# CONTROL OF MOBILE ROBOTS VIA INTERNET

**Andrew Wasley Barbosa**
USP - University of São Paulo - Department of Computer Science and Statistics
andrew@icmc.usp.br

**Rodrigo Elias Bianchi**
USP - University of São Paulo - Department of Computer Science and Statistics
rodrigo@icmc.usp.br

**Roseli Ap. Francelin Romero**
USP - University of São Paulo - Department of Computer Science and Statistics
rafrance@icmc.usp.br

***Abstract.***
*Remote robot control systems are being developed for a wide range of applications, including robot control at dangerous environments, delivery tasks and security monitoring. This paper presents a robot control system with a web interface. The control system is divided in two subsystems: Central Control System (CCS) and the web interface for robot monitoring. Three modules compose the CCS: mapping module, localization module and path planning module. The mapping module models the environment. The localization module estimates the robot positioning in its environment. The path planning module is responsible for controlling the robots actuators, using the data supplied by the other modules, so that the robot can travel between 2 points without hitting objects. The web interface is responsible for sending new tasks for the CCS subsystem, monitoring the robot in an environment map and to show images captured by the robot during the task accomplishment. This remote web interface helps users not familiarized with robotics to operate a robot while keeping the operator away from dangerous environments. Experiments are being conducted at a real environment with a Pioneer I robot.*

***Keywords:*** *Autonomous Navigation, Mobile Robots, Teleoperation*

## 1. Introduction

In the last years, the Internet use grew up very much, taking part of our daily lives as for shopping, for reading news or even for entertainment. For this reason, some researchers have started to use the internet for controlling teleoperation systems [Grange et al., 2000].

In this paper we are concerned with teleoperation systems for controlling mobile robots. The control of a robot is in general a hard task, since the robot should not harm people or cause damage to objects existing in its environment. One of the limitations found in robot controlling is that generally only specialists are able to control the robot. Hence, there are some works in the literature that try to enhance the interaction among users and robots, making it friendlier.

Some interesting examples of robots controlled by the Internet can be cited. Among them, one of the most important robot, is the Xavier [Simmons et al., 1999] of Learning Lab, Carnegie Melon University - CMU - USA; the Rhino robot[1] [Buhmann et al., 1995] of Bonn University, Germany, which was controlled during as a tourguide in the Deutsches Museum, in Bonn. Further, more recently, the Minerva robot [2], belonging also to Learning Lab, CMU - USA. From all of the robots cited, Xavier was the robot that spent more time interacting with people through the Internet. People could to access the site[3] and send some tasks to be realized by Xavier. Approximately 30,000 requisitions were sent to Xavier and this robot had to walk 210 km for realizing the tasks required by the users. All of these robots have approximately 1,10m of height.

In this work, a system is being proposed for controlling small robots through the Internet. This system has been tested in the robot Pioneer I, belonging to LABIC[4] - USP - Brazil, which height is approximately 30 cm. It will be presented the architecture of the system and how the interface works in order the users can teleoperate this robot through the Web.

This paper is organized as follows. In section 2. a description of all of modules that compose the system and how these modules interact each others are presented; in section 3. some results obtained with the various tests performed are discussed. Finally, conclusions and future works are presented in section 4.

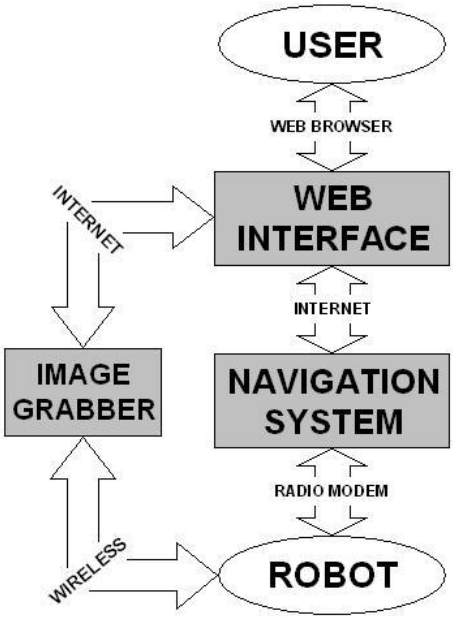



Figure 1: Architecture and functioning of the System

## 2. The Proposed System

In section, the modules that constitute the system are presented as well as the techniques utilized to implement each one of them. The interface module, navigation module and an image grabber module constitute the system. The interface module contains an environment map and a picture of this environment. The image grabber module is responsible to capture images during the entire route driven by the robot. The navigation module is responsible by the autonomous navigation of the robot and is constituted by other three modules: mapping, localization and path planning, described in details in 2.1

In Figure 1, it can be seen a layout of the proposed system. Its functioning consist of following steps. The user presents a requisition to the system by the interface. The interface sends this requisition, through the Internet, to a server that runs an autonomous navigation system. This system receives the requisition from the interface and decides which is the best action to be taken by the robot. This action is sent to the robot via radio modem. The robot by its turn executes the action and returns the information about its new position in the environment, through the radio modem, to the navigation module. The navigation module returns this information to the interface module, which is responsible for updating this new position in the environment map. During all these steps the image grabber module is responsible for updating the picture of the environment in the interface.

The role of the interface module is to receive requisitions presented by the user. These requisitions consist of positions in a map of the environment that the user desires the robot to go. Furthermore, it is responsible to show the user in what position the robot is during the route performed and display pictures of this environment. These images are captured during the navigation of the robot.

The navigation module works in the following manner. First at all, the robot has to explore the environment and build the corresponding map. In our case, the robot builds autonomously a metric map of its environment. This map contains information about the placement of objects and walls in its environment. After, it is generated a topological map based on the metric map obtained and it will be utilized in the path planning. The topological map is obtained manually but we intended to improve this in order for the robot to build it autonomously. The route that robot has to follow is determined by the path planning module presented in 2.1.3Following, the navigation module is described in more detail.

[1]http://www.informatik.uni-bonn.de/ rhino/
[2]http://www.cs.cmu.edu/ Minerva
[3]http://www.cs.cmu.edu/ Xavier
[4]Laboratory of Computational Intelligence

### 2.1 The Navigation Module

The autonomous navigation module is responsible for the control of the robot in order to make it able to do tasks in an autonomous way. It is known that a navigation system has to deal with the following tasks: environment mapping, robot localization and path planning, which are discussed in this section. The modules that compose the autonomous navigation system are show in Figure 2.

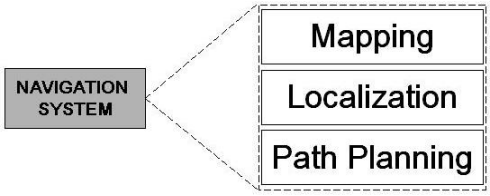



Figure 2: Modules of autonomous navigation system

#### 2.1.1 Mapping

Autonomous mapping of environments is very important because the task of teaching a robot to know where it is requires a detailed map of the environment. This is a hard task because even if one has a map of environment obtained manually, it could not be adequate to the robot since the robot perception of the environment certainly is different of our vision of the environment [Carvalho et al., 2004]. To do tasks in indoor environments, robots must be able to learn and maintain models of the environment. The construction of an ideal environment model is a theme of active research nowadays. There are some points that affect the construction of a precise environment model [Thrun and Büecken, 1996], such as:

i. **Sensors**. Sensors often are not capable to directly measure the quantity of interest(such as the exact location of obstacles).

ii. **Perceptual limitations**. The perceptual range of most sensors is limited to a small range close to the robot. To acquire global information, the robot has to actively explore its environment.

iii. **Sensor noise**. Sensor measurements are typically corrupted by noise, the distribution of which is often unknown (its rarely Gaussian).

iv. **Drift/slippage**. Robot motion is inaccurate. Odometric errors accumulate over time.

v. **Complexity and dynamics**. Robot environments are complex and dynamic, making it principally impossible to maintain exact models.

vi. **Real-time requirements**. Time requirements often demand that the internal model must be simple and easily accessible. For example, fine-grain CAD models are often disadvantageous in actions that must be generated in real-time.

For robots to be able to navigate, it is necessary to provide it an environment map. The map could be built either by hand, i. e., it could be obtained by a designer using any CAD program or in a automatic manner, through environment exploration by the robot. In this latest case, the mapping consists of the task of creating a model of the environment based on the sensor signals. So, the main problem associated with mapping is related with the presence of noisy and reading errors introduced by the sensors.

In our case, the mapping method used in our system is that known as *Occupancy Grid* [Elfes, 1989], which generates a metric map of the environment automatically.

In *Occupancy Grid*, to understand the sensor data (distance measures), it is used an stochastic model of sensor defined by a probabilistic density function, $p(r|z)$, which relates the input $r$ with the space parameter $z$ correspondent to a position in a map.

This density function is subsequently used in a procedure based on Bayesian estimative to determine the probabilities of a cell, in a bidimensional grid. Let us to denote an Occupied Cell by OCC and an Empty Cell by EMP, where $P[s(C) = OCC] + P[s(C) = EMP] = 1$ and $s(C)$ denotes the state of the variable relating to cell $C$.

From the actual estimative of the state of each cell $C_i$, $P[s(C_i) = OCC|\{r\}_t]$, based on observations $\{r\}_t = \{r_1, ..., r_t\}$ and given a new observation, $r_{t+1}$, the estimative updated is given by:

$$P[s(C_i) = OCC|\{r\}_{t+1}] = \frac{p[r_{t+1}|s(C_i) = OCC]P[s(C_i) = OCC|\{r\}_t]}{\sum_{s(C_i)} p[r_{t+1}|s(C_i)]P[s(C_i)|\{r\}_t]} \quad (1)$$

In this recursive formulation, the previous estimative of the state of cell, $P[s(C_i) = OCC|\{r\}_t]$, serves as priori estimative and it is obtained directly of grid. The new estimative is then stored in the map. In this way, a map is obtained that represents the probabilities of occupation of each cell.

In our system, the mapping is realized in the following manner. Initially, the robot builds a small part of the map, which is corresponding to the part of environment that it can sense through the sensors. After that, the robot explores the environment taking in count the built map. To follow, a method of localization, described in the next section, is used for determining the robot's position in that part of the map. Based on this information, a new part of map is built and the robot explores a little more in this new area. Again, a method of localization is used for determining the robot's position in that part of the map. This process is repeated until the entire map has been built.

Further, it is also generated a topological map [Matarić, 1994][Kortenkamp and Weymouth, 1994] to make the path planning easier. Here, the topological map has been built by a graph created by hand based on the metric map. More details about the mapping module could be found in [Bianchi, 2003].

The localization method used to help the construction of the environment map is described in the next section.

### 2.1.2 Localization

Localization of mobile robots is a task that consists of estimating the robot position in a certain environment. The information to estimate the robot position comes from the sensors. Hence, the robot has to get an environment map. There are various methods of robot localization in the literature. Markov Localization is a global technique for estimating the robot position in its environment. This method utilizes a probabilistic structure to maintain the position probabilities on a set of all possible robot positions. This technique is an special case of state probabilistic estimative applied to mobile robot localization [Fox et al., 1999].

In our system, it has been used the Monte Carlo Localization [Thrun et al., 2000], which is an special case of Markov Localization [Fox et al., 1998]. Monte Carlo Localization is based on *particle filters*, which are approximated Bayes filters utilizing examples for obtaining a posteriori estimative of the robot's position. Recently, these filters have been applied successfully for robot localization.

The key idea of Markov localization is to compute a probability distribution over all possible locations in the environment. Let $l = (x, y, \theta)$ denote a location in the state space of the robot, where $x$ and $y$ are the robot's coordinates in a world-centered cartesian reference frame, and $\theta$ is the robot's orientation. The distribution $Bel(l)$ over all locations $l$ expresses the robot's subjective belief for being at position $l$. Initially, $Bel(l)$ reflects the initial state of knowledge: if the robot knows its initial position, $Bel(l)$ is centered on the correct location; if the robot does not know its initial location, $Bel(l)$ is uniformly distributed to reflect the global uncertainty of the robot. As the robot operates, $Bel(l)$ is incrementally refined.

Markov localization applies two different probabilistic models to update $Bel(l)$, an action model to incorporate movements of the robot int $Bel(l)$ and a perception model to update the belief upon sensory input.

In this method, the robot motion is modeled by the conditional probability $p(l \mid l', a)$ specifying the probability that a measured movement action $a$, when executed at $l'$, takes the robot to $l$ position. $Bel(l)$ is updated according to:

$$Bel(l) \longleftarrow \sum_{l'} p(l \mid l', a) \cdot Bel(l'). \tag{2}$$

The term $p(l \mid l', a)$ represents a model of robot's kinematics. In the implementation of the proposed system, it has been assumed that the errors of the odometry are normally distributed.

Let $s$ be a sensor reading and $p(s \mid l)$ the likelihood of perceiving $s$ given that the robot is at position $l$. Then, the sensor readings are integrated according to the Bayesian updating equation,$Bel(l)$, given by:

$$Bel(l) \longleftarrow \alpha p(s \mid l) Bel(l), \tag{3}$$

here $\alpha$ is a normalizer ensuring that $Bel(l)$ sums up to 1 over all $l$.

Both updates steps are only applicable if the problem is *Markovian*, that is, the past sensor readings are conditionally independent of future readings given the location of the robot. The Markov assumption thus assumes that the world is static. While in practice, the approach has been applied even in environments that contained people.

More details about the localization module can be found in [Bianchi, 2003].

### 2.1.3 Path Planning

Mobile robots have to know how to reach a determined local using only the information about the departure and arrived points. For this, it is necessary that robot knows its the position in the environment where it is and what actions must be taken from its actual position. As it has been presented in the previous section, the localization robot is obtained by the localization and mapping modules. After to get the robot's position, the robot have to decide what is the more adequate action to be taken in order it could move in the environment, avoiding collision and aiming to reach the arrived point.

In the system proposed here a path planning module has been created based on the topological map mentioned bellow. Then, the route to be followed by the robot is that obtained by using the Dijkstra algorithm[Johnsonbaugh, 1997].

### 2.2 Image grabber Module

For the user to see the robot performing its tasks, it is necessary a module that captures the images. The image grabber module is responsible for capturing the images from the wireless camera attached to the robot. A video capture card installed at the server processes the images. After captured, the images are sent to the interface, via web, where they are shown to the user. This is a continuous process that allows the user to visualize the environment as the robot moves.

This module uses a wireless camera and the Pinnacle Studio DC 10 plus image capture card. The images are sent directly from the camera to its receptor, which is connected to the capture card.

### 2.3 Web Interface

The Web interface shows an environment map, a picture of this environment and a position of robot in this map. By the interface, the user can send tasks to be realized by the robot. The user needs only to press the mouse in some place of the map for sending the robot to a new requisition in the map. For the robot goes ahead the user needs to press in the button "send robot". So, this requisition is sent to the navigation system, which by its turn, verifies the best trajectory (route) for the robot to take and send an action to be realized by the robot through the modem radio. While the robot moves itself in the environment, its position in the map and a picture of environment are updated in the interface.

It is important to note that robot doesn't go to the coordinates corresponding to the position presented by the user. The robot uses the topological map for finding a map point that is closer to that position presented by the user.

There is also a module that is responsible by the communication between the interface and the navigation module presented in section 2.1 This communication is established by the sockets and allows that the requisitions sent by the users be transmitted, via web, to the system that responsible by the autonomous control of robot.

A interface was developed in JAVA language. This choice is based on the fact this language is widely utilized in applications that run in the Internet and also due to its portability that turns it independent of computer to be used.

Instead the Internet be a communication way considered instable, the data traffic needed for controlling a robot is very small. The client sends only a requisition of task from the user and the server sends only the coordinates $(x, y, \theta)$ to the interface, where $(x, y)$ represents the position of the robot in plane and $\theta$ is the rotate angle.

In Figure 3, the interface developed for this system is presented.

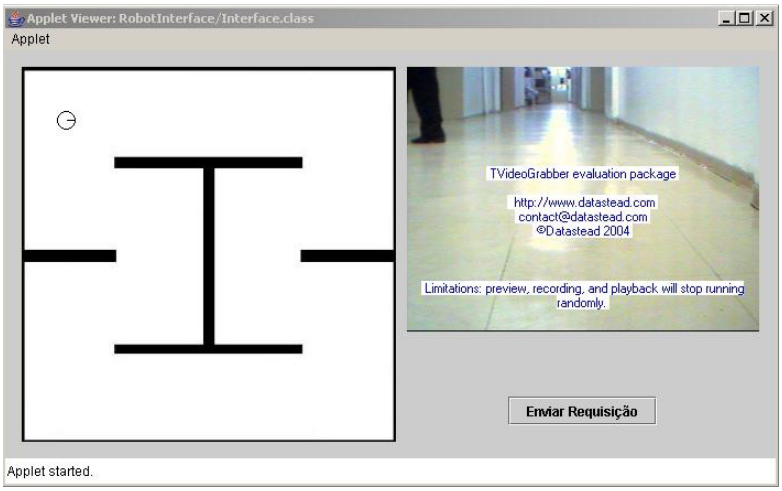



Figure 3: Interface Web for controlling the robot

### 2.4 Robot Hardware

All the experiments performed have been run in a robot Pionner 1 da ActivMedia Robotics[5], shown in Figure 4, existing in LABIC[6] of ICMC-USP[7].

The features of this robot is described below:

- **Motors and position enconders:** The robot is powered by two reversible DC motors. It can move with an approx-

[5]http://www.activemedia.com
[6]Laboratory of Computational Intelligence
[7]Institute of Mathematical and Computation Sciences at University of São Paulo

imate minimum speed of 60 cm per second. Its dead-reckoning capabilities vary a lot with the surface the robot is on. The error distance is about 1 cm per meter; the error in rotation is up 8 degrees per revolution.

- **Sensors:** the robot is equipped with seven ultrasonic proximity sensors, one in the each side and five forward facing. Moreover, a color CCD camera is mounted in a fixed position in top of the robot's console.
- **Gripper:** The robot is also equipped with a 2-axis, 1 degree-of-freedom gripper. The gripper operates between two states: down/open, and up/closed. IR-based 'break beams' embedded at the front and back on each paddle can sense objects between and within the gripper's grasp.
- **Radio-links:** The Pioneer's control software runs off-board, with which the robot communicates through a 9600 baud radio link. The remote software receives status updates from the robot and its sonar sensors at a frequency of 10 Hz. The maximum command rate for charging the motion direction is only 2 Hz(or less) due to limitations of the on-board controller. In designing the software, special attention is paid to the fact that the radio link is unreliable.
- **Sonar sensing:** The Pionner sonars exhibit the typical characteristics of sonar sensors; they seldom measure the distance of the nearest object within their main cone. Instead, they often return values that are significantly smaller or larger than the 'correct' proximity. Sonar sensors are not particularly noisy; they just do not measure proximity. Instead, they measure the time elapsed between emitting and receiving a focused sound impulse. For smooth objects, chances to receive a sonar echo depend on the angle between the main sonar cone and the reflecting object. Sound waves that hit a wall frontally are very likely to be reflected back in the direction of the sensor, whereas sound that hits a wall in a sleep angle is likely to be reflected away in a direction where it cannot be detected. The latter effect is usually referred to as total reflection. As a result, only some of the sonar readings reflect proximity, whereas others do not. In addition sonar measurements can also be corrupted when the motors draw too much power. The power consumption can be deduced from the robot status report.
- **Proximity Laser Scanner:** The sensor is capable of scan the environment with the angle of 180 degrees, angle resolution of 0,5 degrees, maximum distance of 50 m and error up to 131 mm in distances $\geq$ 4 m.


Figure 4: Robot Pionner 1 equipped with SICK PLS laser sensor[Bianchi, 2003]

## 3. Experiments

In this section, are presented results of the several tests. performed. The tests have been performed for validating the system functioning and the interaction between interface and navigation modules.

For testing the navigation system, it was developed a system for simulating a client and generate several random requisitions. The system remained in operation by several hours, without interruption, and executing all of the received requisitions. During this time, the navigation system executed approximately 10.000 requisitions, not occurring any error of localization. As the trajectory module belongs to the navigation system, it has been also validate.

After validate the navigation system, a simulation has been developed, for validating the communication between the interface and the autonomous navigation system,

In Figure 5, it is shown a test, in which two users present their requisitions to the system through the interface. The user $\#1$ presents a requisiton to the system and after that the user $\#2$ present another. The requisitions are set in a queue. The navigation module sent to the robot, the requisition presented by the user $\#1$ by the modem radio and the robot executes this requisition.

In Figure 6, it is shown the path planning (trajectory) done by the robot corresponding to the requisition presented by user $\#1$. After conclude this, the robot begins executing the requisition presented by user $\#2$. In Figure 7, it is shown the path done by the robot corresponding to the requisition presented by user $\#2$.

During the time that the robot executed both requisitions, the interface kept showing the robot in movement, updating its positions in the map all of the time.

In a general manner, no problem occurred in the tests, the robot executed all the presented requisitions, always choosing the small path to walk.

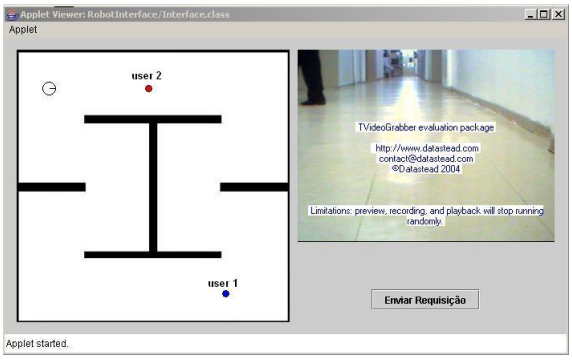
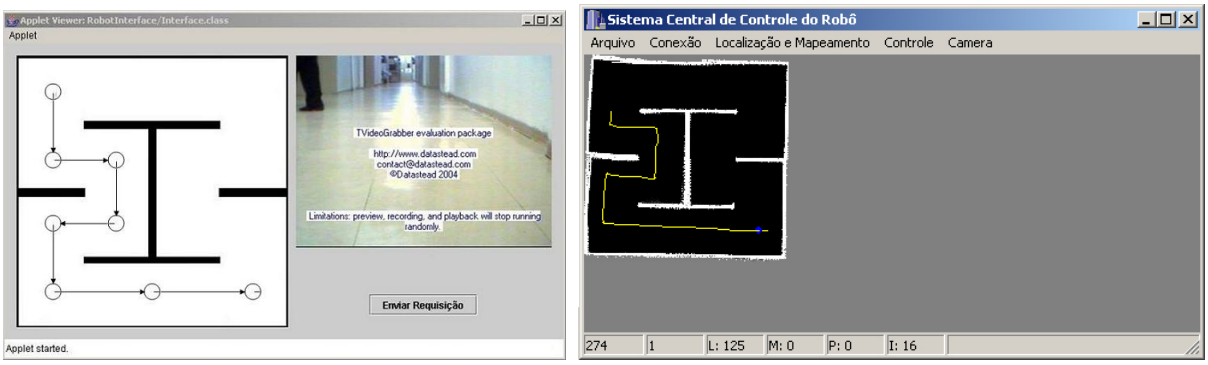



Figure 5: A sample map with 2 users requesting tasks to robot




(a) Web Interface (b) Autonomous navigation system

Figure 6: Path planning for the user #1 requisition


(a) Web Interface (b) Autonomous navigation system

Figure 7: Path planning for the user #2 requisition

## 4. Conclusion

In this work, it was implemented a system for controlling a small mobile robot Pioneer 1, remotely, via Web. This system works very well in all of experiments realized.

Some improvements need to be done to turn this system completely autonomous and working on the Internet. A topologic map of environment, used to facilitate the trajectory control, could be obtained in a automatic way by using Voronoi Diagrams [Thrun and Büecken, 1996]. Finally, the implementation of techniques for environment automatic exploration such as harmonic functions [Prestes et al., 2002] will be also investigated.

## 5. References

Bianchi, R. E. (2003). Sistema de navegação de robôs móveis autônomos para o transporte de documentos. In *Dissertação de Mestrado*.

Buhmann, J. M., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F. E., Strikos, J., and Thrun, S. (1995). The mobile robot RHINO. *AI Magazine*, 16(2):31–38.

Carvalho, A. d., Delbem, A.C.B.and Romero, R., Simões, E., and Telles, G. (2004). Computação bioinspirada. In *Capítulo de livro da XXIII JAI - Jornada de Atualização em Informática - JAI'2004, Anais do XXIV Congresso da SBC*.

Elfes, A. (1989). *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University.

Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207.

Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Grange, S., Fong, T. W., and Baur, C. (2000). Effective vehicle teleoperation on the world wide web. In *International Conference on Robotics and Automation*. IEEE.

Johnsonbaugh, R. (1997). *Discrete Mathematics*. Prentice Hall.

Kortenkamp, D. and Weymouth, T. (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial intelligence*, pages 979–984, Menlo Park. AAAI, AAAI Press/MIT Press.

Matarić, M. J. (1994). Interaction and intelligent behavior. Technical Report AI-TR-1495, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA.

Prestes, E., Engel, P. M., Trevisan, M., and Idiart, M. A. (2002). Exploration method using harmonic functions. *Robotics and Autonomous Systems*, 40(1):25–42.

Simmons, R., Fernandez, J., Goodwin, R., Koenig, S., and O'Sullivan, J. (1999). Xavier: An autonomous mobile robot on the web.

Thrun, S. and Büecken, A. (1996). Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2000). Robust Monte Carlo localization for mobile robots. Technical report, Carnegie Mellon University.