Methodology for Robot Mapping and Navigation in Assisted Living Environments

Syed Atif Mehdi
Robotics Research Lab
University of Kaiserslautern
67653 Kaiserslautern,
Germany
mehdi@informatik.uni-kl.de

Christopher Armbrust
Robotics Research Lab
University of Kaiserslautern
67653 Kaiserslautern,
Germany
armbrust@informatik.uni-kl.de

Jan Koch
Robotics Research Lab
University of Kaiserslautern
67653 Kaiserslautern,
Germany
koch@informatik.uni-kl.de

Karsten Berns
Robotics Research Lab
University of Kaiserslautern
67653 Kaiserslautern,
Germany
berns@informatik.uni-kl.de

ABSTRACT

Robust navigation in living environments demands high requirements on the control system of a robot. Due to typically narrow passages between obstacles, precise navigation is required. To achieve accuracy in navigation, detailed representation of the environment around the robot needs to be developed. Robust and precise mapping of the environment helps in overcoming the dynamics in the living environment like movement of furniture and human beings. Prompt recovery from unreachable paths while navigating is also an essential component of the living environment robots. This paper describes a behaviour-based navigation system in assisted living environments. The navigation system uses a grid map created from data obtained from laser scanner and ultrasonic sensors mounted on a small sized robot, ARTOS. ARTOS is specially designed for indoor living environments able to navigate through narrow corridors and closely placed furniture in the living environment.

Categories and Subject Descriptors

I.2.9 [**Articial Intelligence**]: Robotics—*Autonomous vehicles, Commercial robots and applications, Kinematics and dynamics*

General Terms

Assisted Living Robot, Mapping and Navigation

Keywords

Assisted Living Robot, Mapping, Navigation

1. INTRODUCTION

Current demographic development in most industrialised countries leads to a higher number of elderly people with mental or physical disabilities. These people are often not able to perform all activities of daily life without help and face a higher risk of experiencing medical emergency. Therefore, they usually have to move to assisted living facilities where they are looked after by the nursing staff. As the costs of such facilities are very high, the use of modern technology in assisted living environments is being researched by numerous groups. The aim is to lower the nursing costs and increase the quality of life of persons with disabilities [3], [7], [9], [2].

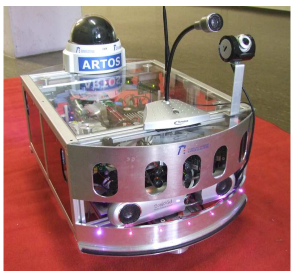


Figure 1: ARTOS (Autonomous Robot for Transport and Service).

BelAmI project[1] is a research initiative that deals with the use of ambient intelligence in the field of assisted living. Fig. 2 shows an overview of an assisted living apartment which is equipped with sensor and camera systems to locate and monitor an elderly person. In addition to these, passive RFID tags have been integrated in the carpet to monitor the movement of a person wearing shoes capable of reading RFID tags [6]. Despite using different systems for localising and monitoring an elderly person, a small autonomous robot ARTOS (Fig. 1), capable of moving through narrow corridors, is

[1]http://www.belami-project.org/

also developed to assist the person in various ways. It serves as transport vessel, supports the communication between the user and other people, facilitates the communication with different ambient intelligence systems and patrol the environment to detect medical emergencies. A detailed description of the mechatronics system of ARTOS can be found in [5].

For these applications, the mapping and navigation system requires that a variety of sensors must be employed as sources for the map creation, reason being the living environments typically contain many obstacles that are invisible to some types of sensors. In addition, precise navigation is mandatory as the robot has to drive through narrow passages like doorways and closely placed furniture. Also the environment can contain different dynamic obstacles, therefore, an early detection of path obstructions followed by a fast reaction is necessary.

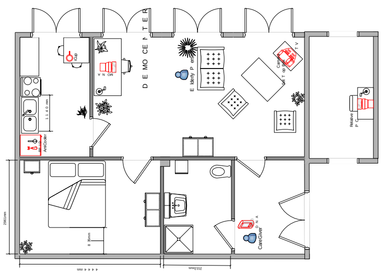


Figure 2: The BelAmI testing apartment.

2. THE MAPPING SYSTEM

Mapping and navigation of robots has always been a challenge to the researchers. Various techniques have been devised for precise mapping and navigation, a more recent methodology, Simultaneous Localisation and Mapping (SLAM) has been developed to localise the robot while building a map of the environment [8]. Although promising results are being achieved but according to [10], different implementations of SLAM pose certain limitations. SLAM works better in corridors and areas which are less populated and where there are predictable features in the environment. The dynamic nature of living environments makes it harder to track features in the environment, e.g. moving a chair may hide some features in the environment, and hence reliable mapping may not be possible. As a result of limited accuracy in mapping and localisation, navigation based on SLAM is prone to failures in natural living environments and hence may not be reliable in elderly care environments.

Considering the significance of mapping methodology and taking advantage of RFID-tags integrated in the carpets of the assisted living environment, mapping in ARTOS is done using ultrasonic sensors and laser scanner storing information in grid maps and localising the robot using RFID-tags.

The structure of the control system is depicted in Figure 3. The hardware abstraction contains the basic sensor processing and actuator control. It delivers data to and receives commands from other components of the system, including the mapping and navigation systems. The anti-collision system has been implemented using the behaviour-based control architecture iB2C [11] as described in [1].

2.1 Map Creation

The main target of the mapping system is to support precise navigation and hence a grid map approach is chosen for map building. A grid map represents the world around the robot as an array of

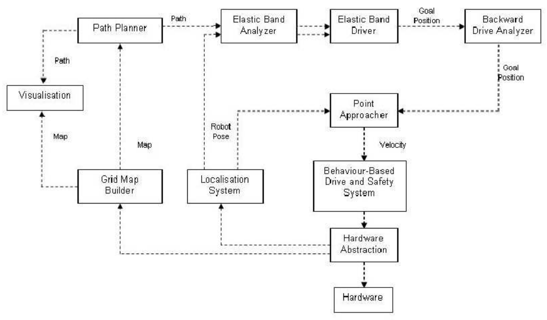


Figure 3: The elements of the control system

(usually uniform) grid cells. Each cell stores information about the area it covers, the most important information is whether the area is occupied or not. The current occupancy belief is represented by an occupancy counter, positive values are used for occupied cells, negative values for free cells and an occupancy counter of zero reflects an unknown occupancy state. The occupancy counters are limited so that a belief in an occupancy state cannot get too strong.

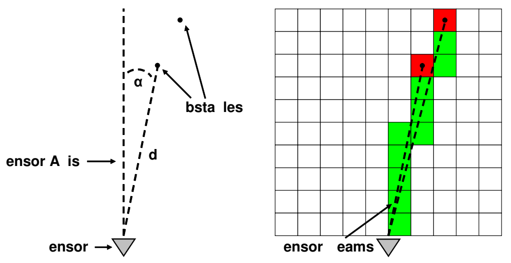
A laser range finder and two chains of ultrasonic sensors are used as source for the grid map creation process. The design of *Grid Map Builder*, the component that contains the mapping functionality, allows addition of an arbitrary number of sensor systems which can be used simultaneously. The sensor data can be obtained in one of the two formats:

1. Polar format: The sensor values are stored as a series of distance-angle-pairs. This format is used for the data from laser range finder.
2. Sector map format: A sector map consists of number of sectors, each of them containing information about the relevant (closest) obstacle in the area it covers. Here, the sectors are circular, so the position of an obstacle is described with polar coordinates. This format is used for the data from ultrasonic sensors.

Depending on the format, the grid map creation is carried out in two different ways.

- When using the polar format, sensor beam tracing is used to alter the occupancy counters within the field of view of the sensor, each sensor beam is traced in small steps. Whenever it hits a grid cell, the corresponding occupancy counter is decremented by 1. The counter of the last cell, i.e. the cell in which the obstacle lies, is incremented. The principle is illustrated by Fig. 4.
- When using polar sector maps, simply tracing sensor beams leads to different problems. The fact that the polar coordinates provide information about the *closest* obstacle in a sector is not used. Given a sector containing an obstacle, all grid cells in this sector which are closer to the sensor can be regarded as free. But with sensor beam tracing, only the occupancy counters of cells which lie on a line between the sensor and the obstacle would be decremented.

One solution is to process the sector map data in two steps as shown in Algorithm 2.1. In the first step, the cells in which obstacles lie are marked, each cell contains at most one obstacle. All sectors have to be traversed, the polar obstacle data has to be transformed into map coordinates, and finally the obstacle containing



(a) Two obstacles in the environment.

(b) Two sensor beams have been traced.

Figure 4: The creation of a grid map by tracing sensor beams (green/light grey: free; red/dark grey: occupied; white: unknown).

cells have to be marked. In the second step, all cells in a rectangular area around the sensor are traversed. For each cell, the coordinates of its centre are transformed into polar coordinates of the sensor coordinate system. The information is then processed to determine whether the cell is closer to the sensor than the obstacle of this sector. If this is the case, it is regarded as free and its occupancy counter is decremented. Fig. 5 illustrates the two steps.

Algorithm 1 Polar Sector Map

```
Step 1:
for all sectors $s$ with obstacle $o$ do
    calculate cell $c$ containing $o$
    increment occupancy counter of $c$
end for

Step 2:
for all cells $c$ in rectangular area around sensor do
    calculate sector $s$ containing $c$
    if (distance($c$) < distance($o$)) then
        decrement occupancy counter of $c$
    end if
end for
```

2.2 Integrating Maps

A grid map is created from data of *one* sensor using the above mentioned procedure. Integration of additional sensors require that the procedure must be repeated for every sensor. This also requires the use of different arrays of occupancy counters for different sensors, otherwise an obstacle that is *only* visible to sensor S_1 would be removed when processing the data of sensor S_2, which does not see it. Therefore, in every execution cycle, the data of each sensor is processed and the corresponding array of occupancy counters is updated. Then the data of these arrays is aggregated to build one combined grid map according to the following rules:

1. If a cell is occupied in at least one sensors' array, then its counter in the combined grid map is set to $+1$.

2. If a cell is not occupied in any of the sensors' arrays and is free in at least one sensor's array, then its counter in the combined grid map is set to -1.

3. If a cell is not occupied or free in any of the sensors' arrays, then its counter in the combined grid map is set to 0.

A grid cell stores additional information than an occupancy counter and the information whether the counter may be changed or not. The most relevant information fields contained in one grid cell are shown in Table 1. The grid map is provided to the path planning component by using the blackboard mechanism.

2.3 Localisation

Localisation in ARTOS is achieved in two ways which results in a more accurate knowledge of the position of the robot. Primarily incremental localisation is done based on odometery information which is further refined by reading landmarks in the form of passive RFID-tags integrated in the carpet by an RFID-reader attached to the bottom of the robot. These landmarks are absolute and helps in determining the orientation of the robot [6]. The level of localisation achieved in this way is sufficient to guarantee precise navigation in living environments.

3. THE NAVIGATION SYSTEM

3.1 Creating Paths

When a new path has to be created, the *Path Planner* first reads the current grid map from the blackboard. It then executes the A*-algorithm [4] to calculate a shortest path from start point s to end point d. This algorithm processes cell by cell, starting with s, until a path to d has been found. In each processing step, the cell with the lowest cost is chosen as next cell to be processed. For a cell c, the cost $f(c)$ is

$$f(c) = g(c) + h(c), \tag{1}$$

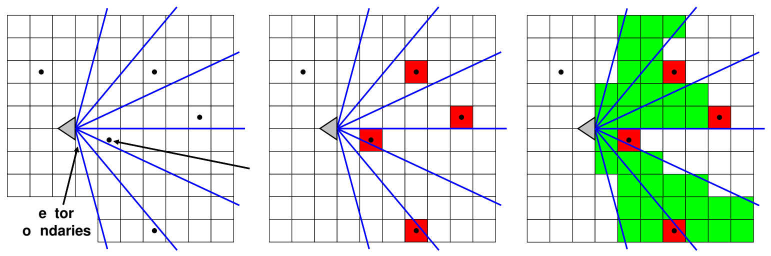
where $g(c)$ denotes the cost for the shortest (known) path going from s to c and $h(c)$ denotes the estimated cost of a path going from c to d. Euclidean distance is used as heuristic function. Before planning a path, the obstacles in the map are enlarged by marking the cells close to them as "neighbours". High costs are assigned to these neighbours so that other free cells may be preferred. As a result, paths do not lead the robot close to obstacles unless it is necessary.

When a path has been created as a sequence of grid cells, the *Path Planner* calculates the "relevance" of points. A point is either "intermediate" or "relevant". In contrast to "intermediate" points, the direction of the path changes at "relevant" points. For optimized path and smooth movement of the robot, it is necessary to tune some of the path points. This can be achieved by considering the fact that the direction of the path does not change at "intermediate" points and can be altered. In contrast, relevant path points changes the direction of robot's movement and hence cannot be modified. To accomodate the dynamically changing environment, path is replanned after certain interval of time.

Finally, a path is represented as a series of triples in the form (x-coordinate, y-coordinate, relevance). The aforementioned blackboard mechanism is used to provide a path to other components of the navigation system and to the visualisation module.

3.2 Driving Forward along Paths

The functionality of driving forward along paths is realised by *Elastic Band Analyzer*, *Elastic Band Driver* and *Point Approacher* modules. The path planned by the *Path Planner* is the shortest path from source to destination. Following exactly those points might result in getting the robot too close to some obstacles which will cause reduction in speed and thus robot might take longer time to reach destination. To overcome this scenario, *Elastic Band* [12] has been used. The *Elastic Band Analyzer* reads the path from



(a) Initial state. (b) After step 1. (c) After step 2.

Figure 5: The creation of a grid map using a polar sector map as data source.

Table 1: The main information fields contained in one grid cell.

Field	Stored Information
occupancy_counter	belief about the occupancy
is_near_obstacle	whether cell is close to an occupied cell or not
is_fixed	whether occupancy belief is fixed or not
g	distance to the start point of a path
h	estimated length of a path going through this cell
update_time	time of the last update of the counter

the blackboard and optimizes the path for smooth movement of robot. It also tries to keep a safe distance between the robot and the obstacles, maintaining the fact that maximum velocity can be achieved.

The *Elastic Band Driver* reads the path from the blackboard and sends the relevant points of the path one by one to the *Point Approacher*. So it encapsulates the blackboard access and offers additional functionality, such as inverting the order of the path points or jumping back or forth to other points.

The *Point Approacher* gets target coordinates and the current robot pose as input and calculates a desired velocity v_{des} and angular velocity ω_{des} depending on the distance d and absolute angle $|\alpha|$ to the target (see Eqs. 2 and 3):

$$v_{\mathrm{des}} = \begin{cases} 0 & ; d \leq d_{\min} \\ 1 & ; d \geq d_{\max} \\ \frac{1}{2} + \frac{1}{2} \cdot \sin\left(\left(\frac{d - d_{\min}}{d_{\max} - d_{\min}} - \frac{1}{2}\right) \cdot \pi\right) & ; \text{else} \end{cases} \tag{2}$$

$$|\omega_{\mathrm{des}}| = \begin{cases} 0 & ; |\alpha| \leq \alpha_{\min} \\ 1 & ; |\alpha| \geq \alpha_{\max} \\ \frac{1}{2} + \frac{1}{2} \cdot \sin\left(\left(\frac{|\alpha| - \alpha_{\min}}{\alpha_{\max} - \alpha_{\min}} - \frac{1}{2}\right) \cdot \pi\right) & ; \text{else} \end{cases} \tag{3}$$

By comparing the robot's orientation to α, ω_{des} can be calculated from $|\omega_{\mathrm{des}}|$. $d_{\min}$, $d_{\max}$, $\alpha_{\min}$, $\alpha_{\max}$ mark the distances and angles at which v_{des} and ω_{des}, respectively, take their extreme values. If $d_{\max}$ is reduced, for example, the robot will drive longer at the maximum speed when approaching a target. If it should decelerate earlier, $d_{\max}$ has to be increased. To smoothen the changes of v_{des} and ω_{des}, sigmoid functions are used.

3.3 Driving Backward along Paths

Living environments are very dynamic in nature. Despite every effort of generating a map of all the fixed and dynamic obstacles, there are many dynamic obstacles that obstruct the shortest path to destination and sometimes makes the destination unreachable. Therefore a module is required to move robot back after acknowledging that it cannot go any further and there is not even enough space to turn around. In this case the only choice left is to come back to a previously visited position where it was safe for robot to move and re-plan the path to the destination. The map has now new information about the dynamic obstacles that prevented the robot to follow original planned path and new path is based on abstaining from these obstacles.

Drive Backward has been realized by considering the velocity of robot after certain time period. If the velocity at a point is above certain threshold, that means it is a comfortable zone for the robot to move around. The current point is recorded as *safe point* in case robot has to come back. There can be two cases in which the velocity can be below threshold. Firstly, it might be that the robot is moving through a tight corridor and there are lots of obstacles around the robot. Secondly, the robot has got stuck at some place and can neither go forward nor turn around. In the first case, after crossing the corridor the robot can regain the high velocity. Therefore, certain time is given as grace period to the robot to regain velocities or change the path by turning around before driving back. In second case, there is no option other than to drive backward to the safe point.

There are certain scenarios (see Fig. 6) in which driving backward results in failure to reach the final destination. An example of a simple situation could be that while driving back, an obstacle may intercept the path and prevent the robot to reach the *safe point*. The robot will remain in *Drive Backward* mode trying to reach the *safe point*, which is no longer possible. This is avoided by deactivating the *Drive Backward* after certain time. Path is replanned from the current point to the final destination and the robot moves forward. Another situation could be that the robot is moving forward in a corridor. At the end it found that a dynamic obstacle has made the final destination unreachable. It can neither move forward nor can take a turn. It tries to drive backward to the *safe*

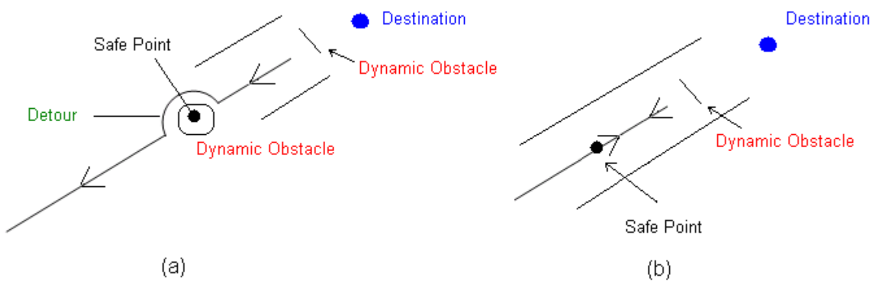


Figure 6: Scenerios of Driving Back

point and replans the path from *safe point* to the final destination and moves forward. Being in a corridor, to-and-fro motion starts resulting in an unreachble final destination. This issue could be resolved by considering two *safe points*. But there may be situations where two *safe points* are not sufficient and hence more *safe points* are required. Experiments conducted in the living environment demonstractes that one *safe point* approach is sufficient and does not require manipulation of multiple *safe points*.

4. EXPERIMENTS

Several experiments have been conducted in the BelAmI Lab at the Fraunhofer IESE (see Fig. 2), a testing facility which is similar to a real apartment, with the aim to test the performance of the system for autonomous navigation. In two experiments, the robot had to drive from one location in the BelAmI Lab to another. It was provided with an initial map containing *only* the walls of the rooms. Furniture and other objects standing or lying on the ground were not represented in the map and so the initially created paths led through obstacles. The difficulty for the control system was to detect path obstructions and find a way around them. In both experiments, the robot had to drive from the entrance to the kitchen. Fig. 7 shows the furniture in the living room through which ARTOS has to plan path while navigating from entrance to the kitchen.

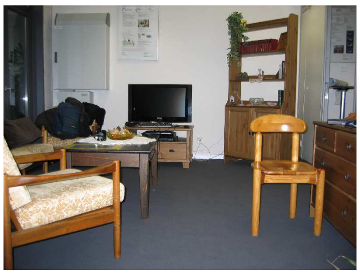


Figure 7: The main obstacles in the living room that ARTOS encountered when driving from the entrance to the kitchen.

The robot's path during the first experiment is shown in Fig. 10. The gaps in the path are caused by the RFID-based pose corrections, which make the estimated pose "jump". A diagram of its velocity is shown in Fig. 8. As can be seen, the robot did not comeback while following the path. This is an important result as it demonstrates that the mapping and obstacle avoidance behaviours detected the obstructions so early that a way leading around them could be calculated before the robot got stuck. Note that the figure

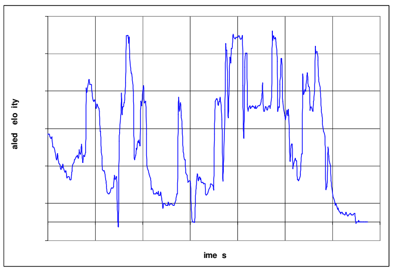


Figure 8: ARTOS' velocity during its tour from the entrance to the kitchen.

depicts the velocity scaled to $[-0.5; 0.5]$, so 0.5 is the maximum value.

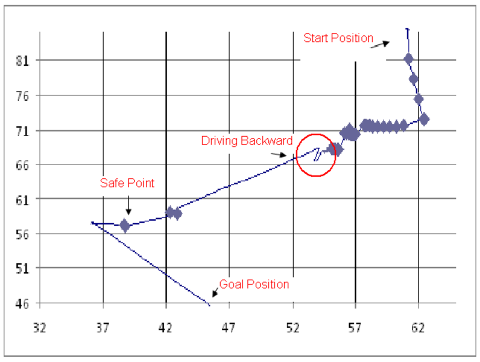


Figure 9: Path to the destination obstructed by obstacles

In the second experiment, the robot took a detour before entering the kitchen. This was caused by the collision avoidance system which detected the door and made the robot turn in the wrong direction. As can be seen in Fig. 11, the navigation system adapted to the new situation and led the robot along a curve back to the correct course.

In another experiment, the robot had to move from start position to end position but this time the path to the destination was completely blocked by dynamic obstacle and the robot had to take an alternate route. Fig. 9 shows the path followed by ARTOS. The robot has to come back from the obstruction, replan the path and then followed a different path to the destination.

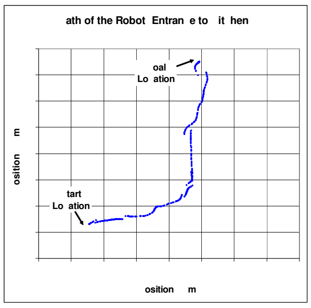


Figure 10: The path of experiment 1.

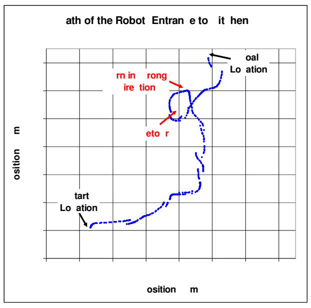


Figure 11: The path of experiment 2.

Many other experiments have been conducted in the BelAmI Lab with varying start and goal points. In some experiments, the collision avoidance caused a detour, but eventually the targets were reached.

5. CONCLUSION AND FUTURE WORK

The mapping and navigation systems presented here enable the robot, ARTOS, to move autonomously in living environments. Considering the dynamics of living environments, different sensor systems are being used as basis for the map creation so that a large variety of obstacles, like furniture, are recognised. The navigation system, using the maps, is able to recover from paths that may lead to dead-end because of dynamic obstacles. In future versions, additional components may be added to make the navigation system react even faster to obstructions. Parts of the anti-collision system shall be revised to allow for turning in small passages.

Acknowledgements

We gratefully acknowledge the funding of the BelAmI project by the State Ministry of Science, Education, Research and Culture of Rhineland-Palatinate, by the Fraunhofer-Society and by the German Federal Ministry of Education and Research.

6. REFERENCES

[1] C. Armbrust, J. Koch, U. Stocker, and K. Berns. Mobile robot navigation support in living environments. In *20. Fachgespräch Autonome Mobile Systeme (AMS)*, pages 341–346, Kaiserslautern, Germany, October 2007.

[2] A. Cherubini, G. Oriolo, F. Macrì, F. Aloise, F. Babiloni, F. Cincotti, and D. Mattia. Development of a multimode navigation system for an assistive robotics project. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 2336–2342, Roma, Italy, April 10–14 2007.

[3] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with rfid technology. In *International Conference on Robotics and Automation (ICRA)*, New Orleans, 2004.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions of Systems Science and Cybernetics*, volume 4, pages 100–107, July 2 1968.

[5] J. Koch, C. Armbrust, and K. Berns. Small service robots for assisted living environments. In *VDI/VDE Fachtagung Robotik*, Munich, Germany, June 11-12 2008.

[6] J. Koch, M. Reichardt, and K. Berns. Universal web interfaces for robot control frameworks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, September 22-26 2008.

[7] V. Kulyukin, A. Banavalikar, and J. Nicholson. Wireless indoor localization with dempster-shafer simple support functions. Technical report, Computer Science Assistive Technology Laboratory, Department of Computer Science, Utah State University, 2005.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence*, pages 593–598, aug 2002.

[9] L. Montesano, J. Minguez, J. Alcubierre, and L. Montano. Towards the adaptation of a robotic wheelchair for cognitive disabled children. In *Proceedings of the Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, pages 710–716, Beijing, China, October 9–15 2006.

[10] R. Ouellette and K. Hirasawa. A comparison of slam implementations for indoor mobile robots. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1479–1484, 29 2007-Nov. 2 2007.

[11] M. Proetzsch, T. Luksch, and K. Berns. The behaviour-based control architecture ib2c for complex robotic systems. In *30th Annual German Conference on Artificial Intelligence (KI)*, pages 494–497, Osnabrück, Germany, September 10-13 2007.

[12] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of IEEE Int. Conference on Robotics and Automation*, pages 802–807, Atlanta, 1993.