Using a Depth Camera for Indoor Robot Localization and Navigation

João Cunha and Eurico Pedrosa and Cristóvão Cruz and António J.R. Neves and Nuno Lau

Abstract— **Depth cameras are a rich source of information for robot indoor localization and safe navigation. The recent availability of the low-cost Kinect sensor provides a valid alternative to other available sensors, namely laser-range finders. This paper presents the first results of the application of a Kinect sensor on a wheeled indoor service robot for elderly assistance. The robot makes use of a metric map of the environment's walls and uses the depth information of the Kinect camera to detect the walls and localize itself in the environment. In our approach an error minimization method is used providing real-time efficient robot pose estimation. Furthermore, the depth camera provides information about the obstacles surrounding the robot, allowing the application of path-finding algorithms such as D* Lite achieving safe and robust navigation. Using the proposed solution, we were able to adapt a robotic soccer robot developed at the University of Aveiro to successfully navigate in a domestic environment, across different rooms without colliding with obstacles in the environment.**

I. Introduction

Depth cameras are revolutionizing robot perception as a replacement for sensors such as laser-range finders and stereo-vision systems. While depth cameras have been available for some years, the market price is an hindering factor for their application in Robotics. However the easily available Microsoft Kinect depth camera provides decent quality depth information of the surrounding environment at a highly competitive price. This factor makes the Kinect significantly appealing as is easily demonstrated by its adoption by research centers across the globe [1][2].

In this paper we present the application of a Kinect sensor in order to adapt a robotic soccer platform to navigate in an indoor unstructured environment. The robot used in this work is a CAMBADA robot. Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture (CAMBADA) is the robotic soccer team of the University of Aveiro (UA) competing in RoboCup Middle Size League (MSL). The CAMBADA project started in 2003 by researchers of the Transverse Activity on Intelligent Robotics (ATRI) research group within Institute of Electronics and Telematics Engineering of Aveiro (IEETA) and students of the Department of Electronics, Telecommunications and Informatics (DETI) from the UA. Since its origin, the CAMBADA team has competed in several national and international competitions having won the last five national championships as well as the 2008 edition of RoboCup World Championship. More recently, the CAMBADA team has placed in third in both RoboCup 2009 in Graz, Austria and RoboCup 2010 in Singapore.

Building on the success and experience of the robotic soccer team, CAMBADA started to pursue challenges in different fields of application, namely the development of a service robot to compete in the RoboCup@Home league[1]. There is no better proof of the successful application of soccer robots in domestic environments than this league. The RoboCup@Home league was created in 2006 from the need to place more emphasis on real world problems, not addressed in robotic soccer [3]. This league is currently the largest league of the RoboCup initiative and includes a vast number of teams that started as soccer teams and then evolved to this robotic paradigm.

The remainder of this paper is structured as follows. Section II discusses related work on indoor robot localization and navigation. The methods applied to extract useful information from the retrieved depth image are presented in Section IV. The used localization algorithm is discussed in Section V. The applied methods for obstacle avoidance and path-finding are detailed in Section VI. The experimental results demonstrating the robustness of the approach are presented in Section VII. Section VIII draws the conclusions of the work described in this paper.

II. Related Work

A. Perception

Current work related to environment perception that includes depth sensing is centred around laser range finders and stereo vision systems. These approaches are either accurate but slow, as in the case of 3D sensing based on laser range finders [4] or computationally intensive, as in the case of stereo vision systems [5]. The Kinect depth sensor brings a fast and relatively high resolution solution for depth sensing, while being considerably cheaper than the referred alternatives [6].

Similar work to the one presented here for wall detection was already presented in [7], with the "Wall without Contradiction method". The version used here is a very simplified alternative, but the core ideia remains the same, which is that a wall is only a wall if the robot can't see behind it.

B. Indoor Localization

Mobile robot indoor localization has been referred as "the most fundamental problem to providing a mobile robot with autonomous capabilities" [8].

Robot localization has been subject of active research in the last decades. All the *de facto* algorithms share the common characteristic of representing the internal pose belief as a probabilistic distribution [9].

Developed approaches can be roughly divided in two different methodologies: Markov and Gaussian Localization or

[1]www.robocupathome.org

Monte-Carlo Localization. A classical and highly successful example of the application of Markov localization is the Minerva museum tour guide robot [10], which successfully navigated in crowded environments. A famous example of Gaussian Localization is based on Kalman filters [11]. Kalman filters are an efficient method for robot tracking. However they assume Gaussian noise and linear motion models. Extensions to the classic Kalman algorithm such as the Extended Kalman filter are widely used as they relax such assumptions. Since Kalman filters can only represent unimodal probabilistic distribution, a common method is to use Multi-Hypothesis Tracking filter, which represent the internal belief by a set of Gaussians.

Another well established approach is the application of particle filters combined with motion models, also known as Monte Carlo Localization [12][13]. The inner belief is represented as a set of particles (hypothesis) that are randomly sampled over the environment. Although more recent than other approaches, Monte Carlo Localization has become one of the most popular localization algorithms in robotics. This is usually attributed to the fact that Monte-Carlo Localization can approximate any probabilistic distribution, such as complex multimodal distributions. This is an obvious advantage in the presence of ambiguous indistinct topologies.

C. *Robotic Navigation*

In robotics, certain tasks requires the robot to navigate in indoor environment. Traditionally the environment is assumed to be static and all objects rigid [14]. A *deliberative approach* in this type of environments is, for example, cell decomposition [15] with A* for shortest path computation. However, an indoor environment has an intrinsic uncertainty and complexity that can not be overcome with this approach. As alternative, a *reactive approach* responds to the sensors stimuli to generate an adequate behavior. An example is the use of potential field methods (PFM) for local planning like *obstacle avoidance* [16]. Nevertheless, *local minima* is a best-known problem in reactive navigation such as PFM [17]. The *local minima* problem can be solved by detecting the *trap-situation* and recalculate the navigation plan with the new information [17].

Another common practice it to combine global path planning with local path planning, such as [18], for fast incremental path planning. Initially, it calculates the shortest path between two points and, as it navigates the unknown, adjusts the path plan accordingly the perceived environment. As consequence, *obstacle avoidance* results naturally from the incremental path planning.

III. System Architecture

The robotic platform used is based on the CAMBADA robotic soccer platform. The robot has a conical base with radius of 24 cm and height of 80 cm. The physical structure is built on a modular approach with three main modules or layers.

The top layer has the robot vision system. Currently the robot uses a single Microsoft Kinect camera placed on top of the robot pointing forwards. This is the main sensor of the robot. The retrieved information is used for localization and path-planning to predefined goals.

The middle layer houses the processing unit, currently a 13” laptop, which collects data from the sensors and computes the commands to the actuators. The laptop executes the vision software along with all high level and decision software and can be seen as the brain of the robot. Beneath the middle layer, a network of micro-controllers is placed to control the low-level sensing/actuation system, or the nervous system of the robot. The sensing and actuation system is highly distributed, using the CAN protocol, meaning the nodes in the network control different functions of the robot, such as motion, odometry and system monitoring.

Finally, the lowest layer is composed of the robot motion system. The robot moves with the aid of a set of three omni-wheels, disposed at the periphery of the robot at angles that differ 120 degrees from each other, powered by three 24V/150W Maxon motors (Figure 1). With this wheel configuration, the robot is capable of holonomic motion, being able to move in a given direction independently of its orientation.

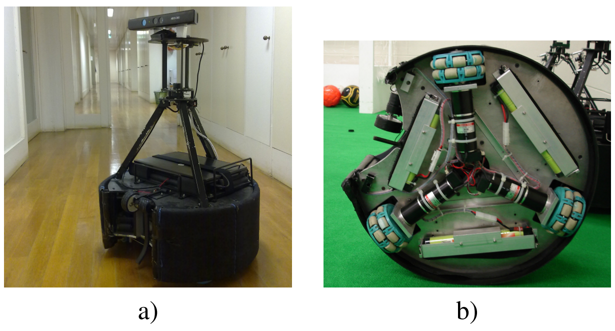

Fig. 1. CAMBADA hardware system: a) The robot platform. b) Detailed view of the motion system.

Following the CAMBADA hardware approach, the software is also distributed. Therefore, five different processes are executed concurrently. All the processes run at the robot’s processing unit in Linux.

Inter-process communication is handled by means of a RealTime DataBase (RTDB) [19] which is physically implemented in shared memory. The RTDB is divided in two regions, the local and shared regions. The local section allows communication between processes running in the robot. The shared section implements a Blackboard communication paradigm and allows communication between processes running in different robots. All shared sections in the RTDB are kept updated by an adaptive broadcasting mechanism that minimizes delay and packet collisions.

The processes composing the CAMBADA software are (Figure 2):

- **Vision** which is responsible for acquiring the visual data from the Kinect sensor.
- **Agent** is the process that integrates the sensor information and constructs the robot’s worldstate. The agent

then decides the commands to be applied, based on the perception of the worldstate.

- **Comm** handles the inter-robot communication, receiving the information shared by other robots and transmitting the data from the shared section of the RTDB.
- **HWcomm** or hardware communication process is responsible for transmitting the data to and from the low-level sensing and actuation system.
- **Monitor** that checks the state of the remaining processes, relaunching them in case of abnormal termination.

Given the real-time constraints of the system, all process scheduling is handled by a library specifically developed for the task, the *Process Manager* [20].

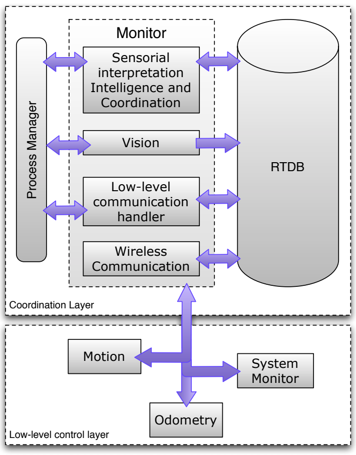

Fig. 2. CAMBADA software architecture.

A. Monitoring station

The monitoring station, also known as basestation, has a determinant role both during the development of an autonomous assistant robot capability as well during its application. The basestation is an adapted version of the CAMBADA team basestation [21] taking in consideration a set of requirements that emerge from the development of a service and assistive robot (Figure 3).

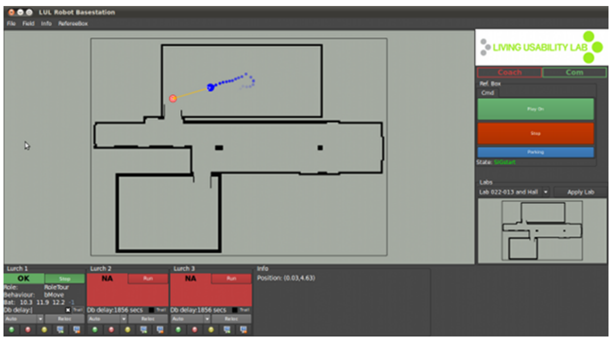

Fig. 3. CAMBADA basestation GUI. The GUI is divided in three panes. The lower pane shows the internal state of the robots. The center pane draws the indoor *blueprint* and the robots location. The right pane hold the robots control panel box (e.g. start, stop) and several operational visual flags.

The basestation application provides a set of tools to perform the control and monitoring of the robot. Regarding the control activity, this application allows high level control of the robot by sending basic commands such as *run*, *stop* and *docking*. It also provides a high level monitoring of the robot internal state, namely its batteries status, current role and behavior, indoor self-localization, current destination point, breadcrumb trail, etc.

Furthermore, this application provides a mechanism that can be used to enforce a specific behavior of the robot, for debugging purposes.

IV. Visual Perception

Humans rely heavily on vision or vision based abstractions to acknowledge the world, to think about the world and to manipulate the world. It is only logical to empower artificial agents with a vision systems with capabilities similar to the human vision system.

The vision subsystem of this robot is constituted by a single depth sensor, the Kinect, fixed at the top of the robot. It is accessed through the freenect [22] library, and provides a depth and color view of the world. With the depth information, we create a 3D metric model of the environment. Using this model, we then extract relevant environment features for the purpose of localization and navigation, namely the walls and the obstacles, The proposed methods consider that the height and pitch of the camera relatively to the ground plane remain constant, parameters that are set when the system is calibrated.

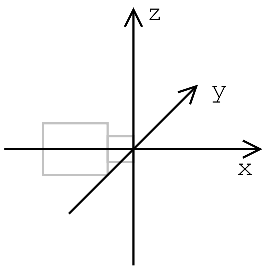

Fig. 4. Camera coordinate system

A. Pre-Processing

Both wall and obstacle detection is done using only the depth image, which has 640×480 pixels. The image is subsampled by a factor of 5 in both dimensions, leaving us with a 128×96 image, which has proven to contain sufficient information for wall and obstacle detection. This decision is not based on current time constraints, but was made to account for future project developments.

B. Walls

A wall is a structure that delimits rooms. It is opaque and connects the floor to the ceiling.

With this simple definition in mind, the approach we follow to identify the walls in the environment is to select all the points with height, relative to the floor, lower than the ceiling height and perform a column-wise search on the remaining points of the image for the point which is farthest from the robot, along the x axis, according to the coordinate system shown in Figure 4. The retrieved points are then used in later stages of processing to allow for robot localization.

C. Obstacles

An obstacle is an occupied volume with which the robot can collide.

The obstacle detection method is similar to the one used for wall detection. To detect the obstacles, we reject the points that have an height, relative to the floor, greater than the robot height plus a margin to account for noise, or that belong the floor. The point is considered to be floor if it's height is lower than a threshold, method that proved to be good enough to cope with the noise. We then perform a column-wise search on the remaining points of the image for the point closest to the robot along the x axis.

At the end of this process we have 128 obstacle points that can be used for occupancy belief update.

V. Indoor Localization

For indoor localization, we successfully adapted the localization algorithm used by the CAMBADA team to estimate a robot position in a robotic soccer field. The algorithm was initially proposed by the MSL team Brainstormers Tribots [23].

The Tribots algorithm [23] constructs a FieldLUT from the soccer field. A FieldLUT is a grid-like data structure where the value of each cell is the distance to the closest field line. The gradient of the FieldLUT values represents the direction to the closest field line. The robot detects the relative position of the field line points through vision and tries to match the seen points with the soccer field map, represented by the FieldLUT. Given a trial position, based on the previous estimate and odometry, the robot calculates the matching error and its gradient and improves the trial position by performing an error minimization method based on gradient descent and the RPROP algorithm [24]. After this optimization process the robot pose is integrated with the odometry data in a Kalman Filter for a refined estimation of the robot pose.

To apply the aforementioned algorithm in a indoor environment the concept of white line was replaced with the walls. In an initial phase, given a map of the environment, usually the building blueprints, a FieldLUT list is created that contains all possible configurations of seen walls for different positions (Algorithm 1). By testing a grid of points over the map, a new FieldLUT is created, and added to the FieldLUT list, when a new configuration of seen walls is detected. As the robot moves through the environment, it dynamically loads the FieldLUT corresponding to its position, which should consist of the walls seen in that part of the map.

The need to use a set of FieldLUTs instead of a single FieldLUT for the entire map arises from the local minimum problem inherent to gradient descent algorithms. Since the walls in a domestic environment have an associated height which is naturally higher than the robot, from a given point in the environment there is usually a set of walls that are out of the line-of-sight of the robot. This scenario doesn't occur in a robotic soccer field where the lines are co-planar with the field. Therefore using a single FieldLUT could match the wall points extracted from the captured images to unseen walls, resulting in erroneous self-localization.

The described method does not solve the initial localization problem. This is solved by applying the visual optimization process on different trial positions evenly spaced over the known map. To reduce the search space of the initial localization, the initial heading of the robot is given by a digital compass.

Algorithm 1 Build FieldLUTs

```
Input: map
  $wallList \leftarrow \emptyset$
  $fieldLUTlist \leftarrow \emptyset$
  $mapping \leftarrow \emptyset$
  $pointList \leftarrow map.generateGrid()$
  for all $point$ in $pointList$ do
    $visibleWalls \leftarrow map.findVisibleWalls(point)$
    if not $wallList.find(visibleWalls)$ then
      $wallList.add(visibleWalls)$
      $fieldLUT \leftarrow buildFieldLUT(visibleWalls)$
      $fieldLUTlist.add(fieldLUT)$
    end if
    $mapping.add(< point, fieldLUT.id >)$
  end for
  return $< fieldLUTlist, mapping >$
```

VI. Goal Directed Navigation

The robotic agent receives a sequence of goal points to go to in a patrolling manner. As it arrives the goal point it decomposes its path to the next goal point in intermediate goal points. The robot navigates between intermediate goal points in a straight line.

The considered metric map is a discrete two-dimensional occupancy grid. Each cell-grid (x, y) has an associated value that yields the believed occupancy. The navigation plan is calculated by a path-finding algorithm supported by the probabilistic occupancy map of the environment. Because of the dynamic nature of the environment the robotic agent uses D* Lite [25], an incremental path-finding algorithm that avoids full re-planning in face of changes in the environment. This methodology enables obstacle avoidance seldom based on incremental path planning (Figure 5).

The environment is represented in an occupancy map implemented by the OctoMap library [26]. Although OctoMap is capable of creating 3D occupancy maps, the environment is projected onto the ground plane, thus constructing a 2D occupancy map of the environment. Each change in the environment, tracked by the occupancy map, is reflected in the corresponding cell-grid value used in the path-finding algorithm.

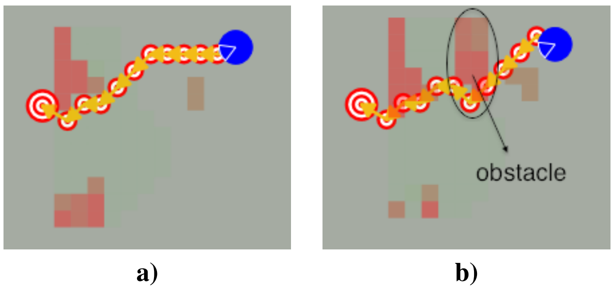

Fig. 5. Incremental path planning: **a)** Path planned to achieve the goal point. **b)** A new obstacle appeared in the path of the robot resulting in a re-planned path adjusted to the changes of the environment.

The perceived obstacles (x, y) points (section IV) are used to update the occupancy map. For each obstacle (x, y) point the corresponding (x, y) node of the occupancy map is updated by casting a ray from the robot current point to the target node, excluding the latter. Every node transversed by the ray is updated as free and the target updated as occupied. However a $maxrange$ value is set to limit the considered sensor range. If an obstacle is beyond the $maxrange$, only the nodes transversed up to $maxrange$ are updated while the remaining nodes remain unchanged, including the target node. However, due to the limited vertical field of view of the Kinect sensor, target nodes where the Kinect sensor can not *see* the floor are updated as occupied without making any assumption about the occupancy of the closer nodes. This is made to prevent the *freeing* of nodes at the Kinect sensor vertical blind region.

VII. Results

Using the work described our robot is able to navigate in a $15m \times 20m$ office environment, consisting in two rooms connected by a long corridor (Figure 3).

The algorithms described in section IV process the images captured by the Kinect depth camera and extract the visible walls and obstacles. Using solely depth information the robotic agent is able to detect walls without having to perform color calibration or without being susceptible to natural lighting conditions (Figure 6).

The robot is able to localize itself in the environment using solely information about the walls coming from a single Kinect depth camera, even in long rooms or corridors, where the seen walls are considerably distant, largely affecting the Kinect measurements with noise. While our localization approach is robust to static environments, problems arise when wall sections become occluded, for instance when a person gets close to the robot. This greatly induces noise in the gradient descent algorithm, since the occluding obstacle is considered as a wall.

With the capability to localize itself in the environment, we are able to provide the robot with goals. A predefined goal is a (x,y) pair representing a target position. Using obstacles detected by the Kinect depth camera, we build a probabilistic occupancy grid, over which a path is planned from the robot current position to the desired goal. Using this approach the robot is able to successfully navigate to the predefined goals avoiding obstacles in its way(Figure 7).

Stage	Time (ms)
Perception	3
Mapping	1
Localization	2
Path Generation	1

TABLE I
COMPUTATION TIMES

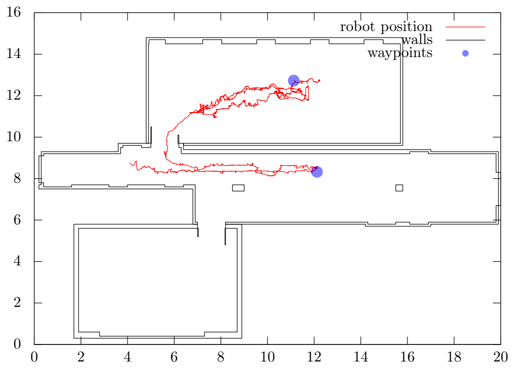

Fig. 7. Robot path between two previously predefined goals. Initially the robot wanders along the wall until it discovers a clear path to the target point. The robot begins its movement near 12.1,12.7.

Table I sumarizes the average time spent by the various stages of computation, on an Intel® Core^{TM} i3 CPU @ 2.40GHz with 2GB of memory.

VIII. Conclusion and Future Work

In this paper we presented an approach to enable a real robot to localize and navigate in an indoor environment using a single Kinect camera. Our approach is based on an a-priori provided map consisting on the walls of the environment. While the robot is able to successfully navigate in static scenes the presence of dynamic obstacles such as people can occlude walls. Future work will consider improvements to the wall detection method to filter false wall points.

While the error minimization method used is able to localize the robot, it fails to provide an accurate initial position in the presence of environments with ambiguous topology. The application of Monte-Carlo Localization with the ability to represent multi-modal probability distributions should be able to address this shortcoming efficiently.

Acknowledgements

This work is part of the COMPETE - Programa Operacional Factores de Competitividade and the European Union (FEDER) under QREN Living Usability Lab for Next Generation Networks (http://www.livinglab.pt/).

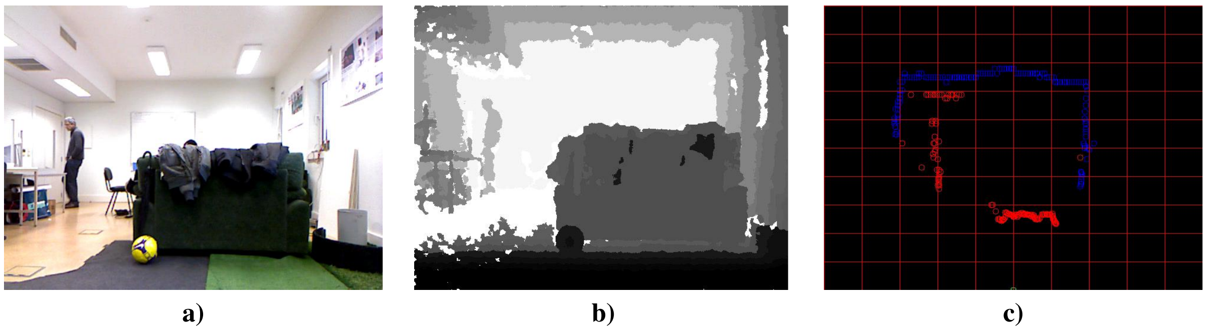

Fig. 6. Kinect vision system: **a)** The image captured by the Kinect rgb camera. **b)** The same image captured by the Kinect depth camera. **c)** The 2D vision of the extracted information of the depth camera, the blue points are walls and the red points are obstacles. The robot is placed in the bottom center of the image.

REFERENCES

[1] A. A. Ramey, V. González-Pacheco, and M. A. Salichs, "Integration of a low-cost rgb-d sensor in a social robot for gesture recognition," in *Proceedings of the 6th International Conference on Human Robot Interaction, HRI 2011, Lausanne, Switzerland, March 6-9, 2011*, A. Billard, P. H. K. Jr., J. A. Adams, and J. G. Trafton, Eds. ACM, 2011, pp. 229–230.

[2] "Willow garage - ros 3d contest," http://www.ros.org/wiki/openni/Contests/ROS%203D/, online, Accessed in May 2010.

[3] T. van der Zant and T. Wisspeintner, "Robocup x: A proposal for a new league where robocup goes real world," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Computer Science, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds., vol. 4020. Springer, 2005, pp. 166–172.

[4] O. Wulf, K. Arras, H. Christensen, and B. Wagner, "2d mapping of cluttered indoor environments by means of 3d perception," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, 26-may 1, 2004, pp. 4204 – 4209 Vol.4.

[5] M. Brown, D. Burschka, and G. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 993–1008, 2003.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments," in *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[7] H. Moradi, J. Choi, E. Kim, and S. Lee, "A real-time wall detection method for indoor environments," in *IROS*, 2006, pp. 4551–4557.

[8] I. Cox, "Blanche - an experiment in guidance and navigation of an autonomous vehicle," *IEEE Transactions on Robotics and Automation*, 1991.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, R. C. Arkin, Ed. The MIT Press, 2005.

[10] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. R. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *I. J. Robotic Res.*, vol. 19, no. 11, pp. 972–999, 2000.

[11] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[12] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artif. Intell.*, vol. 128, no. 1-2, pp. 99–141, 2001.

[13] C. C. T. Kwok, D. Fox, and M. Meila, "Adaptive real-time particle filters for robot localization," in *ICRA*. IEEE, 2003, pp. 2836–2841.

[14] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566 –580, aug 1996.

[15] J. Latombe, *Robot motion planning*. Springer Verlag, 1990.

[16] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, apr 1991, pp. 1398 –1404 vol.2.

[17] R. Tilove, "Local obstacle avoidance for mobile robots based on the method of artificial potentials," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, may 1990, pp. 566 –571 vol.1.

[18] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 1, 2002, pp. 968 – 975 vol.1.

[19] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L. S. Lopes, "Coordinating distributed autonomous agents with a real-time database: The cambada project," in *ISCIS*, ser. Lecture Notes in Computer Science, C. Aykanat, T. Dayar, and I. Korpeoglu, Eds., vol. 3280. Springer, 2004, pp. 876–886.

[20] P. Pedreiras and L. Almeida, "Task management for soft real-time applications based on general purpose operating systems," in *Robotic Soccer*. I-Tech Education and Publishing, 2007, pp. 598–607.

[21] N. M. Figueiredo, A. J. R. Neves, N. Lau, A. Pereira, and G. A. Corrente, "Control and monitoring of a robotic soccer team: The base station application," in *EPIA*, ser. Lecture Notes in Computer Science, L. S. Lopes, N. Lau, P. Mariano, and L. M. Rocha, Eds., vol. 5816. Springer, 2009, pp. 299–309.

[22] libfreenect homepage, "Available http://openkinect.org/wiki/Main_Page," accessed in February 2011.

[23] M. Lauer, S. Lange, and M. Riedmiller, "Calculating the perfect match: An efficient and accurate approach for robot self-localization," in *RoboCup*, ser. Lecture Notes in Computer Science, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds., vol. 4020. Springer, 2005, pp. 142–153.

[24] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*, 1993, p. 586–591.

[25] S. Koenig and M. Likhachev, "D* Lite," in *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, Alberta, Canada, July 2002, pp. 476–483.

[26] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010, software available at http://octomap.sf.net/. [Online]. Available: http://octomap.sf.net/