelchair, there persist several disadvantages in interfacing with a vehicle by head movements compared to the application of a common joystick. Future work must therefore bring up better solutions for the system's intrinsic inaccuracies like feedback effects or oscillations in steering control. Using the IMU for commanding a path planning module, we found that the number of triggering voice commands must be reduced for the sake of usability.

ACKNOWLEDGMENTS

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) in context of the Sonderforschungsbereich/Transregio 8 *Spatial Cognition*.

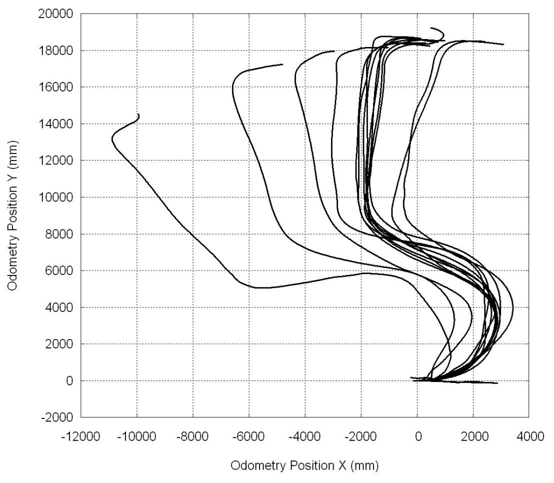
References

[1] D. L. Jaffe, "An ultrasonic head position interface for wheelchair control," *Journal of Medical Systems*, vol. 6, no. 4, pp. 337–342, 1982.

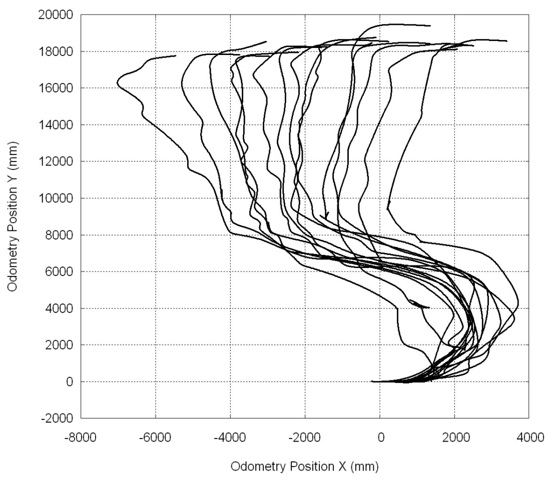
[2] J. M. Ford, "Ultrasonic head controller for powered wheelchairs," *Journal of Rehabilitation Research and Development*, vol. 32, no. 3, pp. 280–284, 1995.

[3] Y.-L. Chen, S.-C. Chen, W.-L. Chen, and J.-F. Lin, "A head orientated wheelchair for people with disabilities," *Disability and Rehabilitation*, vol. 25, no. 6, pp. 249–253, 2003.

[4] U. Canzler and K.-F. Kraiss, "Person-adaptive facial feature analysis for an advanced wheelchair user-interface," in *Proceedings of the IEEE Intl. Conf. on Mechatronics and Robotics*, 2004.



(a) Paths driven by a standard joystick.



(b) Paths driven by a head-joystick.

Figure 5. In an experimental evaluation, 15 subjects were asked to navigate *Rolland* on an approximately $25m$ long s-like shape. Both plots show the driven paths in drifting odometry coordinates, whereas the left-hand one depicts the results driven via a common joystick, and the right-hand one via a head-joystick respectively.

[5] A. Lankenau and T. Röfer, "A safe and versatile mobility assistant," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 29–37, 2001.

[6] C. Mandel, K. Hübner, and T. Vierhuff, "Towards an autonomous wheelchair: Cognitive aspects in service robotics," in *Proceedings of Towards Autonomous Robotic Systems (TAROS)*, 2005.

[7] C. Mandel, U. Frese, and T. Röfer, "Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs," in *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[8] XSens Motion Technologies. (2006) Xsens inertial measurement unit product description and documentations. XSens Technologies. [Online]. Available: http://www.xsens.com/

[9] T. Röfer, H.-D. Burkhard, J. Hoffmann, M. Jüngel, D. Göhring, M. Lötzsch, U. Düffert, M. Spranger, B. Altmeyer, V. Goetzke, O. Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbel, W. Nistico, C. Schumann, and M. Wachter, "Germanteam robocup 2004, team-report," online, Tech. Rep., 2004. [Online]. Available: http://www.germanteam.org

[10] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, - 1989.

[11] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, 02 2004.

[12] J. Minguez and L. Montano, "The ego-kinodynamic space: Collision avoidance for any shape mobile robots with kinematic and dynamic constraints." [Online]. Available: citeseer.ist.psu.edu/minguez03egokinodynamic.html

[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotic Research*, no. 1, pp. 90–98.

[14] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance, Tech. Rep. IAI-TR-95-13, 1 1995. [Online]. Available: citeseer.ist.psu.edu/fox97dynamic.html

[15] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002, pp. 508–513.

[16] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[17] Nuance Communication Technologies. (2007) Nuance vocon speech recognizer product description and documentations. Nuance Technologies. [Online]. Available: http://www.nuance.com/vocon

[18] T. Röfer and M. Jüngel, "Vision-based fast and reactive monte-carlo localization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

Design Improvements for Proportional Control of Autonomous Wheelchairs Via 3DOF Orientation Tracker

Christian Mandel[1], Udo Frese[2], and Thomas Röfer[2]

[1] University of Bremen, P.O.Box 330440, 28334 Bremen, Germany,
cmandel@uni-bremen.de,
WWW home page: http://www.informatik.uni-bremen.de/~cman/
[2] DFKI Lab Bremen, Safe and Secure Cognitive Systems,
Robert-Hooke-Straße 5, 28359 Bremen, Germany

Abstract. This paper presents a three degrees of freedom orientation tracker as suitable controlling equipment for an automated wheelchair. Mounted at the back of an operator's head by the help of an easy to wear frontlet, the device permanently outputs the user's head posture which can be used as a joystick-like signal. Within an experimental evaluation we demonstrate the applicability of the proposed control interface even for untrained users.

1 Introduction

Electrical wheelchairs make up one of the fundamental tools that ease the rehabilitation time or the everyday life of disabled people. In most of the cases the handicapped's inability to walk by his/her own is substituted by a general control loop that embeds the operator's remaining sensomotor capabilities into the vehicle's actuating body. The most common example for these kinds of systems requires the user to push a joystick into the desired direction of movement and therewith triggering a proportional translatory and rotatory motor activity of the wheelchair. Unfortunately, such an intuitive interface is impractical for certain groups of patients. For example, people suffering from certain spinal cord injury dysfunctions can only move body parts that are located above their shoulders. In this work we propose the use of a head-mounted three degrees of freedom orientation tracker[3] to give the target group mentioned above an easy way of controlling their vehicle by intuitive head movements. In particular we will pursue our own preliminary work[1], that culminated in an empiric study with 15 untrained participants, each of them comparing the applicability of our basic implementation of an IMU-based head-joystick versus a common joystick.

We begin in section 2 with a general survey on methods that focus on human-robot interfaces for automated wheelchairs. In section 3 we continue with

[3] Throughout this work we will abbreviate *three degrees of freedom orientation tracker* or rather *inertial measurement unit* by IMU.

the presentation of the (semi-)autonomous wheelchair *Rolland* that is used in our laboratory as a demonstrator for developments in the field of service- and rehabilitation-robotics. Confer [2–4] for a broader overview. An entering guide to the major software modules applied throughout this work completes this section. Then in section 4 we introduce the basic implementation of an IMU-based head-joystick that converts information about the pilot's head posture into appropriate control commands for a (semi-)autonomous wheelchair. After section 5 proceeds with several technical improvements that draw on the algorithmic evaluation of the IMU's output data, section 6 presents results of an experimental evaluation that compares the performance of untrained participants testing the different versions of the proposed head-controller. We conclude in section 7 with a critical assessment of the gathered results.

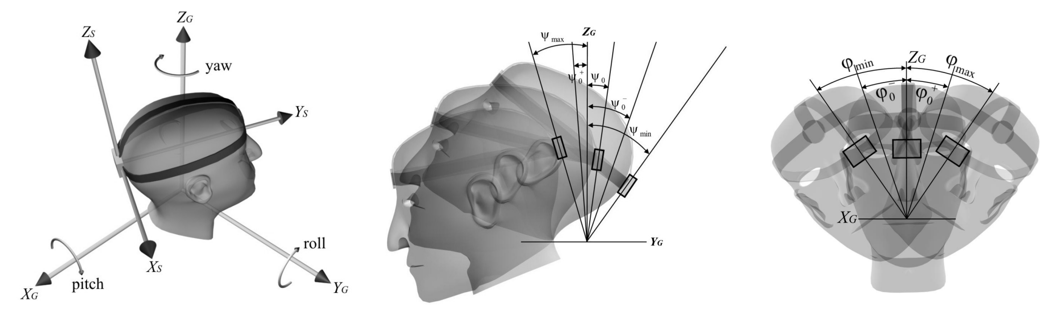
2 Related Work

Human-robot interaction (HRI) is an active research field that investigates methods of interfacing with (semi)-autonomous robot systems. In the following we give a short survey on sophisticated techniques that enable paralysed people to operate electrical wheelchairs.

Jaffe proposes in [5] a head position interface that has been evaluated within a clinical trial [6]. Due to the applied sensorial hardware, i.e. two ultrasonic sensors mounted at the wheelchair's headrest, the system can only measure the head's movement within a two-dimensional plane that lies parallel to the ground. Chen and colleagues propose in a more recent work [7] the use of two inertial sensors that are attached to the head of a wheelchair-bound person. The overall system triggers discrete commands, e.g. $70cm/s$ translational speed when the operator moves his/her head forward once, or $100cm/s$ translational speed when the operator moves his/her head forward twice. A different approach is presented by Canzler and Kraiss [8]. By the help of computer vision, the authors analyse facial features like head posture, direction of gaze, and lip movement. Hereby they are able to recognize at least four gesture dependent commands like *go*, *stop*, *left* and *right*.

3 Hardware and Software Prerequisites

Our experimental platform *Rolland* is based on the electrical wheelchair *Champ 1.594*, produced by the German company *Meyra*. With its modular configuration it is supported by miscellaneous hardware components. For the experiments that where conducted throughout this work, *Rolland* was supplied by two *Siemens LS4* laser range finders. Mounted beneath the feet of a human operator, they sense distances to nearby obstacles. The second basic component comes along by two *Lenord+Bauer Gel 248* incremental encoders, that measure the rotational speeds of the two actuated wheels for doing dead-reckoning.



(a) All measured angles describe the posture of the IMU's local coordinate system S w.r.t. a fixed global coordinate system G.

(b) *left*: Maximal pitch deflection ψ_{max}, minimal pitch deflection ψ_{min}, and mean pitch deflection ψ_0. ψ_0^+ and ψ_0^- describe the pitch dead zone, i.e. only pitch angles exceeding these values will be accepted as control commands. *right*: Roll angles $\varphi_{max}, \varphi_0^+, \varphi_0, \varphi_0^-, \varphi_{min}$ are defined in analogy.

Fig. 1. The IMU that is mounted at the back of the user's head with the help of an easy to wear frontlet outputs head posture angles that are converted into steering commands. The stressed angles in the middle and right illustration are calculated during the calibration phase of the imu and given in the global coordinate system G.

For the purpose of measuring globally correct head posture angles we apply the *XSens MTx* IMU, that is a small-scale[4] electronical device which provides inertial information, namely 3D acceleration, 3D rate of turn, and 3D earth-magnetic field [9]. By the combination of the output of the device's internal accelerometers, gyroscopes, and magnetometers the IMU outputs also drift-free absolute 3D orientation data. In order to use this device for measuring the posture of a person's head, we have mounted the IMU at the head's back with the help of a small-sized and easy to wear frontlet, cf. Fig.1(a).

The two most important software modules that are involved throughout this work are the *Drive Controller* and the so-called *Safety Layer*. The former one gathers raw data coming from the IMU, and converts this information into desired translational and rotational speeds for the vehicle. Confer sections 4 and 5 for a detailed account on the proposed drive controller. The key concept in the implementation of the safety layer is the *Virtual Sensor*. For a given initial orientation (θ) of the robot and a pair of translational (v) and rotational (w) speeds, it stores the indices of cells of a local obstacle map that the robot's shape would occupy when initiating an immediate full stop manoeuvre. A set of precomputed virtual sensors for all combinations of (θ, v, w) then allow us to check the safety of any driving command issued by the drive controller.

4 Basic Implementation of an IMU-based Head-Joystick

In this section we present implementation details for a software module that interprets IMU-readings as head-joystick signals. Within the targeted application

[4] outline dimensions: $53 * 38 * 21\,mm^3$, weight: $30\,g$

scenario, the operator permanently controls his vehicle by pitching his head forwards and backwards in order to control translational velocity. In analogy, left and right roll movements of the user's head control rotational velocity. For a clarification of the possible directions of head movement confer Fig. 1(a). Throughout this work we have set the configuration of the IMU to output Euler angles that describe the orientation of the IMU's local coordinate system S with respect to the fixed global coordinate system G. We refer to a single IMU reading by the triple $\mathcal{I} = (\psi, \varphi, \theta)$. The nomenclature of the three axis of rotation and the valid output intervals are given in (1).

$$\begin{aligned} \psi &= pitch = rotation\, around\, X_G \in [-90°...90°] \\ \varphi &= roll = rotation\, around\, Y_G \in [-180°...180°] \\ \theta &= yaw = rotation\, around\, Z_G \in [-180°...180°] \end{aligned} \tag{1}$$

4.1 Head-Joystick Calibration

Before the IMU can be used as a joystick-replacement, the device has to be calibrated in the sense that the minimal, the mean, and the maximal deflection of the operator's head has to be adopted. For this purpose, let P_{min} and P_{max} be two sets of IMU-readings that have been taken while the user has pitched his head with maximal deflection forwards and backwards respectively. In analogy, let R_{min} and R_{max} be two sets of IMU-readings that characterise the minimal and the maximal roll deflection of the user's head. By computing the arithmetic mean of the ψ and φ components of all four sets, we get a good approximation for the user's minimal and maximal head deflections, i.e. ψ_{max}, ψ_{min}, φ_{max}, and φ_{min}. Furthermore we describe the rest position of the person's head by $\psi_0 = \frac{\psi_{max}+\psi_{min}}{2}$ and $\varphi_0 = \frac{\varphi_{max}+\varphi_{min}}{2}$. In order to allow the wheelchair-bound person to move his or her head somewhat without causing an unintended command, we now define a dead zone around ψ_0 and φ_0 by introducing ψ_0^+, ψ_0^-, φ_0^+, and φ_0^-, cf. Fig. 1(b) for an illustration of the defined angles. A valid head-joystick command is now solely defined for input values $(\psi_{valid}, \varphi_{valid})$ that satisfy (2).

$$\begin{aligned} \psi_{valid} &\in \left[\psi_{max}...\psi_0^+\right] \cup \left[\psi_0^-...\psi_{min}\right] \\ \varphi_{valid} &\in \left[\varphi_{max}...\varphi_0^+\right] \cup \left[\varphi_0^-...\varphi_{min}\right] \end{aligned} \tag{2}$$

4.2 Computation of Proportional Control Commands

After having defined the domain of valid head posture angles, we can specify the transfer function for mapping these angles onto the rotational and translational speeds of the vehicle. Initially we chose a linear transfer function in analogy to the proportional characteristic curve of a common joystick. Constants c_v and c_w in the formulae for the computation of translational and rotational speeds (3) are used to map v and w onto the velocity-domain of a particular vehicle.

$$v = c_v \frac{\psi_{valid} - \begin{cases} \psi_0^+ & : \ \psi_{valid} > \psi_0^+ \\ \psi_0^- & : \ \psi_{valid} < \psi_0^- \end{cases}}{\begin{array}{rcl} \psi_{max} - \psi_0^+ & : & \psi_{valid} > \psi_0^+ \\ -\psi_{min} + \psi_0^+ & : & \psi_{valid} < \psi_0^- \end{array}}$$
$$w = c_w \frac{\varphi_{valid} - \begin{cases} \varphi_0^+ & : \ \varphi_{valid} > \varphi_0^+ \\ \varphi_0^- & : \ \varphi_{valid} < \varphi_0^- \end{cases}}{\begin{array}{rcl} \varphi_{max} - \varphi_0^+ & : & \varphi_{valid} > \varphi_0^+ \\ -\varphi_{min} + \varphi_0^+ & : & \varphi_{valid} < \varphi_0^- \end{array}} \qquad (3)$$

For an illustration of the resulting linear transfer function confer Fig.2(a). Please note that the figure also depicts a quadratic and a cubic transfer function that will be discussed in section 5.2.

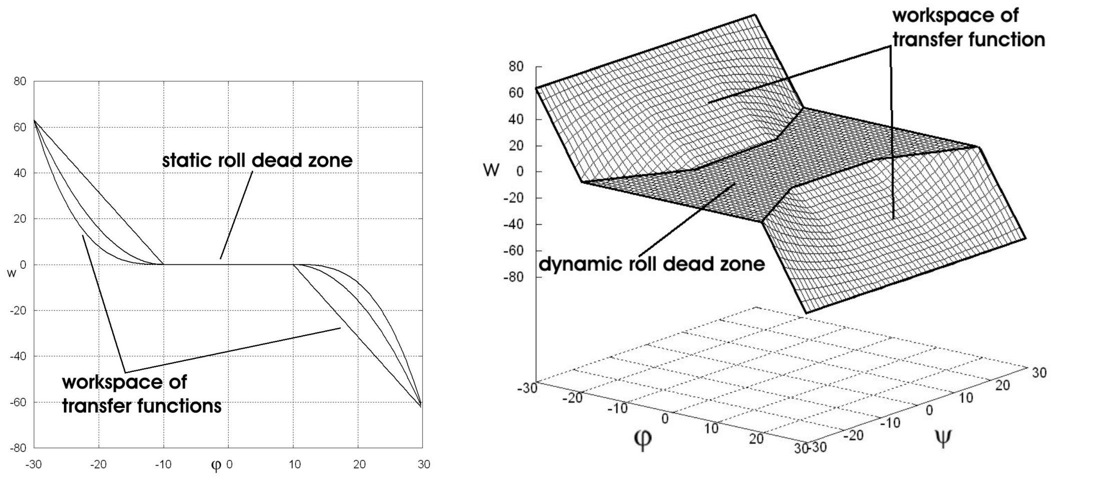
5 Design-Improvements for an IMU-based Head-Joystick

At the end of a first implementation phase we conducted an empirical study in which 15 participants tested a version of an IMU-based head-joystick that corresponds to the described system in section 4. This section now introduces several improvements relating to the algorithmic treatment of the IMU's head posture measurements resulting in an increased overall driving performance.

5.1 Dynamic Dead Zone for Head Roll Movements

A major shortcoming in the first implementation of an IMU-based head-joystick could be observed in situations of unrestricted straight ahead movement. While moving with high translational speed on an almost straight line, roll movements of the user's head that slightly exceeded the static bounds of the φ dead zone caused significant oscillations around the desired path. This effect can easily be identified by the comparison of Fig. 3(a) and Fig. 3(b). The upper part of both plots depict paths that have been executed in a straight corridor. In order to overcome the described problem, we have implemented a dynamic dead zone for the head's roll movements. The basic idea is to increase the clearance of unconsidered roll movements for driving situations with high translational speed, i.e. for situations where the user's head is far pitched up or down respectively. A reformulation of the roll dead zone is given in (4) and pictured in Fig.2(b).

$$\begin{aligned} \varphi_{valid} \in \left[\varphi_{max} \cdots \dot{\varphi}_0^+\right] \cup \left[\dot{\varphi}_0^- \cdots \varphi_{min}\right] \\ with\ t = c_{dz} \frac{|\psi| - \psi_0^+}{\psi_{max} - \psi_0^+} \left(\varphi_{max} - \varphi_0^+\right), \\ \dot{\varphi}_0^+ = \varphi_0^+ + t, \\ \dot{\varphi}_0^- = \varphi_0^- - t \end{aligned} \qquad (4)$$



(a) Linear, quadratic, and cubic transfer functions that map the head's roll angle φ onto the rotational velocity w. The φ dead zone ranges from $-10°$ to $10°$.

(b) Linear transfer function that maps the head's roll angle φ and its pitch angle ψ onto the rotational velocity w. We call this two-valued function a dynamic roll dead zone because it also constrains the validity of a given roll angle to the current pitch angle, that is actually deciding the translational speed v.

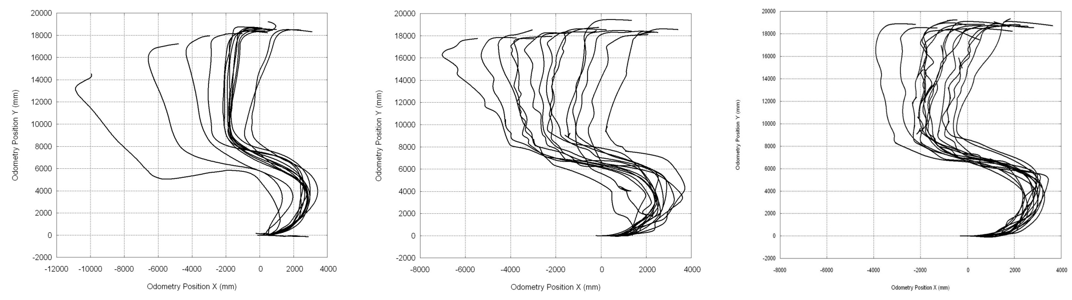
Fig. 2. Both plots show transfer functions for head roll movements, i.e. the functional dependency of the rotational velocity w from the head's roll angle φ in the left figure, and the dependency of w from the head's roll angle φ *and* its pitch angle ψ respectively.

5.2 Transfer Functions of Higher Order

Unintended slight head movements that exceed the pitch or roll dead zone, cause undesirable translational or rotational movements. Even if the formulation of a dynamic dead zone for the head's roll angle scales down this effect, there persists the basic necessity to reduce oscillations in the driven path. For this reason we have implemented transfer functions of higher order that weight, in contrast to a common proportional joystick, input angles by a quadratic or cubic transfer function respectively. Fig. 2(a) exemplarily shows a linear, a quadratic, and a cubic transfer function for the head's roll angle φ. It is easy to see that head movements throughout the whole workspace are weaklier assessed by higher order transfer functions than by linear ones.

6 Experimental Evaluation

The refined version of an IMU-based head-joystick has been tested in a further experimental evaluation phase. In analogy to the previous survey, we asked 15 participants to steer *Rolland* on an approximately $25m$ long s-shaped course in our laboratory. In particular we studied the test person's ability to hold the vehicle on a straight line course without causing intense oscillations. A first look



(a) Paths driven by a standard joystick. (b) Paths driven by the basic version of a head-joystick, cf. section 4. (c) Paths driven by the refined version of a head-joystick, cf. section 5.

Fig. 3. In each of the two evaluation phases we asked 15 test persons to navigate *Rolland* on an approximately $25m$ long s-like shape. All three plots show the driven paths in drifting odometry coordinates, whereby we can explain the strong deviations culminating in the aimed target at the upper right part of the plots.

on Fig. 3 reveals the paths that where driven by a common joystick and by the head-joystick in its two different stages of development. Although we expected the improved head-joystick version in Fig. 3(c) to show less variations from accurate straight ahead movement, minor problems in precise rotational control were still observable. A different point of view is given in Table 1. Compared with the basic implementation of the head-joystick, the refined version that applied a quadratic transfer function along with a dynamic roll dead zone, outperforms the basic version in terms of safety layer interventions. This metric predicates the driver's ability to safely manoeuvre along the given course. Confer section 3 for a brief account on the safety layer's operation mode.

7 Conclusion

For a special class of patients relying on electrical wheelchairs that are controllable without hand-use, we have implemented an interface based on a 3dof orientation tracker that is mounted at the back of the operator's head. We have shown that the evaluation of the user's head-posture is appropriate for controlling translational and rotational velocities of an automated wheelchair. Although the conducted experiments support this appreciation, their analysis leave several open questions. For example it remains unclear how actually handicapped people judge the proposed user interface. Therefor we have to conduct further long time experiments with the targeted audience. A second open question is whether problems in the rotational control can be solved by more sophisticated filtering techniques that are applied to the sensor's raw data. Finally it is worth to consider the user friendliness of a device that is attached to the back of the head and currently connected via a serial data cable. A final version for example should exchange the measured data with the computing unit via a wireless link.

Table 1. The table contrasts benchmark data of 15 participants that tested the autonomous wheelchair *Rolland* by using a common joystick, a basic implementation of an IMU-based head-joystick, and a refined version of the head-joystick. For a discussion of the recorded data confer section 6.

Criterion	**Common Joystick**	**IMU as Head-Joystick**	**Refined Head-Joystick**
ϕ time of travel [s]	30.73	55.03	61.78
ϕ length of travel [m]	22.45	25.03	26.88
ϕ average speed [m/s]	0.76	0.50	0.49
ϕ safety layer [ms] interventions	111.04	445.76	93.87

ACKNOWLEDGMENTS

This work has been partly funded by the Deutsche Forschungsgemeinschaft (DFG) in context of the Sonderforschungsbereich/Transregio 8 *Spatial Cognition.*

References

1. Mandel, C., Röfer, T., Frese, U.: Applying a 3dof orientation tracker as a human-robot interface for autonomous wheelchairs. In: Proceedings of the IEEE Intl. Conf. on Rehabilitation Robotics (ICORR). (2007) Submitted.
2. Lankenau, A., Röfer, T.: A safe and versatile mobility assistant. IEEE Robotics and Automation Magazine **8**(1) (2001) 29–37
3. Mandel, C., Hübner, K., Vierhuff, T.: Towards an autonomous wheelchair: Cognitive aspects in service robotics. In: Proceedings of Towards Autonomous Robotic Systems (TAROS). (2005)
4. Mandel, C., Frese, U., Röfer, T.: Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In: Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS). (2006)
5. Jaffe, D.L.: An ultrasonic head position interface for wheelchair control. Journal of Medical Systems **6**(4) (1982) 337–342
6. Ford, J.M.: Ultrasonic head controller for powered wheelchairs. Journal of Rehabilitation Research and Development **32**(3) (1995) 280–284
7. Chen, Y.L., Chen, S.C., Chen, W.L., Lin, J.F.: A head orientated wheelchair for people with disabilities. Disability and Rehabilitation **25**(6) (2003) 249–253
8. Canzler, U., Kraiss, K.F.: Person-adaptive facial feature analysis for an advanced wheelchair user-interface. In: Proceedings of the IEEE Intl. Conf. on Mechatronics and Robotics. (2004)
9. XSens Motion Technologies: Xsens inertial measurement unit product description and documentations (2006)

Comparison of Wheelchair User Interfaces for the Paralysed: Head-Joystick vs. Verbal Path Selection from an offered Route-Set

Christian Mandel* Udo Frese†
*University of Bremen, Department of Computer Science, P.O.Box 330440, 28334 Bremen, Germany
†DFKI Lab Bremen, Safe and Secure Cognitive Systems, Robert-Hooke-Straße 5, 28359 Bremen, Germany

Abstract— **Within this work we compare two human-robot interfaces that enable handicapped people who may not move their arms and legs to steer an automated wheelchair. As a technically more ambitious solution we first propose a simple speech recognition system that enables the paralysed to select a path from a set of spatially meaningful solutions, that is subsequently executed by a geometric path planner. The contrasting approach is given by a proportional head-joystick that evaluates the user's head posture with the help of a small-size three degrees of freedom orientation tracker. Both systems are assessed by the outcome of an empirical study in which untrained participants used the presented interfaces to navigate in a populated office environment.**

Index Terms— **Assistive Robots, Human Robot Interaction, Motion Planning, Navigation**

I. Introduction

Electrical wheelchairs give handicapped people the possibility to regain mobility in everyday life. Unfortunately, people suffering from certain spinal cord injury dysfunctions can only move body parts located above their shoulders, which leaves them unable to control a wheelchair via a common joystick interface. For this target group we propose the use of a simple speech interface with which the operator of an intelligent wheelchair selects a desired movement from an offered set of navigable routes, cf. Fig.2(a) for an illustration of potential routes in an outdoor environment. The proposed routes are derived from a frequently updated Voronoi graph that represents the immediate neighbourhood of the wheelchair. By means of a consistent denomination of spatially equivalent routes in terms of the ICAO-alphabet[1] we ensure that the operator has to command a new route only in situations where he wants to change the driving behaviour, e.g. entering a room after travelling along a corridor. The selected routes are executed by the combination of a geometric path planning module that computes obstacle-free cubic Bezier curves along with an underlying path- and velocity controller.

In order to assess the presented method, we compare it with a technically less complex approach of controlling an automated wheelchair by people suffering from paralysis of all four limbs. The system of contrast that has been first described in [12], [10] employs a three degrees of freedom orientation tracker mounted at the back of the operator's head for measuring its current posture. In analogy to a common joystick the user commands forward or backward movement by pitching his head down or up, and left or right movement by rolling his head left or right respectively.

The remainder of this paper is organized as follows. We begin in section II with an in-depth survey on human-robot interfaces, particularly focusing on the handling of autonomous wheelchairs. Section III then primarily describes our approach of controlling an autonomous wheelchair by verbal selection of proposed routes which are subsequently executed by an geometric path planning module. In the second part of this section we sketch the design of an head-joystick based steering approach for electrical wheelchairs. Both presented methods are compared in section IV by means of an experimental evaluation where we compare the participants ability to steer an autonomous wheelchair on a predefined course in a populated office environment. We conclude in section V with an assessment of the gathered results and an outlook on future work.

II. Related Work

The operation of electrical wheelchairs by disabled people through sophisticated interface techniques has its roots in the research field of human-robot interaction (HRI). In the following we give an outline on existing techniques that generally can be classified into systems that place the operator in-between the control loop of the vehicle, or those that interpret and execute abstract commands that specify more complex behaviours.

Within the first class of systems Jaffe proposes a head position interface [6]. It utilises two ultrasonic sensors that are mounted at the headrest of an electrical wheelchair in order to sense the posture of the operator's head within a two-dimensional plane parallel to the ground. The overall approach that has been evaluated within a clinical trial by 17 participants [3] maps the head's movement to proportional joystick-like signals. A comparable system that has been developed by Chen and colleagues [2] evaluates the output of two inertial tilt sensors mounted at the back of the user's head. By moving his or her head the user triggers discrete speed and steering commands, e.g. $70 cm/s$ translational speed when the head is

[1]By naming routes after the alphabet of the *International Civil Aviation Organization* we try to reduce speech recognizer failures since all words out of the alphabet are selected in a way that they are easily to distinguish, e.g. no monosyllabic words that have a mutual different sound.

moved forward once or $100cm/s$ translational speed when the head is moved forward twice respectively.

Within the second class of systems mentioned above falls the work of Canzler and Kraiss [1]. The authors describe the analysis of facial features like head posture, direction of gaze, and lip movement via a sophisticated computer vision system. Therewith they show the possibility to recognize at least four gesture dependent commands like *go*, *stop*, *left* and *right* for the user of an automated wheelchair. Control via the selection of fundamental behaviours is also implemented by the general computer interface *EagleEyes* that has first been described by Gips [4] and has been ported to the application scenario of commanding the robotic wheelchair system *Wheelesley* [18].

III. Analysed Wheelchair Interfaces

A. Control Via Path-Selection from an Offered Route-Set

In the following we describe the flow of procedures necessary for controlling an autonomous wheelchair by verbally commanded routes. The involved actions can be roughly subdivided into the representation of the environment along with the computation of navigable routes, and the execution of user selected routes.

1) Voronoi-Graph as Navigation-Domain: Our experimental platform, the autonomous wheelchair *Rolland* [11] cf. Fig.1, is equipped with two laser range finders that sense nearby obstacles to the front and to the back of the vehicle in an height of about $15cm$. In combination with the information coming from two incremental encoders that measure the speeds of the two actuated wheels, the system maintains the basic representation of its environment, the so-called *Evidence Grid*. Within this square array of cells that each stores the evidence that it is occupied by an obstacle, the wheelchair is always located in the center and rotated in correspondence to its real-world heading. The current implementation of the evidence grid, that is formally defined in (1), maintains a $7.5 * 7.5m^2$ grid out of $300 * 300cells$ resulting in an spatial resolution of $2.5 * 2.5cm^2$, cf. Fig.2(b) for an illustration.

$$EG := \{egc(x,y)\} : \begin{cases} egc(x,y) \; = \; 1 \; (\mathit{unoccupied}) \\ \quad \vdots \qquad\qquad \vdots \\ egc(x,y) \; = \; 256 \; (\mathit{occupied}) \end{cases} \tag{1}$$

The so-called *Distance Grid* is derived from the evidence grid and the next stride in the representation of the environment. Having the same dimensions and resolution as the evidence grid, the distance grid is calculated by a fast double sweep algorithm that computes for each free cell the metrical distance to the next occupied cell. In our current implementation of the distance grid, that is formally defined in (2), we treat a cell in the evidence grid as occupied when its value is greater than 128, cf. Fig.2(c) for an illustration.

$$\begin{gathered} DG := \{dgc(x,y)\} : \\ \forall egc(x',y') \in EG, egc(x,y) \in EG \\ \begin{cases} dgc(x,y) \; = \; 0 : \; \begin{matrix} egc(x,y) > 128 \\ \vee \quad \overrightarrow{(x,y)} = \overrightarrow{(x',y')} \end{matrix} \\ dgc(x,y) \; = \; min \mid \overrightarrow{(x,y)} - \overrightarrow{(x',y')} \mid : otherwise \end{cases} \end{gathered} \tag{2}$$

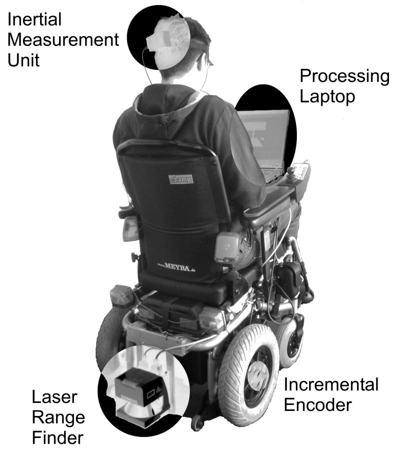


Fig. 1. The autonomous wheelchair *Rolland* along with its sensorial equipment and the processing laptop.

Computing the *Voronoi Graph* as the final stride of environmental representation requires two steps. First, we compute for each $dgc(x,y)$ out of DG whether the distance between its generating points, i.e. the occupied cells $egc(x',y')$ and $egc(x'',y'')$ out of EG that gave $dgc(x,y)$ its value, is greater than a given threshold c_d. Later on, this constant value will determine the minimal free space that is required to mark a region as navigable by an edge out of the Voronoi graph. In our current implementation we chose $c_d = 70cm$. Within the second step of computing the Voronoi graph, we search the *Voronoi Diagram* resulting from step one for points that are connected to only one or more than two neighbours. These points are finally inserted into the Voronoi graph's list of nodes V. The list of edges E consists of pairs of references to elements in V that are connected by points out of the Voronoi diagram. The Voronoi graph that is formally defined in (3) has been implemented within the general *RouteGraph*-framework [8].

$$\begin{gathered} \mathcal{VG} := (V,E) : \\ \begin{cases} VD := \{vd(x,y)\} : \mid \overrightarrow{(egc(x',y'))} - \overrightarrow{(egc(x'',y''))} \mid > c_d \\ V := \{v\} : \; \begin{matrix} v \in VD, & \mid neighbours(v) \in VD \mid = 1 \\ & \vee \mid neighbours(v) \in VD \mid > 2 \end{matrix} \\ E := \{e = (v_s, v_g)\} : v_s \in V, v_g \in V \end{cases} \end{gathered} \tag{3}$$

Within the current application scenario the system graphically presents the user navigable routes that are derived from the Voronoi graph defined above. For this purpose VG is first searched by an A*-algorithm for paths connecting the vertex closest to the current odometry position, i.e. $v_{start} \in V$, to any other vertex $v_{goal} \in V$. The resulting set of paths P is then filtered for paths whose goal v_{goal} is not included in any other path $p \in P$. Within the filtered set P' we only find navigable paths that direct to spatially emphasised targets that cover desirable in-between goals by the way. For a formal

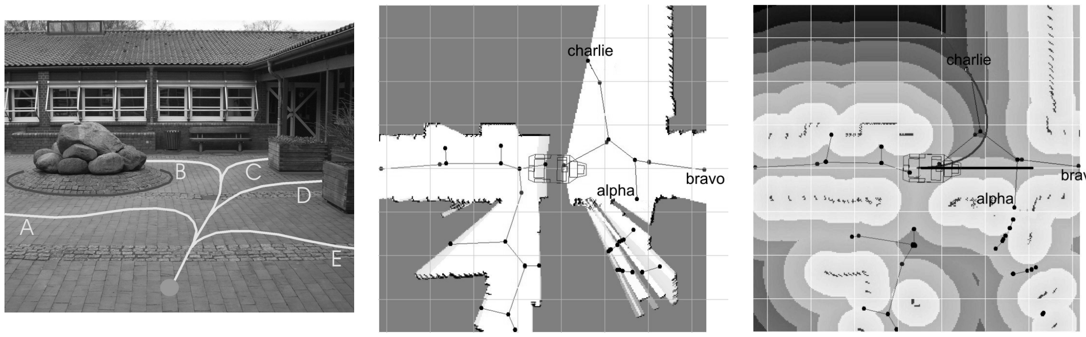


Fig. 2. The illustration of fundamental ideas behind the approach of commanding an autonomous wheelchair by verbal route selection from a set of offered routes ranges from Fig.2(a), the clarification of navigable routes in an unstructured environment, over Fig.2(b), the *Evidence Grid* as the basic environmental representation, up to Fig.2(c), the *Distance Grid* including the local *Voronoi Graph* along with three named routes and an cubic Bezier spline connecting the current odometry pose to vertex *charlie*.

definition of P and P' confer (4).

$$\begin{array}{lll} P & := & \{p\} : p := \{e_0, ..., e_n\}, e_0, ..., e_n \in E \\ P' & := & \{p_i\} : p_i \in P, v^i_{goal} \ni p_j \forall p_j \in P, i \neq j \end{array} \tag{4}$$

In order to offer the operator a verbally controlled selection mechanism of favoured paths, the target vertices of all paths $p' \in P'$ are now tagged by an unique name that must be easily recognisable by *Vocon*[13], the employed speech recognizer. We chose the earlier mentioned ICAO-alphabet (Alfa, Bravo, Charlie, ...) as the name-pool due to the promising properties of its elements for the recognition process. It finally remains crucial to assure that spatially equivalent paths are consistently named within consecutive processing cycles. This problem is complicated by a permanently altering Voronoi graph, caused by sensorial noise or changes in the environment and the evidence grid respectively. We tackle that issue by treating two paths as spatially equivalent only when the metrical distance between their target vertices is not greater than a given threshold. In the case of abrupt changes in the Voronoi graph that are caused by new obstacles, paths from former computation cycles may vanish and entire new paths may appear. In such cases the chosen approach stops the execution of the vanished path, and proposes the new appeared ones. This method seems to be justified since the topological change in the environment ultimately asks for a new driving command.

2) Autonomous Navigation via Bezier Curve Path Planner: The capability of our experimental platform Rolland to navigate autonomously in populated and unstructured environments is based on a geometric path planning approach that we first described in [9]. Within that work it was used to execute coarse verbal route descriptions such as: *"Leave the room, turn right, and go to the second room on the left."*. Basically, the chosen navigation method is based on the work of Hwang et al. [5], in which the authors give a general survey on navigation approaches that apply cubic Bezier curves to model obstacle-free paths within dynamic environments. The application of this kind of curve is appealing because of its spatial flexibility and the small number of determining control points $\vec{cp}_0, ..., \vec{cp}_3$, cf. (5) for a formal definition of cubic Bezier curves.

$$\begin{aligned} \vec{bc}(t) &= \vec{a}t^3 + \vec{b}t^2 + \vec{c}t + \vec{cp}_0,\ t \in [0..1] \\ \text{with } \vec{c} &= 3(\vec{cp}_1 - \vec{cp}_0), \\ \vec{b} &= 3(\vec{cp}_2 - \vec{cp}_1) - \vec{c}, \\ \vec{a} &= \vec{cp}_3 - \vec{cp}_0 - \vec{b} - \vec{c} \end{aligned} \tag{5}$$

Since the first and the last control point $\vec{cp}_0$ and $\vec{cp}_3$ determine the beginning and the end of the Bezier curve, they can easily be derived from the user-selected path $p_{sel} \in P'$ as the start-vertex v_{start} of its first edge e_0 and the goal-vertex v_{goal} of its last edge e_n respectively. The remaining control points $\vec{cp}_1$ and $\vec{cp}_2$ now span the basic search space over the cubic Bezier curves that

1) connect $\vec{cp}_0$, i.e. the current odometry position, with $\vec{cp}_3$, i.e. the user-selected goal of p_{sel},
2) are smoothly aligned with θ_s, i.e. the orientation at the current odometry position, in $\vec{cp}_0$ and with θ_g, i.e. the desired orientation in the user-selected goal of p_{sel}[2].
3) are obstacle-free in the sense that a robot-contour shifted tangentially along the curve does not intersect with any $egc(x, y) > 128 \in EG$.

In order to meet the the first and the second requirement, $\vec{cp}_1$ and $\vec{cp}_2$ are computed as points on vectors passing through $\vec{cp}_0$ and $\vec{cp}_3$, aligned to θ_s and θ_g respectively. The algebraic sign of the second summand in (6) denotes the desired direction of motion, whereas the upper sign designates forward motion and the lower one backward motion. In our current implementation we chose $l_1max = 5000mm$ and $l_2max = 5000mm$. With a fixed increment of $\delta = 75mm$, this yields an overall search space of about 4400 Bezier curves to test per frame.

$$\begin{aligned} \vec{cp}_1(l_1) &= \vec{cp}_0 \pm l_1 \begin{pmatrix} \cos(\theta_s) \\ \sin(\theta_s) \end{pmatrix},\ l_1max > l_1 > 0 \\ \vec{cp}_2(l_2) &= \vec{cp}_3 \mp l_2 \begin{pmatrix} \cos(\theta_g) \\ \sin(\theta_g) \end{pmatrix},\ l_2max > l_2 > 0 \end{aligned} \tag{6}$$

[2]The orientation in the user-selected goal of p_{sel} is given as the angle between the final edge e_n of p_{sel} and the x-axis of the odometry coordinate system.

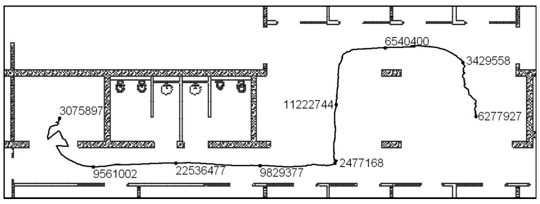


Fig. 3. The depicted trajectory is annotated with selected numbers of pixel-wise collision test operations that have been executed in order to check about 4400 Bezier curves for safety in each case.

Taking into account the third requirement that asks for obstacle-free paths, we compute for every Bezier curve out of (6) the time $t_{coll} \in [0...1]$ that indicates the first point of the curve where the shape of the robot, tangentially located on with its reference point, overlaps with an obstacle cell out of the evidence grid. Therefore any of the 4400 Bezier curves is discretized into 100 successive curve points on which we test a discretized contour of the wheelchair out of 132 contour points for collision. This yields an upper bound of about $O(58 \cdot 10^6)$ pixel-wise collision tests per computation cycle. In order to reduce the computational payload we basically skip the test of a contour cell cc' that is located nearby a contour cell cc from which we know that its minimal distance to the next obstacle is greater than the distance $\mid cc' - cc \mid$. To illustrate the result of this optimization process, Fig.3 shows an autonomously executed trajectory, as it was estimated by an adaption of the Monte Carlo localization approach used by the German RoboCup Team [15]. The driven path that corresponds to the task for several participants of our experimental evaluation phase, cf. sec. IV, is annotated by the number of completed pixel-wise collision tests. Along the approximately $25m$ long path we observed an average of $\approx 6.2 \cdot 10^6$, a minimum of ≈ 115000 and a maximum of $\approx 22 \cdot 10^6$ collision test operations per computation cycle.

We finally choose a solution of the presented path planing problem in fulfilment of requirements 1) - 3), that is obstacle-free and that minimizes the overall time of travel. The upper velocity-bound of the wheelchair at point $\vec{bc}(t)$ is therefore determined by the minimal distance between the robot-contour tangentially located at $\vec{bc}(t)$ to any obstacle-point, and the curvature $c(t)$. Once computed the final trajectory, it can be executed by the application of a path controller [14], that minimses the errors in orientation, translational velocity and lateral distance between the current wheelchair position and the closest pose on the trajectory.

B. Control Via Proportional Head-Joystick

In comparison to the preceding paragraph we will now describe our approach of controlling an automated wheelchair by interpreting the output of a miniaturized three degrees of freedom orientation tracker[3] as joystick-like signals. Therefore we will first detail the technical characteristics of the applied sensor along with its raw output data. Subsequently

[3]Throughout the rest of this paper we will abbreviate *three degrees of freedom orientation tracker* or rather *inertial measurement unit* by *IMU*.

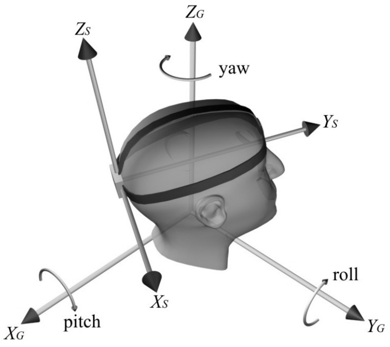


Fig. 4. The IMU that is mounted at the back of the user's head outputs drift-free *pitch*, *roll*, and *yaw* angles, i.e. the orientation of the device's local coordinate system S with respect to the fixed global coordinate system G.

we characterise the algorithmic treatment of the outputted 3d-angles, i.e. their conversion into translational and rotational speed commands.

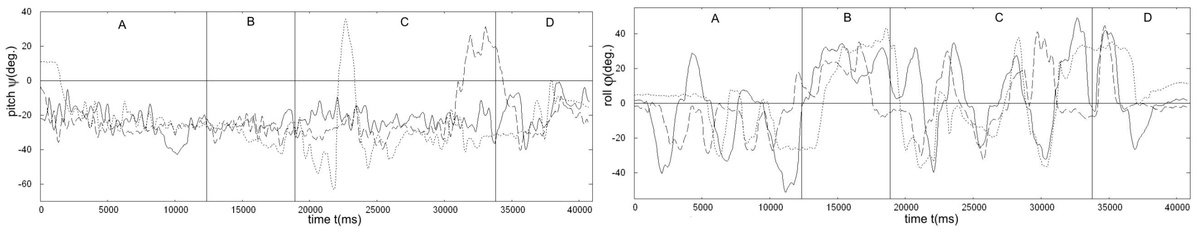
1) Measuring Head Posture Angles using an IMU: The *XSens MTx* IMU [17] is a small-scale electronical device that measures $53 \cdot 38 \cdot 21mm^3$ and weights about $30g$. With its included accelerometers, gyroscopes, and magnetometers the IMU not only outputs inertial information, namely 3D acceleration, 3D rate of turn, and 3D earth-magnetic field, but also drift-free absolute 3D orientation data. Attached to the back of the user's head with the help of an easy to wear frontlet, the IMU outputs Euler angles that describe the head posture by the rotation of the IMU's local coordinate system S with respect to the fixed global coordinate system G, cf. Fig.4. A single data reading of the IMU is given as the triple $\mathcal{I} = (\psi, \varphi, \theta)$ and more precisely formulated in (7).

$$\begin{aligned} \psi &= pitch = rotation\,around\,X_G \in [-90°...90°] \\ \varphi &= roll = rotation\,around\,Y_G \in [-180°...180°] \\ \theta &= yaw = rotation\,around\,Z_G \in [-180°...180°] \end{aligned} \tag{7}$$

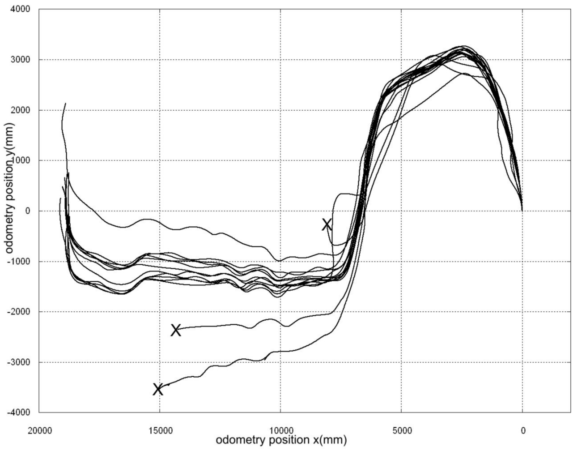
2) From Head Posture Angles to Steering Commands: The first step in the pipeline of processes that are necessary to convert a single IMU reading $\mathcal{I}$ into a pair of translational and rotational speeds $\mathcal{V} = (v, w)$ is the adoption of the minimal, maximal, and the mean deflection of the operator's head. For this purpose we ask the user to pitch and roll his or her head with maximal deflection at system start-up, while taking calibration measurements. The averaged limits ψ_{max}, ψ_{min}, φ_{max}, and φ_{min} now allow us to introduce a static dead zone around $\psi_0 = \frac{\psi_{max}+\psi_{min}}{2}$ and $\varphi_0 = \frac{\varphi_{max}+\varphi_{min}}{2}$. The dead zone as can be seen in (8) is later on used to filter unintended control commands that result from minor head movements.

$$\begin{aligned} \psi_{valid} \in \; & [\psi_0 - c_p \mid \psi_0 - \psi_{min} \mid ...\psi_{min}] \cup \\ & [\psi_{max}...\psi_0 + c_p \mid \psi_{max} - \psi_0 \mid] \\ \varphi_{valid} \in \; & [\varphi_0 - c_r \mid \varphi_0 - \varphi_{min} \mid ...\varphi_{min}] \cup \\ & [\varphi_{max}...\varphi_0 + c_r \mid \varphi_{max} - \varphi_0 \mid] \end{aligned} \tag{8}$$

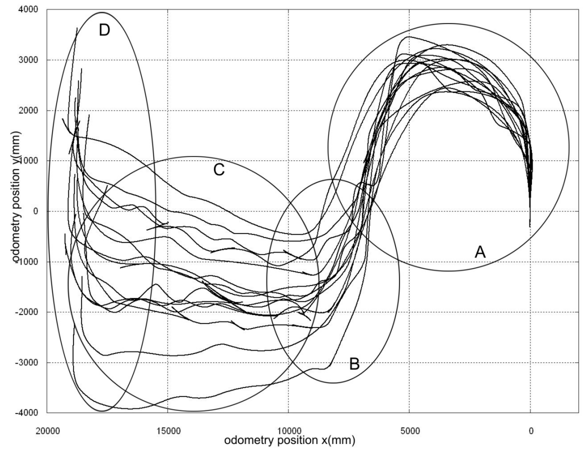
At this stage, a valid IMU reading $\mathcal{I}_{valid} = (\psi_{valid}, \varphi_{valid})$ neglects the yaw component θ since the unconsidered rotation around Z_G allows the user to look around while driving



(a) Head pitch angles ψ (left) and roll angles φ (right) of three selected participants developing during the navigation on an approximately 25m long s-like shape, cf. Fig.5(c). Note that curves are shifted and scaled in t.



(b) Autonomous executed paths commanded by 5 participants using the verbal selection mechanism presented in sec. III-A. Aborted test runs are marked by a terminating X.



(c) Paths driven by 15 participants using the head-joystick from sec. III-B.

Fig. 5. The above plots show experimental data that has been gathered during an evaluation phase where 15 participants employed the proposed head joystick, cf. sec. III-B, and 5 participants the introduced verbal control scheme repeatedly, cf. sec. III-A.

his or her vehicle. In order to finally compute a velocity command $\mathcal{V} = (v, w)$ for the wheelchair, we just multiply each component of $\mathcal{I}_{valid}$ by a weighting factor that includes the reciprocal of the corresponding interval's width from (8), and a normalizing component that maps v and w onto the velocity-domain of our particular vehicle.

IV. Experimental Evaluation

The two wheelchair interfaces presented in the preceding sections have been evaluated within an empiric test series that monitored the performance of a total of 20 untrained participants to navigate in an unstructured and populated office environment. We chose a constant setting that asked every test person to trace an approximately $25m$ long reference trajectory, comprising open space navigation in our office's coffe break area, corridor following, and a final narrow entrance passage, cf. Fig.3. Because of the conceptual difference in the two applied interfaces, we will now separately present the experimental findings.

A. Evaluation of Path-Selection from Offered Route-Set

Within Fig.5(b) one can see the odometry-based paths that five participants repeatedly drove by using the presented path-selection approach from sec. III-A. A first look on the plot reveals that three out of fiveteen test runs have been aborted due to diverse failures. In the case marked by the uppermost X, the subject abandoned its attempt because it had extreme problems to interpret the map which visually displayed the offered set of routes, cf. Fig.2(b). Albeit other participants reported the same problem as less significant, it must be considered to reduce the cognitive demands in reading the related map. A possible solution is not to rotate the icon for the wheelchair w.r.t. the physical heading, but instead to rotate the whole map content and letting the contour of the wheelchair always look upwards. The aborted test runs marked by the two lower X are due to a basic shortcoming of the offered paths underlying data, i.e. the Voronoi graph based on sensorial input. In both of the two addressed cases the wheelchair stopped because of a topological change in the path tracked before. The change that was triggered by the appearing passageway at the end of the reference trajectory was however so minimal that the subsequently offered paths rapidly alternated between a straight corridor path and a branchning turn. This problem rendered the choice of a new target path impossible for the participant. Future work has to eliminate this problem by the introduction of a more stable Voronoi graph. The work of Wallgrün et al. [16] tackles that issue by restricting a given Voronoi graph to relevant vertices that are generated by major features in the evidence grid.

B. Head-Joystick Evaluation

A first impression of the participants ability to manoeuvre Rolland on the given reference trajectory by using the described head-joystick is given in Fig.5(c). The paths that together start in the middle right part of the plot are depicted in drifting odometry coordinates, whereby we can explain the strong deviations in the aimed target. One of the primary observations, path oscillations especially on the straight line navigation sector in the lower left part of the plot, can be explained by the examination of three selected test runs in Fig.5(a). The two parts show the progression of the user's head pitch angle ψ that affects translational speed, and the roll angle φ that controls rotational speed, both over time t. Analysing part C of the roll angle plot that corresponds to the straight line corridor movement, we can see that the participants controlled this section by alternating head roll movements with a direction of up to $\pm 40°$. Although we believe that more experienced users are able to maintain precise straight ahead movement, a software solution in form of a so-called *Drive Assistant* would be appropriate for the unexperienced. This module would replace the actual *Safety Layer* that currently brakes in case of dangerous obstacles, by an obstacle-avoidance behaviour that alters unsafe driving commands smoothly by itself. At this point it shall be mentioned that we also observed high frequent but low amplitude oscillations of the head's roll angle φ within our first implementation. The concept of a *dynamic roll dead zone* that basically increases the clearance of unconsidered roll movements in driving situations with high translational speed eased this effect.

V. Conclusions

We have presented the implementation and experimental evaluation of two wheelchair user interfaces that allow the paralysed to control his or her vehicle without hand movements. In order to address a mutual comparison of both methods we first want to point out their conceptual difference. In [7] the authors distinguish human-robot interfaces into so-called *front-end approaches* that completely specify the task to be performed, and *incremental approaches* that decompose task descriptions into elementary actions. The head-joystick that falls into the later category comprises its main advantage of allowing for dynamic changes during the execution phase, i.e. the user is embedded in the control loop. In contrast to that, the verbal path selection approach benefits from the major advantage of the first class of interfaces in that it gives the operator maximum freedom during the autonomous execution phase.

Without paying attention to any particular user-preferences we carefully have to improve the shortcomings of both approaches. For the verbal path selection mechanism this is the already addressed stabilization of the Voronoi graph, and an improvement of the geometric path planner. The later issue primarily has to investigate the application of different types of curves in the case where Bezier curves do not adequately model specific navigation tasks, e.g. turn on the spot manoeuvres. Future work for the head-joystick interface will mainly involve long time experiments with the targeted audience from which we expect to find requirements for everyday service.

Acknowledgment

This work has been partly supported by the Deutsche Forschungsgemeinschaft (DFG) in context of the Sonderforschungsbereich/Transregio 8 *Spatial Cognition*. We also thank the Bremen members of the EU-project *SHARE-it* for their active support.

References

[1] U. Canzler and K.-F. Kraiss. Person-adaptive facial feature analysis for an advanced wheelchair user-interface. In *Proceedings of the IEEE Intl. Conf. on Mechatronics and Robotics*, 2004.
[2] Y.-L. Chen, S.-C. Chen, W.-L. Chen, and J.-F. Lin. A head orientated wheelchair for people with disabilities. *Disability and Rehabilitation*, 25(6):249–253, 2003.
[3] James M. Ford. Ultrasonic head controller for powered wheelchairs. *Journal of Rehabilitation Research and Development*, 32(3):280–284, 1995.
[4] J. Gips. On building intelligence into eagleeyes. In *Lecture Notes in AI: Assistive Technology and Artificial Intelligence*, 1998.
[5] Jung-Hoon Hwang, Ronald C. Arkin, and Dong-Soo Kwon. Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
[6] David L. Jaffe. An ultrasonic head position interface for wheelchair control. *Journal of Medical Systems*, 6(4):337–342, 1982.
[7] A. Knoll, B. Hildebrandt, and J. Zhang. Instructing cooperating assembly robots through situated dialogues in natural language. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997.
[8] B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T Mossakowski, and R. Ross. Specification of an ontology for route graphs. In *Spatial Cognition IV, Lecture Notes in Artificial Intelligence*, 2004.
[9] C. Mandel, U. Frese, and T. Röfer. Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In *Proceedings of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.
[10] C. Mandel, U. Frese, and T. Röfer. Design improvements for proportional control of autonomous wheelchairs via 3dof orientation tracker. In *Proceedings of the International Work Conference on Artificial Neural Networks (IWANN)*, 2007. To Appear.
[11] C. Mandel, K. Hübner, and T. Vierhuff. Towards an autonomous wheelchair: Cognitive aspects in service robotics. In *Proceedings of Towards Autonomous Robotic Systems (TAROS)*, 2005.
[12] C. Mandel, T. Röfer, and U. Frese. Applying a 3dof orientation tracker as a human-robot interface for autonomous wheelchairs. In *Proceedings of the IEEE Intl. Conf. on Rehabilitation Robotics (ICORR)*, 2007. Submitted.
[13] Nuance Communication Technologies. Nuance vocon speech recognizer product description and documentations, 2007.
[14] G. Oriolo, A. Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: Design, 2002.
[15] T. Röfer and M. Jüngel. Vision-based fast and reactive monte-carlo localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.
[16] Jan Oliver Wallgrün. Autonomous construction of hierarchical voronoi-based route graph representations. In Christian Freksa, Markus Knauff, Bernd Krieg-Brückner, Bernhard Nebel, and Thomas Barkowsky, editors, *Spatial Cognition IV. Reasoning, Action, Interaction: International Conference Spatial Cognition 2004*, volume 3343 of *Lecture Notes in Artificial Intelligence*, pages 413–433. Springer; Berlin; http://www.springer.de, 2005.
[17] XSens Motion Technologies. Xsens inertial measurement unit product description and documentations, 2006.
[18] H.A. Yanco. Wheelesley, a robotic wheelchair system: Indoor navigation and user interface. In *Lecture Notes in AI: Assistive Technology and Artificial Intelligence*, 1998.

Non-invasive Brain-Computer Interfaces for Semi-Autonomous Assistive Devices

Authors: Bernhard Graimann[1], Brendan Allison[2], Christian Mandel[3], Thorsten Lueth[4], Diana Valbuena[5], Axel Graeser[6]

[1]Institute of Automation, University of Bremen, and BCI Lab, Graz University of Technology, graimann@iat.uni-bremen.de
[2]Institute of Automation, University of Bremen, allison@iat.uni-bremen.de
[3]Institute of Computer Science, University of Bremen, cmandel@uni.bremen.de
[4]Institute of Automation, University of Bremen, lueth@iat.uni-bremen.de
[5]Institute of Automation, University of Bremen, valbuena@iat.uni-bremen.de
[6]Institute of Automation, University of Bremen, ag@iat.uni-bremen.de

1.1 Introduction

Ever since the first human brain waves were recorded by the German scientist Hans Berger in the late 1920s, it was speculated that this EEG technologies could be used to decipher thoughts or control external devices. The idea of using brain waves to produce control signals – to establish a direct interface between the human brain and a machine or computer – also appeared early in the science fiction genre. Early television series like Star Trek and movies like Firefox envisaged such applications. At that time, limitations in computing power, signal processing techniques, and cognitive neuroscience kept such ideas in the realm of science fiction. However, improvements in these technologies have led to the first publications about brain-computer communication in the early 1970's, which demonstrated a proof of principle [Vid73]. About 15 years later, ongoing improvements in relevant technologies paved the way for brain-computer interfaces that could yield more than just proof of principle results [FD88, WMN91].

Direct communication from man to machine has now advanced well beyond science fiction. Brain-computer interfaces (BCIs), which are tools that allow communication and control via direct measures of brain activity, have been widely described in the literature. Most BCI research efforts focus on developing communication tools for people who need them most – persons with severe physical disabilities that prevent them from communicating via other means [WBM02]. However, although BCIs exhibit serious drawbacks relative to other interfaces, very new research developments suggest that BCIs may soon become much more powerful, flexible, usable tools, providing improved communication to severely disabled users and opening new applications and new user groups [AWW07, AGG07, BC07, GAG07].

In this chapter, we will first review the foundations of modern BCI systems. The next section presents a discussion of the limitations of current BCIs. After a general representation about BCI signal processing, we present the Bremen SSVEP BCI as an example of an easily applicable and robust BCI. Further, we introduce new BCI applications that are about to become possible through intelligent assistive devices such as rehabilitation robots. The last section will discuss future BCI directions. We show how the BCI technologies developed at Bremen, as well as other recent research developments, can compensate for many BCI limitations and adumbrate new BCI users and applications.

1.2 Modern Brain-computer interfaces

1.2.1 Mental strategies and brain patterns

A BCI detects, interprets and classifies specific patterns of activity in ongoing brain signals that are associated with specific intentions, tasks or events. These tasks or events can be either exogenous or endogenous. Correspondingly, one may differentiate between exogenous and endogenous BCIs [KB05]. Exogenous systems rely on activity elicited in the brain by external stimuli, such as visual evoked potentials or auditory evoked potentials. In contrast, endogenous systems do not require an external stimulus. Users learn to produce specific patterns at will. Through neurofeedback training, users learn to voluntarily modulate specific brain patterns. The length of the training is naturally subject-dependent, but it also depends on the mental strategy used, feedback and pattern classification parameters, the subjects'goals, and the training environment.

Most BCIs rely on either *selective attention* or *motor imagery*. Commonly used brain patterns are the SSVEP (steady state visual evoked potential), P300 (the 300ms component of an evoked potential), and ERD/ERS (event-related de-synchronization). In a selective attention task, the user focuses attention on a particular stimulus. The stimulus may oscillate at a frequency from 5-50Hz, or flash infrequently. Rapidly oscillating stimuli produce SSVEP activity in the brain at corresponding frequencies. This SSVEP activity becomes larger if subjects focus attention on a specific stimulus, and thus SSVEP BCI users can communicate which stimulus is of interest to them [MMC00]. Transient stimuli that a user silently counts (rather than ignores) will produce a P300 [DSW00]. Motor imagery BCIs work because ERD/ERS activity changes when people imagine or perform movement. Thus, people can imagine different movements and thereby produce ERD/ERS changes that convey user intent [PMS06]. All BCI patterns can be characterized in the time and frequency domain as well as according to the topography. The relation between mental tasks and the corresponding brain patterns produced are shown in the concept map in Figure 1 .

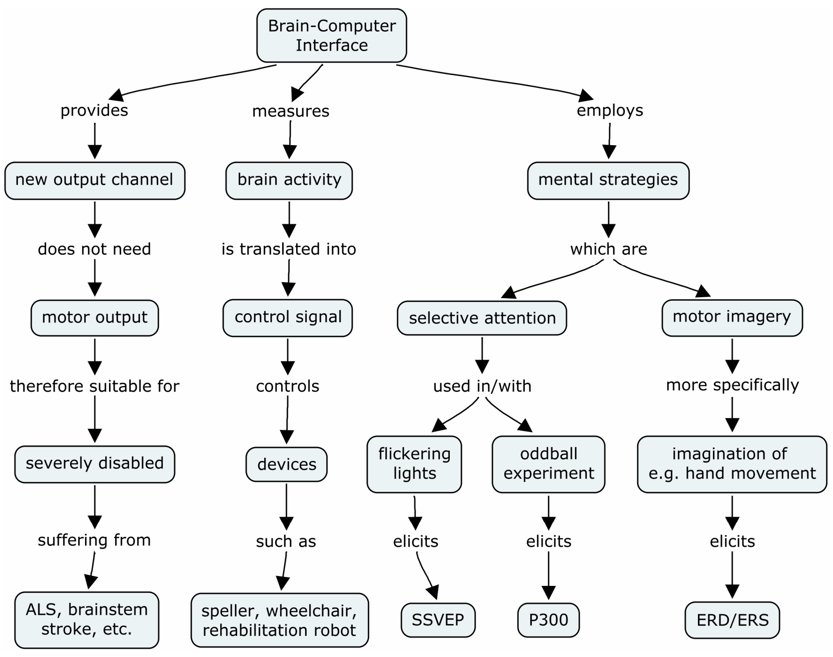


Figure 1: Concept map – Brain-Computer Interface

The main difference between BCIs and other human-computer interaction systems is that BCIs require no muscle activity or physical movement. Unlike other systems, which require muscle activity, BCIs provide “non-muscular” communication or require only little muscular activity such as gaze shifting. One of the most important reasons that this is significant is that the main application of current BCI systems is to provide assistive devices for people with severe disabilities. Some people have a medical condition called locked-in syndrome (caused by diseases that affect the neuromuscular pathways such as brainstem stroke or amyotrophic lateral sclerosis) and thus have lost all ability to communicate. Their cognitive functions, however, are completely intact. BCIs provide those patients the only possibility to interact with their environment. Although allow only limited communication capabilities, as pointed out in the next section, because of absence of alternatives, it is utmost importance for people with such severe disabilities.

1.2.2 Limitations of Brain-Computer Interfaces

Most non-invasive BCIs measure brain signals via electrodes mounted on the scalp. These signals are stochastic and nonstationary, have a small signal-to-noise ratio, a poor spatial resolution, and are very susceptible to external and internal interferences. Moreover, the underlying probability distributions are unknown and the data samples available for machine learning and statistical inference are usually very limited. Therefore, robust signal processing methods are needed to reliably detect and classify voluntarily generated brain patterns. Although state-of-the-art signal processing methods are applied in BCI research, the output of a BCI is still unreliable, and the information transfer rates are very small compared to conventional human interaction interfaces such as keyboards and mice.

BCIs are notorious for poor information throughput. Although online BCI systems have exhibited performance slightly above 60 bits per minute [GXC03], such performance is not typical of most users in real world settings. There is often considerable variation across subjects and BCI usage sessions. Further the information transfer rate also depends on the mental strategy employed. Typically, BCI with selective attention strategies are faster than those using, for instance, motor imagery. Unfortunately, there is no study available that investigated the average information transfer rate for various BCI systems over a larger user population. However, it has to be assumed that these averaged values would be considerably lower than the optimal single user performances reported in recent publications, which are between 30 and 60 bits per minute [BDK07, FVG07, GXC03]. Further it has to be noted that a minority of subjects exhibit little or no control [BDK07, GEH03]. The reason is not clear but even long sustained training cannot improve performance for those subjects.

Information throughput, in BCIs or any communication channel, depends on the number of selections available to the user, the probability of correctly identifying the selection that corresponds to user intent, and the number of selections possible per minute. Although some BCIs present subjects with only two selections [e.g., WMN91, PGN06], other BCIs allow 36 or more selections [AP03, SKM06]. BCI accuracy varies considerably, but modern BCIs can attain performance above 85% with most subjects. Further, any BCI can improve accuracy at the expense of selections per minute by using longer periods of brain activity or response verification [MSW03]. BCIs typically offer about 10 selections per minute, though this varies widely as well.

Both the hardware and software currently available for brain-computer communication is more suitable for experiments in the lab than for practical applications in real-world environments like the users home. The hardware needed for an EEG BCI requires a trained expert to precisely position the EEG cap, scrape the skin where each electrode will go, apply gooey electrode gel, further abrade the skin, and continue this process until all electrodes – often a large number of electrodes is necessary – produce a clean signal. This process is not painful, but is not exceptionally pleasant. After each BCI usage session, the cap and the user’s hair must be washed. BCIs not only require an expert to help setup the necessary hardware, but also to configure and adapt key software parameters [AWW07]. Hence, BCI users are dependent on expert help to setup, clean, and configure their BCI. The conclusions and future prospects section below presents some avenues toward making BCIs more intelligent, self–adaptive, reliable, and practical.

1.2.3 BCI signal processing

The building blocks of a BCI are signal acquisition, signal processing, and the application interface. Figure 2 shows the basic scheme of a BCI. A BCI analyzes and assesses the activity of the brain and transforms this activity, often through complicated signal processing techniques, into control signals that operate assistive devices. Brain activity suitable for a BCI can be monitored or recorded in various ways. The recording methods can be invasive or non-invasive and can record electrical activity or metabolic brain responses. The most prevalent non-invasive method is the EEG, which records the brain's electrical activity. EEG recording is relatively simple, readily available, inexpensive, and non-invasive. However, it also has some disadvantages such as poor signal-to-noise ratio and very limited spatial resolution. Invasive methods such as intracortical recordings or electrocorticogram (ECoG) alleviate the limitations of the EEG. The ECoG, which records activity from the surface of the cortex, and unlike intracortical recordings does not penetrate brain tissue, is considered to be moderately invasive. The spectro-temporal properties of ECoG signals are similar to EEG signals, but the signal quality in terms of SNR and spatial resolution is much better. Besides EEG and ECoG, intra-cortical recordings capturing spiking activity from single neurons or neuron assemblies or methods for measuring metabolic activation (blood oxygenation) are other ways of recording brain activity. Here, the focus lies on EEG, because this is the most common and practical tool for BCIs. Typically, BCI signal processing is subdivided in preprocessing, feature extraction, and detection or classification.

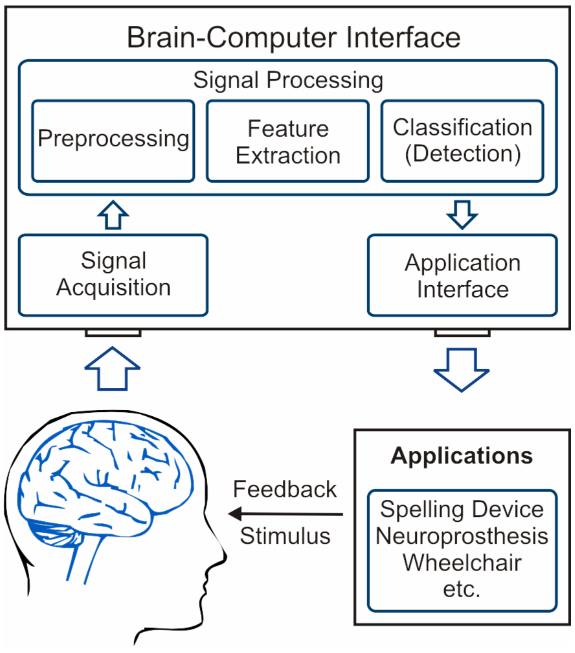


Figure 2: Basic scheme of a brain-computer interface (modified from [Gra06])

Preprocessing

Preprocessing aims at improving the spatial resolution and signal-to-noise ratio (SNR) of the recorded signals. Spatial filters, which instantaneously linearly combine samples of different channels, are often used in BCI preprocessing. Orthogonal source derivation is a classical spatial filter in EEG processing. Due to its simplicity and efficacy, the small Laplacian, a simplified form of orthogonal source derivation, is often used in ERD/ERS based BCIs [PGN06, MMD97]. More advanced spatial filters can be derived from principal component analysis (PCA) or independent component analysis (ICA). PCA seeks uncorrelated components with maximal variance, while ICA tries to decompose signals into statistically independent components. ICA, PCA, and combinations of both methods have been applied to BCIs based on SSVEP, P300, and ERD/ERS [FVG07, HTH07, NBL06). A method that has been applied particularly successfully to discriminate ERD/ERS patterns is Common Spatial Patterns, CSP [RMP00, DBC04, GP06] and its extensions [LBC05, DMH07]. In contrast to previously mentioned spatial filters, CSP is a supervised method and thus requires labeled data (class information). It is based on the simultaneous diagonalization of two matrices—the variance matrices of the two populations—and finds projections with the biggest difference in variance between the two classes.

Feature Extraction

The second step in the BCI signal processing chain is feature extraction. Sometimes preprocessing and feature extraction are seen as one unit. The reason is that both preprocessing and feature extraction have the same aim, to simplify the problem for the subsequent detection or classification task. Basically, both units are supposed to improve signal quality, i.e. improve signal-to-noise ratio and spatial resolution, by extracting discriminant information from the signal while simultaneously eliminating redundant and disturbing information. However, the signal processing methods for preprocessing and feature extraction are considerably different, and therefore they are treated as individual blocks here. Most feature extraction methods employed produce features characterized either in the time or in the frequency domain and their statistics is of first or second order (i.e. mean or variance) [PGN06]. Examples are simple averaging for extracting evoked potentials or calculating the variance (power) of the signal over the entire signal or specific frequency ranges. Variance features are derived from simple bandpass filters, wavelet filters, or of parametric and non-parametric spectrum estimators [GHL04, PGN06].

Detection and Classification

A BCI essentially transforms the brain signal into a control signal. This transformation can be seen as a mapping from the high-dimensional input signal to a signal of considerably reduced dimension. In this view, a BCI is basically a classifier that maps the input to classes of which each class corresponds to a command. Robust behavior is an essential feature in BCI signal processing. This is the reason why linear classifiers such as Fisher linear discriminant (FDA) or support vector machines (SVM) with linear kernels are most often used in BCIs [MKD04, PGN06]. They are less prone to outliers and show better generalization than non-linear classifiers. Both FDA and SVM are binary classifiers, but can be easily extended to multiple classes by applying a one-vs-one or one-vs-rest scheme. In order to further reduce overfitting and improve the robustness of the classifiers, regularization has been suggested for the training of BCI classifiers (Mueller, 2004).

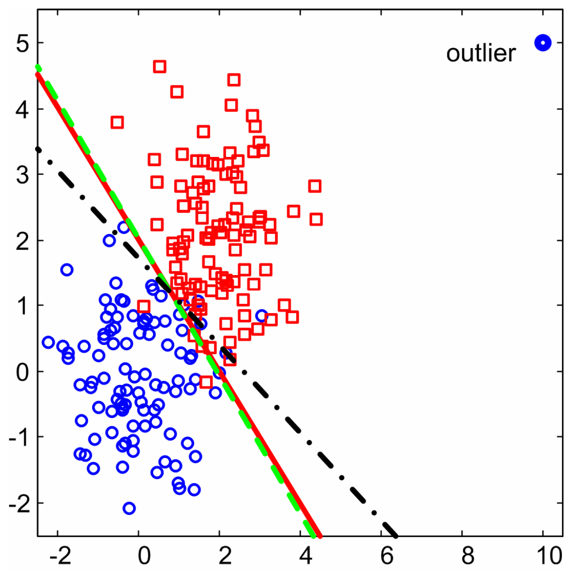


Figure 3: Influence of outliers on Fisher linear discriminant analysis as classifier.

The influence of outliers is demonstrated in Figure 3. It shows samples of two Gaussian distributions with the same variance but different means. The solid straight line represents the optimal linear decision function, the dashed line is the result of FDA whereas the outlier in the upper right corner has been neglected, and the dotted line represents the result of FDA without neglecting the outlier. Obviously, the impact of the outlier is large and can change the decision function represented by the straight lines in Figure 3 considerably. The standard way to calculate FDA is to maximize the ratio of the between-class variance to the within-class variance

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}, \tag{1.1.1}$$

where **w** is the weight vector (linear projection vector), m_c and s_c denote the means and standard deviations for each class, $c \in \{1,2\}$, and $\mathbf{S}_\mathrm{B}$ and $\mathbf{S}_\mathrm{W}$ are the between class-covariance matrix and the within-class covariance matrix, respectively. The solution for this optimization problem is [Fuk90]

$$\mathbf{w} = \mathbf{S}_W^{-1}(m_2 - m_1). \tag{1.1.2}$$

Applying this solution on the data shown in Figure 3 gives the defective result represented by the dashed line if the outlier is included in the data. In the presence of outliers, a more robust result can be obtained when regularization is used. A possible regularization when the problem is defined as a mathematical programming approach [MAB03]:

$$\min_{\mathbf{w},b,\xi} \|\mathbf{w}\|_2^2 + \frac{\lambda}{K}\|\xi\|_2^2 \quad \text{subject to the constraint} \qquad (1.1.4)$$
$$y_k(\mathbf{w}^T\mathbf{x}_k + b) = 1 - \xi_k \quad \text{for } k = 1,...,K$$

with ξ the slack variables, $\mathbf{x}_k$ the k-th sample vector (feature vector) of K samples, b the bias that defines together with $\mathbf{w}$ the decision boundary. The slack variables are implicitly determined by the quadratic programming machine, which is usually used to solve this constrained optimization problem. The regularization coefficient λ, however, has to be determined for each data set explicitly. It is a so called hyperparameter. Like other parameters as well, hyperparameters have to be adjusted to the data, but there are no direct optimization methods to do that. They are usually determined by an iterative process embedded in a cross-validation procedure, where the results on a subset of the data (training set) are cross-validated with the results obtained on another subset (test set).

Parameter optimization and evaluation

The parameters used to identify and classify brain activity in a BCI are typically calculated from off-line data. That is, the setup of the classifier is done on data recorded in a previous BCI session. The new classifier determined in the off-line analysis is then used in the next session. In order to obtain good generalization the classifier has to be validated carefully. The data has to be divided into training and test set. All parameters including the hyperparameters of a method has to be determined on the training set, the evaluation has to be done on the test set. It is essential that the test data is not involved in any way before the parameters are defined, otherwise the generalization would be compromised and less robust results would be achieved [MAB03, DMH07]. One way to perform proper cross-validation is to split the data in three disjunct sets: training, validation, and test set. The parameters are determined on the training and validation set, and the final evaluation, i.e. the estimation of the generalization error, is performed on the test set. In BCI signal processing, however, the available data for the training of classifiers is typically very limited. It is important to note that the absolute number of samples available is not important, but rather the number of independent data samples is important. In a recording sampled with 1000 Hz where the subject performed 50 mental tasks, the amount of available and statistically independent data for training a classifier is essentially the same as in the recording sampled with 250 Hz. It is the number of trials that is important and not the number of samples provided the sampling theorem is satisfied. This is the reason why splitting the data in three portions is seldom an option in BCI signal processing. A solution for this dilemma is the use of nested cross-validation [MKD04, DMH07]. In this procedure the training set is again split in inner-training and inner-test sets. On these inner sets an inner cross-validation is performed to determine the parameters of the signal processing approach. The generalization error is determined on the normal (outer) cross-validation. In this way, a robust set up of the parameters can be achieved.

1.3 A robust BCI based on SSVEP

A high information transfer rate is very important in BCIs. Apart from that, there are also other characteristics of a BCI which are very important for a robust and practically applicable BCI:

- The BCI system should be less prone to artifacts and external interferences.
- The number of electrodes required should be small.
- An exact electrode placement is not necessary.
- The number of parameters that have to be optimized for each individual user should be as small as possible.
- The system should account of the inherent non-stationarity of brain signals.

Almost all BCI systems currently available combine only one or two of these practical features. One notable exception is the Bremen BCI which is based on SSVEP and implements a robust signal processing methodology that provides high information transfer rates and also the five desirable characteristics mentioned above [FVG07, FLV07].

As already mentioned previously, SSVEP patterns are the result of a visual selective attention task in which BCI users focus their attention on light sources that flash or flicker with particular frequencies above 5 Hz. The SSVEP is best described in the frequency domain, since it consists of frequency components that have the same fundamental frequency as the stimulation but also includes higher harmonics. This is seen in Figure 4A where the spectral density of a 20 second period with repetitive stimulation with a flickering light of 11 Hz is shown. The SSVEP is clearly visible at the fundamental frequency of 11 Hz and the harmonic frequencies at 22 and 33 Hz. In this case the SSVEP could be discriminated from the neighboring frequency components that indicate the ongoing background activity. However, in a real-time BCI, the user cannot wait 20 seconds for a response. The processing time frames have to be much shorter. The spectral density depicted in Figure 4B is calculated from a 1 second period and shows that the discrimination is not so simple anymore. In this case, more complicated signal processing methods have to be employed to achieve robust detection of the SSVEP components and also to discriminate between different SSVEP patterns evoked from different stimulation frequencies.

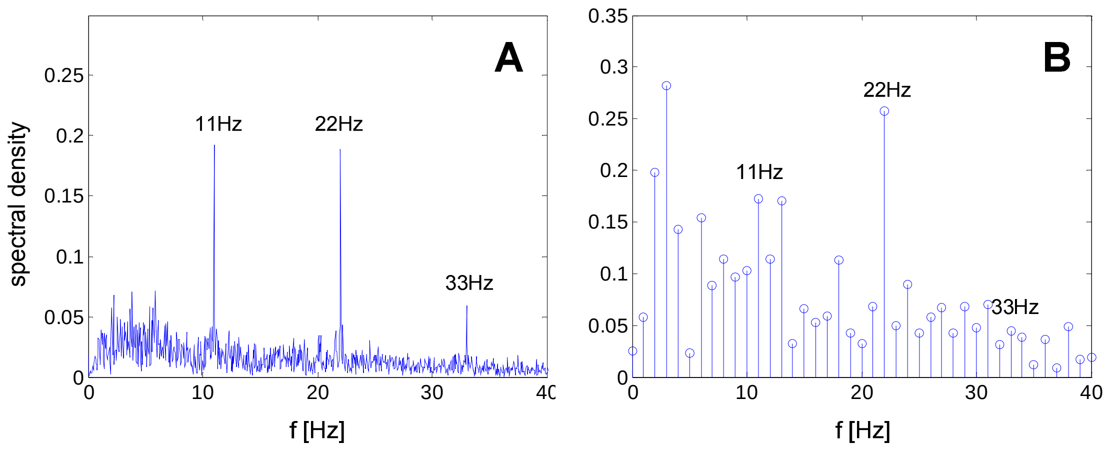


Figure 4: Steady state visual evoked potentials. Fig 4A shows the spectral density of a 20 second period. Fig. 4B shows the spectral density of a 1 second period.

In a typical BCI application, an array of flickering lights with different frequencies is used as stimuli. Each flickering light and its corresponding SSVEP are associated with a certain control command of the BCI application. The task of the BCI system is to reliably detect SSVEPs in the ongoing brain signal and also to determine its frequency in order to identify which command the user wants to convey. For high information transfer rates three parameters are important: the accuracy of detection, the speed of detection, and the number of frequencies that can be discriminated. These three parameters are not independent of each other. In fact, they are closely related. As indicated in Figure 4, shorter time segments speed up the detection time but also reduce the accuracy. Reduced accuracy also influences the number of frequencies that can be differentiated reliably. Instead of trying to improve each of these three parameters individually, the information transfer rate can also be improved by increasing the signal-to-noise ratio in general.

The Bremen BCI implements such an approach. The brain signal is modeled as a composite of SSVEP response, background brain activity, and noise. Short time principle component analysis is used to create a spatial filter that linearly combines the signals of all electrodes in a way that the background activity and noise is minimized. The algorithm is briefly outlined in the following. More details can be found in [FVG07].

The assumed model of the brain signal of a channel *i* is

$$y_i(t) = \sum_{k=1}^{N_h} a_{i,k} \sin(2\pi k f t + \phi_{i,k}) + \sum_j b_{i,j} z_j(t) + e_i(t)\,, \qquad (1.1.5)$$

where the first part is the SSVEP modeled as sinusoidals with the fundamental frequency *f* and its harmonics *k·f* (*k*>1), and the corresponding amplitude $a_{i,k}$ and phase $\phi_{i,k}$. The second part describes the background activity as a sum of background brain activity and nuisance signals. The last part describes the noise component in the measurement.

Assuming a time frame of N_t samples and a number of N_y channels the model can be written as [FVG07]:

$$\mathbf{Y} = \mathbf{XA} + \mathbf{ZB} + \mathbf{E}, \tag{1.1.6}$$

where $\mathbf{Y} = [y_1, ..., y_{N_y}]$ is a N_t x N_y matrix with the channel signals in the columns, **X** contains the sine and cosine pairs from the SSVEP model in its columns, **Z** the background activity, **A** and **B** contain the corresponding amplitudes, **E** is the noise matrix.

In order to minimize the activity that is not related to the SSVEP, a linear combination is sought that reduces the variance in the background activity and in the noise component of Equ. 1.1.6. To this end, any potential SSVEP activity is removed from the electrode signals. This is done by orthogonal projection.

$$\tilde{\mathbf{Y}} = \mathbf{Y} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{1.1.6}$$

The remaining signal $\tilde{\mathbf{Y}}$ contains approximately only background activity and noise, i.e.,

$$\tilde{\mathbf{Y}} \approx \mathbf{ZB} + \mathbf{E}. \tag{1.1.6}$$

The linear combination, i.e. the weight vector **w**, for minimizing the variance of $\tilde{\mathbf{Y}}$ is found by optimizing

$$\min_{\mathbf{w}} \left\| \tilde{\mathbf{Y}}\mathbf{w} \right\|^2 = \min_{\mathbf{w}} \mathbf{w}^T \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}} \mathbf{w}, \tag{1.1.6}$$

which has the solution in the eigenvector that corresponds to the smallest eigenvalue of the covariance of $\tilde{\mathbf{Y}}$. The solution is known as principal component analysis (PCA). However, in PCA the components with the largest variance are usually used. Here, the components with the smallest variances – also know as the minor components – are needed. To increase the robustness, not only the eigenvector of the smallest eigenvalue but also those eigenvectors of the next largest eigenvalues, so that altogether about 10% of the variance of the data is included, are used to construct the spatial filter **W**. The spatially filtered signal is then

$$\mathbf{S} = \mathbf{YW}. \tag{1.1.6}$$

The test statistic which is an average of the signal-to-noise ratio over all N_s spatially filtered components and all N_h SSVEP frequencies (including harmonics) is calculated by

$$T = \frac{1}{N_s N_h} \sum_{l=1}^{N_s} \sum_{k=1}^{N_h} \frac{P_{k,l}}{\sigma_{k,l}^2}. \tag{1.1.6}$$

$P_{k,l}$ is the power in the *k*th SSVEP harmonic frequency in the spatially filtered signal *l*; $\sigma_{k,l}^2$ is the corresponding estimate of the background and noise activity.

In a BCI application with several stimuli, a test statistic for each fundamental frequency of the flickering light is calculated from data blocks of a length of 1 second. Each statistic is compared with an empirically determined threshold. If a test statistic exceeds the threshold, then the corresponding frequency is detected and the control command associated with this frequency is initiated.

This algorithm is simple to apply and gives reliable and robust results. As shown in [FVG07] only a small number of channels (6 channels or even less) is necessary to achieve good results. The exact electrode placement is not critical, since the spatial filter is adaptively calculated for each data block. This is also the reason why the algorithm is able to account for the non-stationarity in the data. The influence of artifacts is small, because the algorithm considers only information from very narrow frequency bands (SSVEP fundamental frequency and harmonics). Also, only one parameter, the detection threshold, has to be determined. This is straightforward and typically only requires less than a minute. Most importantly, however, the algorithm is also able to achieve high information transfer rates. This has been shown in [FLV06], where the system was applied in two different BCI spelling tasks. In this study, the subjects navigated the cursor in a matrix like arrangement of letters or selected columns and rows of a letter matrix by selective attention on five flickering lights. Each light encoded either cursor movement commands (left, right, up, down, and select), or the indices of the columns or rows of the letter matrix. Eleven subjects participated in this study and produced information transfer rates between 11 and 43 bits/min, with an average of about 27-30 bits/min. The average classification rate was about 97.5%. The best subject was able to write 9 letter per minute. Although higher information transfer rates have been reported [GXC03], these results are remarkable since no subject specific optimization of the BCI was performed and only five flickering lights were used. The information transfer rates could be easily improved, at least for those subjects showing very prominent SSVEP response, by increasing the number of flickering lights.

1.4 New semi-autonomous applications for BCIs

Most BCIs have been used to control computer applications such as spelling devices, simple computer games, limited environmental control, and generic cursor control applications [PGN06, SD06, BDK07]. Only a limited number of systems have also allowed control of more sophisticated devices, including orthoses, robotic arms, and mobile robots [AWW07, MSP05, MEM04]. It appears that BCIs can control any application that other interfaces can control, provided these applications can function effectively with the low information throughput of BCIs. BCIs are not well suited to controlling more complex details of demanding applications because of two reasons: Complex applications increase the mental workload of the user and can thus negatively affect BCI performance; and complicated tasks require a number of sub-task, which, when controlled on a low-level bases, can be time consuming and result in fatigue and user frustration. This underscores the importance of reducing the demand on BCI systems and the burden on BCI users through effective goal-oriented protocols. Goal-oriented or high-level BCI control means the BCI simply communicates the user's goal (the task) to the intelligent application device which autonomously performs all necessary sub-tasks to achieve the goal. In contrast, low-level control means the BCI manages all the intricate interactions involved in achieving the task or goal [Wol07].

In any interface, users should not be required to control unnecessary low-level details of system operation. This is especially important with BCIs; allowing low-level control of a wheelchair or robot arm, for example, would not only be slow and frustrating but also dangerous. Therefore, developing robust BCI applications capable of providing effective real world control requires developing intelligent mechanisms that can mediate between the user's goals and the individual actions needed to implement those goals.

Wolpaw [Wol07] compares low-level control to actions of the spinal cord. In a healthy person, the brain is primarily responsible for identifying goals, and the peripheral nervous system bears more responsibility for determining how to best implement these goals. Once a person decides on the goal of drinking water, low-level details of how to move individual muscle groups to attain that goal are identified and implemented without conscious processing. If a BCI system aims to effectively replace the peripheral nervous system, the BCI system including the BCI application must also be able to generate low-level commands as needed to attain specific goals. This idea may also be interpreted through Marr [Mar82], which discusses the three stages necessary to accomplish a task: theory (or goal), algorithm, and implementation. Any BCI can offload the 'implementation' component, but only a goal-oriented BCI in cooperation with intelligent assistive devices can also identify the algorithmic details necessary to attain a goal.

An intelligent BCI system should, however, also provide the user with the option of implementing lower level commands if desired. This is necessary if the high-level commands do not contain the exact goal that a user seeks to attain. An assistive robotic system that automatically pours water if the user conveys thirst may not satisfy a user who instead wants soda, juice, or wine. In these cases, the user might not mind the additional time required to perform tasks that are not preprogrammed. An ideal intelligent system would identify frequently issued command sequences and adapt accordingly, perhaps developing a new option to get juice if the user often does so. On the other hand, many current BCI applications that only provide low-level control would benefit if they would provide high-level control as well. BCI spellers, for instance, could accept high-level commands to prepare letter templates, which are then filled in by the user with low-level commands for spelling the individual words.

A related component of a robust intelligent BCI is application flexibility. Current BCIs support only specifically designed applications. A speller application designed for one BCI cannot be controlled by a different BCI. The large number of assistive devices available with well defined standard interfaces but not specifically designed for BCI has been largely neglected by BCI research so far. Instead of using those already existing devices which has been developed and improved over years, assistive applications for BCI are often “reinvented” and tailored for a particular system. Future BCIs should strive to provide users with increased application flexibility through a straightforward, usable interface that can effectively control a range of already existing assistive devices, software, and appliances. By leveraging existing protocols designed to allow control of common appliances, such as X10, UPNP, or Zigbee, BCI systems could provide greatly increased flexibility without extensive new developments.

In summary, much of the focus in the published literature and popular media is on new BCI applications. While these initial validation efforts are important and promising, it is also crucial to follow up on these efforts by developing improved interfaces through more intelligent, flexible systems. In the following section, two robust intelligent systems developed and investigated at the University of Bremen are introduced and discussed.

1.4.1 Rehabilitation robot

The term rehabilitation robot describes a broad range of assistive devices including powered orthotic or prosthetic devices, exoskeletons, powered feeding devices, automatic guided vehicles such as intelligent wheelchairs, and wheelchairs with mounted manipulators (robot arms) [Hil03]. As assistive devices they are designed to meet the requirements of and address the problems confronted by people with disabilities. The scale of disabilities can range from slightly disabled to most severely disabled people. Depending on user’s impairment (scale of disability) various forms of input mechanisms are used to control rehabilitation robots. Joystick, keyboard, and mouse are typical input devices for less severely disabled people. Sip/puff switches, interpretation of voice or head position, eye gaze, and electromygraphic activity are input modalities for people with disabilities that prevent them from using other interfaces. Control by BCIs is usually the last resort for the most severely disabled people.

The rehabilitation robot FRIEND II (**F**unctional **R**obot Arm with User Fr**ien**dly Interface for **d**isabled People) developed at the Institute of Automation at the University of Bremen is a semi-autonomous system designed to assist disabled people in activities of daily living [IMG05]. The main components of FRIEND II are a conventional wheelchair, a manipulator (robot arm) with 7 degrees-of-freedom, a gripper with force/torque sensor, a stereo camera system mounted on a pan-tilt head, a smart tray with tactile surface and weight sensors, and a computing unit consisting of three independent industrial PCs. Figure 5 shows the FRIEND II system and an able-bodied user who controls the system with a BCI.

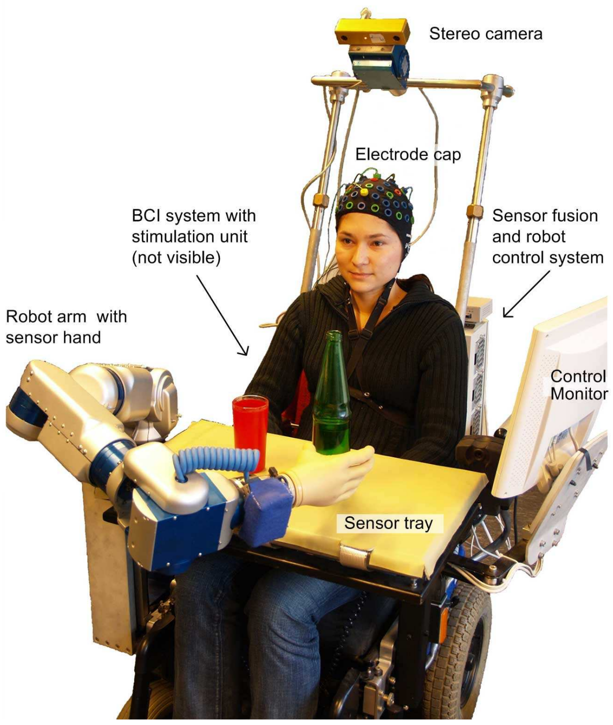

Stereo camera
Electrode cap
BCI system with stimulation unit (not visible)
Sensor fusion and robot control system
Robot arm with sensor hand
Control Monitor
Sensor tray

Figure 5: Rehabilitation robot FRIEND II controlled by the Bremen BCI

The stereo camera system and the smart tray form together a robust and redundant system that is able to reliably localize objects on the tray largely independent from lighting conditions. FRIEND II is able to perform certain operations completely autonomously. An example of such an operation is a “pour in beverage” scenario. In this scenario, the system detects the bottle and the glass both located at arbitrary positions on the tray, it grabs the bottle, moves the bottle to the glass while automatically avoiding any obstacles on the tray, it fills the glass with liquid from the bottle while continuously controlling the fill level of the glass, and finally puts the bottle back in its original position – again avoiding any possible collisions.

Besides this goal-oriented, high-level control approach for autonomously performed tasks, FRIEND II also provides the option to perform low-level control. This is necessary because the system operates in an unstructured environment and uncertainties in sensor data can, in rare cases, result in slightly erroneous estimation of the position of objects. In such cases the user can take over the control of the manipulator and adjust the gripper location. This is done in an intuitive manner using Cartesian commands and still with support from the system, which controls the redundancy of the manipulator in parallel to simplify the task for the user. After gripper adjustment by the user, the system can proceed with the execution of the remaining tasks in an autonomous mode [LOF07].

This flexibility in providing goal-oriented, high-level commands and also low-level, direct control commands is facilitated by the robot control architecture MASSiVE (Multilayer Architecture for Semi-Autonomous Service Robots with Verified Task Executions) [PMC07]. In order to enable user involvement in task execution, the

MASSiVE architecture consists of four modules, shown in Figure 6. The reactive-layer abstracts the hardware specific functionality of sensors and actuators. The sequencer is responsible for task planning, and the human-machine interface (HMI) governs user interactions and provides feedback.

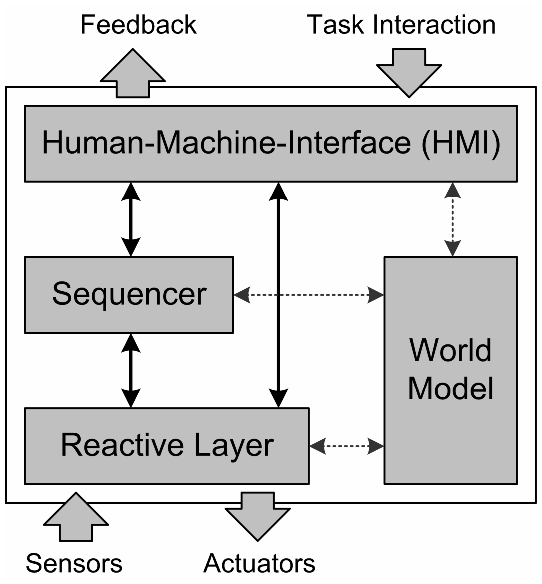


Figure 6: Robot control architecture MASSiVE

A task is selected and started through an input device. At present, keyboard, mouse, joystick, voice and BCI control are supported. The HMI manages the different input devices. A selected high-level task is sent from the HMI to the sequencer, which plans the task according to pre-defined task knowledge and petri-nets. This produces is a list of sub-tasks that the reactive layer will execute. If a problem would occur during the execution of these sub-tasks and the user has to be involved, the sequencer calls a special user interaction sub-task using its interface to the HMI. The situation can then be addressed by the user [PMC07]. From the sequencer's point of view, these user interaction sub-tasks are like normal sub-tasks executed by hardware components. This approach reduces the complexity of the semi-autonomous system. In the context of planning and execution, the world model acts as a knowledge base that stores data for different levels of abstraction. On one hand, the sequencer needs data describing objects during the task planning process (e.g., what kind of objects are involved in the task scenario) without further knowledge (symbolic data). On the other hand, detailed information about those objects is needed during execution of an already planned task in the reactive layer (sub-symbolic data). Hence, the sequencer operates on the symbolic, the reactive layer on the sub-symbolic part of the world model. The connection between both kinds of data is established via object anchoring [Pre05]. The directed connection between the world model and HMI evolves from the need to give feedback to the user during execution of a user interaction sub-task. Thus, the HMI only operates on the sub-symbolic part of the world model.

In order to investigate the feasibility to control the rehabilitation robot by a brain-computer interface, the Bremen SSVEP BCI was connected to the robot control architecture. Communication between the BCI system and MASSiVE was established by a simple TCP/IP connection. Four flickering lights with the frequencies 13Hz, 14Hz, 15Hz, and 16Hz and encoding four commands (select next, select previous, start task, cancel task) were used to select goal-oriented, high-level tasks performed by FRIEND II. Seven subjects in the age of 25 to 35 and without previous BCI experience participated in this study. Each user were asked to perform a predefined sequence of 10 commands to select high-level robot commands. Errors made and the time required to complete the command sequence was measured. The experiment was conducted in an office like environment without any special shielding or other precautions to avoid or reduce external noise or interferences from other electronic devices or the power line. Two of the seven subjects were not able to produce a sufficiently strong SSVEP response, and thus were not able to perform the task. The other five subjects achieved in average a classification rate of 96% and selection speed of 4.61 seconds per command. After the experiment all participants were asked if the task was fatiguing, if the flickering lights caused any inconvenience, and if the selection task required considerable mental load. All participants found the task non-fatiguing, none reported about any inconvenience concerning the flickering lights, and none found that the task required a high mental load. Details about this study can be found in [VCF07].

These results demonstrate that a goal-oriented control of a rehabilitation robot with a BCI is possible. It also demonstrates the robustness of the Bremen BCI system, because the test environment was unlike in most other BCI studies not a shielded room with ideal conditions for EEG recording. The fact that two out of seven subjects were not able to produce SSVEP cannot be accounted to the inefficiency of the BCI system. Rather this is a result of using untrained subjects for this study. Although many subjects can perform selective attention tasks without any training, some subjects need training to learn to perform the task of selective attention effectively.

1.4.2 Smart wheelchairs

In this section another type of rehabilitation robot, a smart wheelchair, is considered as a potential intelligent application for BCI control. A smart wheelchair is an assistive device that provides mobility to individuals who have trouble using a powered wheelchair due to motor or other impairments. Smart wheelchairs employ artificial control systems that augment or replace user control. Smart wheelchairs may employ sensors, such as sonar or infrared sensors, or laser range finders to detect obstacles or map the topography of the environment. Together with sophisticated algorithms including path planning, behavior-based control, and artificial reasoning, this sensor information is used to provide obstacle avoidance, safe object approach, straight path maintenance, and solutions for other navigational issues. Smart wheelchairs can thereby considerably reduce the motor and cognitive requirements for operating a wheelchair [LBJ99].

Wheelchairs may allow different levels of control. At a low-level, the user is responsible for path planning and most of the navigational activities, and the smart wheelchair provides collision-avoidance. This operating level works well with reliable continuous input modes such as a joystick. For discrete input commands such as voice control or BCI input, more intelligent wheelchair behavior is required. In these cases, the system should assist the user with path planning decisions and provide semi-autonomous control. For an even higher level of control – real goal-oriented control – where the user only supplies the target destination, the artificial control system has to plan the path completely and navigate to the destination autonomously. More details about smart wheelchairs can be found in the review article [Sim05].

The Bremen semi-autonomous wheelchair Rolland III has been developed by the Institute of Computer Science at the University of Bremen. It features a conventional motorized wheelchair equipped with two laser range finders. The wheelchair can deal with a number of input modalities such as low-level joystick control or high-level discrete control. Autonomous and semi-autonomous navigation is supported. This flexibility is achieved by continuous local mapping of the environment in real-time and sophisticated path planning and obstacle avoidance algorithms [MF07]. Figure 1 shows the smart wheelchair controlled by the Bremen BCI.

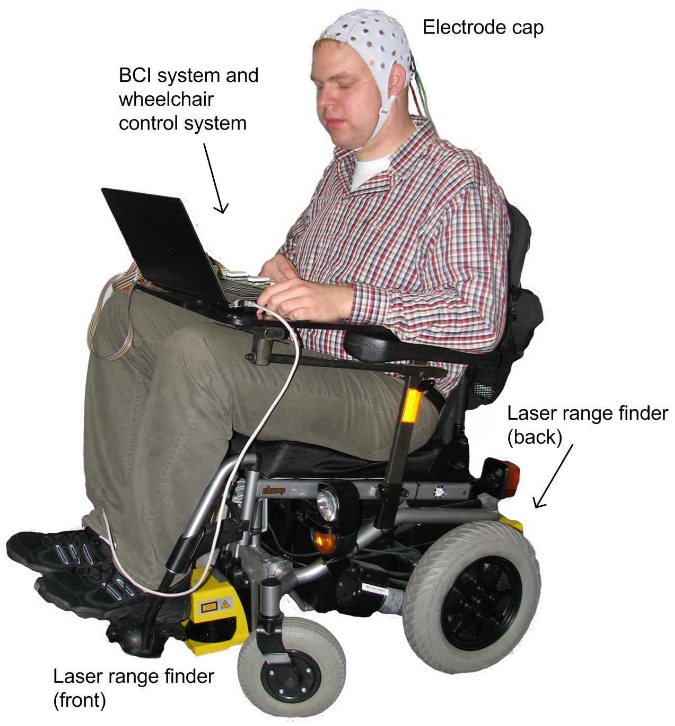

Electrode cap
BCI system and wheelchair control system
Laser range finder (back)
Laser range finder (front)

Figure 7: Bremen Autonomous Wheelchair Rolland III controlled by the Bremen BCI

For BCI control, Rolland currently operates in a semi-autonomous mode and accepts the following set of commands: go straight ahead, go left, go right, turn around. It projects these specific instructions onto a route graph representation of the surrounding environment [KFL05]. The route graph is a multi-layered and graph-structured representation of the environment in which each graph layer describes the workspace on a different level of abstraction. To interpret the BCI instructions, the Voronoi layer of the graph representation, which includes metrical and topological information of the free and occupied space of the surrounding, is evaluated. This comprehensive spatial representation is first searched for all navigable routes that might move the wheelchair to the periphery of the visible neighborhood. Next, the set of navigable routes is evaluated against the given directional command by searching the route that best fits the actual instruction. This process employs fuzzy spatial relations that were first introduced to interpret coarse verbal route descriptions [MFR06]. The process assesses the branching angle between two consecutive segments of a given route. The best route is finally passed to the obstacle avoiding navigation module.

Initial pilot research has been conducted to assess the Bremen SSVEP BCI with Rolland. This pilot work sought to show that a BCI based on selective attention can effectively control an intelligent device even if the user is distracted by other tasks like observing the surrounding environment, interacting with other people, and planning next navigational steps. This is not clear a priori. Since SSVEP BCIs require users to focus attention on specific stimuli, any task that distracts the user could reduce BCI performance. Furthermore, navigation is a difficult task, even with assistance via the intelligent behavior of the wheelchair. The time constraints in a navigation scenario, such as the need to convey the “turn left” command before the next possible left turn, could further increase the user’s stress and workload.

Only one subject has been assessed to date. He successfully drove Rolland through the hallway area of a typical office building with a speed of 0.56 m/sec. No effort was made to reduce external noise sources such as electronic devices, 50 Hz current, or fluorescent lighting. Control relied on selective attention to one of four LEDs representing the wheelchair control commands described above. The subject sent about 40 navigation commands with classification accuracy above 95%. The subject sent commands over a period of about 30 minutes, but frequently paused to talk to colleagues, and thus effective information throughput is difficult to measure. Hence, this pilot work shows that an SSVEP BCI can robustly control a wheelchair in a real world setting provided the wheelchair is intelligent enough to provide obstacle avoidance and assist in path planning. However, toward the end of the session, the subject reported fatigue, and accurate classification required longer periods of sustained attention. One likely cause of this fatigue is that the stimulation unit were located on the wheelchair tray (above the subject’s lab), and thus the subject had to look down to see it. The subject also had to frequently shift gaze between the LED apparatus and the surrounding environment. This required considerable head and eye movement, as well as frequently switching attention, that could make a system both fatiguing and difficult or impossible to use for persons with motor disabilities. Thus, improved display parameters might eliminate the need for physical movements and significantly reduce fatigue. Several such improvements are currently explored, along with and their effective integration with other display components such as feedback from the BCI,

This is only an initial report, and many other avenues for improvement are being investigated. Although the subject could control the wheelchair, the initial results demonstrated that the ongoing distraction from different sources affect performance. Allison et al. [AP06] reported that background activity did not significantly impair SSVEP BCI performance, but that answering complex questions did require disengaging from the BCI. It is very likely that training could improve a BCI user's ability to effectively multitask, just as experienced keyboard users or drivers can handle moderate amounts of distraction better than novices.

Currently, the Bremen BCI communicates only uni-directionally with both FRIEND II and Rolland. That is, goal-oriented, high-level commands are sent to the intelligent assistive device, but no direct information from this device is sent back to and used by the BCI. However, the overall system (BCI and intelligent application) could greatly benefit from bi-directional communication. Since direct brain-computer communication is unreliable, BCI decisions are always associated with uncertainty. In the Bremen SSVEP BCI used for controlling Rolland, for instance, four commands are possible, each corresponding to a different LED. When a command is generated, the most likely command out of the four is selected. This likelihood is determined from the signal-to-noise ratio (SNR) of the SSVEP components that correspond to LED frequencies. Sometimes, it is difficult to determine which LED the user is focused on, since the SNR of two or more SSVEP components may be similar. Without further information, the BCI may select the wrong command. However, if additional information provided by the assistive device is available, e.g. that a certain command is not plausible in the current context, a more appropriate decision could be made. For example, the BCI system would not select a command to move forward if it knew (through sensors) that the wheelchair was facing a wall and could not move forward. This would lead to more robust decisions and improved information transfer rates. This idea is conceptually similar to incorporating context to determine which navigation command is sent to a mobile robot [MEM04].

1.5 Conclusion and future prospects

It is unlikely that BCI with dramatically better information throughput will be developed in the near future. Current tools for measuring brain activity are simply not powerful enough. The EEG, for example, allows a fairly low bandwidth signal that only reflects a tiny percentage of the brain's activity. Signals from different brain areas and mental processes often overlap, are badly smeared by the scalp, and are difficult to distinguish from background brain activity that is not relevant to BCI control. EEG activity is stochastic, nonstationary, and often inconsistent across subjects and within and across usage sessions. Electrical noise from other physiological signals, movements, outside devices further degrades signal quality. Improved robust signal processing approaches can improve throughput [FVG07; BKD06; PGN06], but significant new neuroimaging technology is also needed.

However, providing the user with the desired application(s) and choices can make a huge difference in the effective information throughput. A patient may prefer a slow BCI that allows changing bed position over a faster BCI that only allows spelling. Similarly, intelligent systems that manage unnecessary details of system operation can substantially improve effective information throughput. For example, a wheelchair and robotic arm system that requires BCI users to control each movement would be slow, frustrating, and possibly even dangerous. Such a system would only be practical if the user could instead select from common goals, such as opening a door, getting water, or making coffee [VCF07, Wol07]. Effective interfaces, error detection and correction, word and sentence completion, display parameters that do not produce fatigue or eyestrain, and other mechanisms can further increase the robustness of BCI systems and effectively improve information throughput.

BCI hardware and software are quickly becoming more usable, and the need for expert assistance – or indeed any assistance – may be eliminated. New work has described dry EEG electrodes that do not require gel, precise positioning, nor skin abrasion [FCM07]. Dry electrode systems might require no expert assistance for setup or cleanup, and could be easily integrated into a baseball cap or headband. High impedance amplifiers and electrodes, combined with wireless recordings, could make BCIs even more robust in noisy environments.

The need for expert help to configure each BCI might be addressed through a intelligent software that runs each new user through a series of tests. These tests would provide information that an automated expert system could use to configure the best BCI for each user. Ideally, this expert system could utilize a variety of pattern classification tools, stimulus and/or feedback parameters, and user queries to identify the best BCI for each user. The system would have to be robust to changes in each user's EEG over time, requiring self – adaptive mechanisms [DMH07]. Since performance may differ widely across BCI approaches [BC07], this software should also examine P300, SSVEP, and ERD approaches. These tools would require substantial development work, but (in concert with dry electrodes) might eliminate one of the biggest obstacles to BCI use: dependence on other persons, specifically experts. Thus, new intelligent algorithms capable of replacing human experts could make BCI systems much more robust and practical to individual users. Systems such as the Bremen SSVEP BCI, which can adapt itself according to different facets of each users' EEG activity, are an important step in this direction.

In summary, some BCI limitations are quite trenchant, such as the fundamental limitations of modern brain imaging technology. On the other hand, although modern BCIs are difficult to use, not very robust, and require outside expert help, these drawbacks may become substantially less potent over the next several years. As a result, BCIs will become more useful to typical users – people with severe motor disabilities – and may become useful to other users in some situations. Robust BCIs in combination with intelligent BCI applications that effectively integrate goal-oriented protocols and are able to largely compensate the limitations of direct brain-computer communication will play a crucial role in this next generation of BCI systems.

Advances in BCI technology will make BCIs more appealing to new user groups. BCI systems may provide communication and control to users with less severe disabilities, and even healthy users in some situations. BCIs may also provide new means of treating stroke, autism, and other disorders (BC07, GAG07). These new BCI applications and groups will require new intelligent BCI components to address different challenges, such as making sure that users receive the appropriate visual, proprioceptive, and other feedback to best recover motor function.

As BCIs become more popular with different user groups, increasing commercial possibilities will likely encourage new applied research efforts that will make BCIs even more practical. Consumer demand for reduced cost, increased performance, and greater flexibility and robustness may contribute substantially to making BCIs into more mainstream tools.

References

[AP03] Allison, B.Z., Pineda, J.A.: ERPs evoked by different matrix sizes: implications for a brain computer interface (BCI) system. IEEE Trans Neural Syst Rehabil Eng, 11, 110-113 (2003)

[AP06] Allison, B.Z., Pineda, J.A.: Effects of SOA and flash pattern manipulations on ERPs, performance, and preference: Implications for a BCI system. International Journal of Psychophysiology, 59, 127-140 (2006)

[AWW07] Allison, B.Z., Wolpaw, E.W., Wolpaw, A.R.: Brain-computer interface systems: progress and prospects. Expert Review of Medical Devices, 4, 463-474 (2007)

[AGG07] Allison, B.Z., Graimann, B., and Gräser, A: Why use a BCI if you are healthy? In International Conference on Advances in Computer Entertainment Salzburg, 7-11 (2007)

[BC07] Birbaumer, N., Cohen, L.G.: Brain-computer interfaces: communication and restoration of movement in paralysis. Journal of Physiology-London, 579, 621-636 (2007)

[BDK07] Blankertz, B., Dornhege, G., Krauledat, M., Muller, K.R., Curio, G.: The non-invasive Berlin Brain-Computer Interface: Fast acquisition of effective performance in untrained subjects. Neuroimage, 37, 539-550 (2007)

[DSW00] Donchin, E., Spencer, K.M., Wijesinghe, R.: The mental prosthesis: assessing the speed of a P300-based brain-computer interface. IEEE Trans Rehabil Eng, 8, 174-179 (2000)

[DBC04] Dornhege, G., Blankertz, B., Curio, G., Muller, K.R.: Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms. IEEE transactions on bio-medical engineering, 51, 993-1002 (2004)

[DMH07] Dornhege, G., Millan, J.R., Hinterberger, T., McFarland, D.J., Müller, K.R., eds. Toward Brain-Computer Interfacing. MIT Press (2007).

[FCM07] Fonseca, C., Cunha, J.P.S., Martins, R.E.: A novel dry active electrode for EEG recording. IEEE Trans. Biomed. Eng., 54, 162-165 (2007)

[FVG07] Friman, O., Volosyak, I., Graeser, A.: Multiple channel detection of steady-state visual evoked potentials for brain-computer interfaces. IEEE transactions on bio-medical engineering, 54, 742-750 (2007)

[FLV07] Friman, O., Lueth, T., Volosyak, I., Graeser, A.: Spelling with Steady-State Visual Evoked Potentials. In The 3rd International IEEE EMBS Conference on Neural Engineeering Hawai (2007)

[Fuk90] Fukunaga, K.: Introduction to statistical pattern recognition. 2nd edn: Academic Press, Boston (1990)

[GXC03] Gao, X., Xu, D., Cheng, M., Gao, S.: A BCI-based environmental controller for the motion-disabled. IEEE Trans Neural Syst Rehabil Eng, 11, 137-140 (2003)

[GHL04] Graimann, B., Huggins, J.E., Levine, S.P., Pfurtscheller, G.: Toward a direct brain interface based on human subdural recordings and wavelet-packet analysis. IEEE transactions on bio-medical engineering, 51, 954-962 (2004)

[GP06] Graimann, B., Pfurtscheller, G.: Quantification and visualization of event-related changes in oscillatory brain activity in the time-frequency domain. In: Event-related Dynamics of Brain Oscillations., C. Neuper, W. Klimesch, eds. Elsevier, 79-97 (2006)

[Gra06] Graimann, B.: Event-related (de)synchronization in bioelectrical brain signals and its use in Brain-Computer Communication. Habilitationsschrift: Graz University of Technology (2006)

[GAG07] Graimann, B., Allison, B.Z., Gräser, A.: New applications for non-invasive brain- computer interfaces and the need for engaging training environments. In International Conference on Advances in Computer Entertainment Salzburg, 25-28 (2007)

[GEH03] Guger, C., Edlinger, G., Harkam, W., Niedermayer, I., Pfurtscheller, G.: How many people are able to operate an EEG-based brain-computer interface (BCI)? IEEE Trans Neural Syst Rehabil Eng, 11, 145-147 (2003)

[HLT07] Hill, J., Lal, T.N., Tangermann, M., Hinterberger, T., Widman, G., Elger, C.E.: Classifying Event-Related Desynchronization in EEG, ECoG, and MEG Signals. In: Toward Brain-Computer Interfacing, G. Dornhege, J.R. Millan, T. Hinterberger, D.J. McFarland, K.R. Müller, eds. MIT Press, 235-259 (2007)

[Hil03] Hillman, M.: Rehabilitation robotics from past to present - a historical perspective. In The 8th International Conference on Rehabilitation Robotics, ICORR 2003, 101-105 (2003)

[IMG05] Ivlev, O., Martens, C., Graeser, A.: Rehabilitation robots FRIEND-I and FRIEND-II with the dexterous lightweight manipulator. In Prcoceedings of the 3rd International Congress on Restoration of (wheeled) mobility in SCI rehabilitation, 111-123 (2005)

[KB05] Kleber, B., Birbaumer, N.: Direct brain communication: Neuroelectric and metabolic approaches at Tuebingen. Cogn Process, 6, 65-74 (2005)

[KFL05] Krieg-Brueckner, B., Frese, U., Luettich, K., Mandel, C., Mossakowski, T., Ross, R.: Specification of an Ontology for Route Graphs. In: Spatial Cognition IV, C. Freska, M. Knauff, B. Krieg-Brueckner, B. Nebel, T. Barkowsky, eds. Springer, 390-412 (2005)

[LBC05] Lemm, S., Blankertz, B., Curio, G., Muller, K.R.: Spatio-spectral filters for improving the classification of single trial EEG. IEEE transactions on bio-medical engineering, 52, 1541-1548 (2005)

[LBJ99] Levine, S.P., Bell, D.A., Jaros, L.A., Simpson, R.C., Koren, Y., Borenstein, J.: The NavChair Assistive Wheelchair Navigation System. IEEE Trans Rehabil Eng, 7, 443-451 (1999)

[LOF07] Lueth, T., Ojdanic, D., Friman, O., Prenzel, O., Graeser, A.: Low-level control in a semi-autonmous rehabilitation robotic system via a Brain-Computer Interface. In The 10th International Conference on Rehabilitation Robotics, ICORR 2007, 721-728 (2007)

[MF07] Mandel, C., Frese, U.: Comparison of Wheelchair User Interfaces for the Paralysed: Head-Joystick vs. Verbal Path Selection from an offered Route-Set. In Proceedings of the 3rd European Conference on Mobile Robots, ECMR 2007, 217–222 (2007)

[MFR06] Mandel, C., Frese, U., Roefer, T.: Robot Navigation based on the Mapping of Coarse Qualitative Route Descriptions to Route Graphs. In In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006, 205-210 (2006)

[Mar82] Marr, D.: Vision : a computational investigation into the human representation and processing of visual information. W.H. Freeman, San Francisco (1982)

[MMD97] McFarland, D.J., McCane, L.M., David, S.V., Wolpaw, J.R.: Spatial filter selection for EEG-based communication. Electroencephalogr Clin Neurophysiol, 103, 386-394 (1997)

[MSW03] McFarland, D.J., Sarnacki, W.A., Wolpaw, J.R.: Brain-computer interface (BCI) operation: optimizing information transfer rates. Biological psychology, 63, 237-251 (2003)

[MMC00] Middendorf, M., McMillan, G., Calhoun, G., Jones, K.S.: Brain-computer interfaces based on the steady-state visual-evoked response. IEEE Trans Rehabil Eng, 8, 211-214 (2000)

[MEM04] Millan, J.R., Renkens, F., Mourino, J., Gerstner, W.: Noninvasive brain-actuated control of a mobile robot by human EEG. IEEE transactions on bio-medical engineering, 51, 1026-1033 (2004)

[MAB03] Müller, K.R., Anderson, C.W., Birch, G.E.: Linear and nonlinear methods for brain-computer interfaces. IEEE Trans Neural Syst Rehabil Eng, 11, 165-169 (2003)

[MKD04] Müller, K.-R., Krauledat, M., Dornhege, G., Curio, G., Blankertz, B.: Machine Learning Techniques for Brain-Computer Interfaces. Biomed Tech (Berl), 49, 11-22 (2004)

[MSP05] Müller-Putz, G.R., Scherer, R., Pfurtscheller, G., Rupp, R.: EEG-based neuroprosthesis control: a step towards clinical practice. Neurosci Lett, 382, 169-174 (2005)

[MBL06] Naeem, M., Brunner, C., Leeb, R., Graimann, B., Pfurtscheller, G.: Seperability of four-class motor imagery data using independent components analysis. Journal of neural engineering, 3, 208-216 (2006)

[PN01] Pfurtscheller, G., Neuper, C.: Motor imagery and direct brain-computer communication. Proceedings of the IEEE, 89, 1123 (2001)

[PMS06] Pfurtscheller G, Müller-Putz GR, Schlogl A et al. 15 years of BCI research at Graz University of Technology: Current projects. IEEE Trans. Neural Syst. Rehabil. Eng. 14, 205–210 (2006)

[PGN06] Pfurtscheller, G., Graimann, B., Neuper, C.: EEG-based Brain-Computer Interface Systems and Signal Processing. In: Encyclopedia of Biomedical Engineering, M. Akay, ed. John Wiley & Sons, New Jersey, 1156-1166. (2006)

[Pre05] Prenzel, O.: Semi-autonomous object anchoring for service-robots. Methods and Applications in Automation, 1, 57-68 (2005)

[PMC07] Prenzel, O., Martens, C., Cyriacks, M., Wang, C., Graeser, A.: System-controlled user interaction within the service robotic control architecture MASSiVE. Robotica, 25, 237-244 (2007)

[RMP00] Ramoser, H., Muller-Gerking, J., Pfurtscheller, G.: Optimal spatial filtering of single trial EEG during imagined hand movement. IEEE Trans Rehabil Eng, 8, 441-446 (2000)

[SKM06] Sellers, E.W., Krusienski, D.J., McFarland, D.J., Vaughan, T.M., Wolpaw, J.R.: A P300 event-related potential brain-computer interface (BCI): the effects of matrix size and inter stimulus interval on performance. Biological psychology, 73, 242-252 (2006)

[SD06] Sellers, E.W., Donchin, E.: A P300-based brain-computer interface: Initial tests by ALS patients. Clin Neurophysiol, 117, 538-548 (2006)

[Sim05] Simpson, R.C.: Smart wheelchairs: A literature review. Journal of Rehabilitation Research & Development, 42, 423-436 (2005)

[VCF07] Valbuena, D., Cyriacks, M., Friman, O., Volosyak, I., Graeser, A.: Brain-Computer Interface for high-level control of rehabilitation robotic systems. In The 10th International Conference on Rehabilitation Robotics, ICORR 2007, 619-625 (2007)

[Vid73] Vidal, J.J.: Toward direct brain-computer communication. Annu Rev Biophys Bioeng, 2, 157-180 (1973)

[WMN91] Wolpaw, J.R., McFarland, D.J., Neat, G.W., Forneris, C.A.: An EEG-based brain-computer interface for cursor control. Electroencephalogr Clin Neurophysiol, 78, 252-259 (1991)

[WBM02] Wolpaw, J.R., Birbaumer, N., McFarland, D.J., Pfurtscheller, G., Vaughan, T.M.: Brain-computer interfaces for communication and control. Clin Neurophysiol, 113, 767-791 (2002)

[Wol07] Wolpaw, J.R.: Brain-computer interfaces as new brain output pathways. The Journal of physiology, 579, 613-619 (2007)