POSE CONTROL IN MOBILE ROBOTS

Clovis Peruchi Scotti*, Eng. Mathias José Kreutz Erdtmann*,
Prof. Dr.-Ing. Marcelo Ricardo Stemmer*

*Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis, Santa Catarina, Brasil

Emails: scotti@das.ufsc.br, erdtmann@das.ufsc.br, marcelo@das.ufsc.br

Resumo— Para algumas configurações especiais de robôs móveis e do espaço de trabalho, pode ser relativamente fácil controlar a posição e orientação do robô, mas ao confrontar-se com estruturas complexas o problema torna-se de difícil resolução geral. Este artigo propõe técnicas para o processo de geração de controle de posição, ilustrando os problemas que podem ser relevantes durante a implementação física de robôs móveis e as soluções típicas para tais problemas. Uma abordagem simples é utilizada como um incentivo ao projeto modularizado de robôs móveis, incluindo capacidades de controle de posição e, conseqüentemente, medição de posição.

Palavras-chave— robôs móveis, navegação em robôs, medição de posição.

Abstract— For some special configurations of the mobile robot and its workspace, it is fairly easy to control the mobile robot pose, but when confronting to complex structures the problem is much harder to solve in a general way. This paper proposes some guidelines in the process of pose control, showing the problems that may occur while implementing real mobile robots and the classic solutions for such problems. A minimalistic approach is used as a way to get good modularized design of mobile robots, including pose control capabilities and thus pose measurement.

Keywords— mobile robot, robot navigation, pose measurement.

1 Introduction

The problem of pose control in mobile robots includes modeling, physical and computational aspects. All these aspects make such problem interesting and challenging, since the best balance between all of them must be found to implement a good mobile robot.

Understanding how the process of robot displacement works makes possible to implement good strategies that are coherent with the mobot structure and still fully functional.

This paper describes some basic definitions and assumptions over the environment and physical attributes of mobile robots, showing some of the results of these assumptions with a sample mobile robot structure.

The minimalist approach is used in order to keep the implementation process at the higher rank, reducing the overhead of the mobile robot design process.

2 Definitions

This section intends to define the subjects of the study in the simplest way possible, including only the necessary concepts to be able to get some good results. These concepts must be simple in order to simplify the implementation effort by classifying and modularizing the mobile robots components.

This way, the motors are defined, as well as the relation between they and the mobile robot:

Definition 1 (Motor set) *A motor set over p is defined as a transformation M_u over $\mathbb{R}^n$, parameterized by the input vector u in the Input Set U, such that $p(t) : \mathbb{R} \to W \subseteq \mathbb{R}^n$ is the solution of the (differential) equation:*

$$\dot{p} = M_u(p) = M(p, u), \quad M : \mathbb{R}^n \times U \to \mathbb{R}^n$$

Definition 2 (Mobile Robot (Mobot)) *A Mobot is a rigid body in workspace $W \subseteq \mathbb{R}^n$ with pose $p_r \in W$ and a set of motors over p_r.*

A simple model of a mobot is thus presented here, under the name of AlphaMobot. It models a 2-dimensional non-holomonic mobot which position and direction have to be controlled.

So let the AlphaMobot pose $p_r = (x, y, \theta)$ be defined as the position of the rigid body in a XY plane and its rotation in the z-axis related to the x-axis (Figure 1(a)). The inputs $u = (u_v, u_\omega)$ of the motor set are the tangential and the rotational speed of the mobot, respectively (Figure 1(b)). The tangential speed is defined as the speed of the mobot in the current direction θ, thus leading to:

$$\dot{p}_r = M(x, y, \theta, u_v, u_\omega) \Rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_v \cdot \cos\theta \\ u_v \cdot \sin\theta \\ u_\omega \end{bmatrix}$$

This suffices all the needs of Definition 2, using the whole $\mathbb{R}^3$ as workspace W and supposing that the Motors accept any range of input ($U = \mathbb{R}^2$).

The purpose of a mobot is to movement itself in the workspace. It uses its set of motors to do so, but there is still the need to define the motors

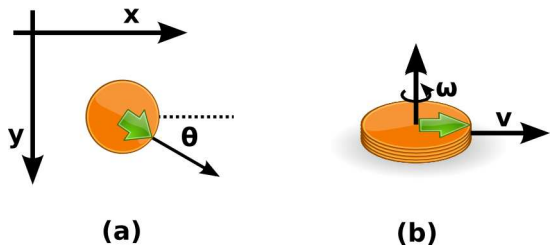

Figure 1: AlphaMobot: (a) pose and (b) motor set variables

input. If these inputs are human-controlled, the mobot will be something like a (remote) controlled vehicle. Although this can be interesting in some aspects, if the mobot can control the inputs by itself, then it will be possible to define places to go instead of motors input.

In order to deal with this input control, the Mobot Basic Problem is defined as follows:

Definition 3 (Mobot Basic Problem) *Given a mobot in W with initial position $p_r(0) = p_0$ and a desired final position $p_1 \in W$, a solution of the mobot basic problem is an input function u ($u : t \to U$) such that $p_r(t_f) = p_1$ with $t_f < \infty$.*

And so a basic mobot is a mobot that, from any given start pose, defines its motors input so that it could move itself to any desired pose.

Definition 4 (Basic Mobot) *A basic mobot in W is a mobot in which it is possible to solve the mobot basic problem for every p_0 and $p_1 \in W$.*

A basic mobot is a good start point for pose controlled mobots. It is still not very sophisticated, but the first step in building a mobot will be accomplished by solving the basic mobot problem.

3 Results

Using the definitions of the previous section and some assumptions over the systems proprieties it can be easier to prove that some specific kind of mobot solves the basic problem. In particular, this section proves that the AlphaMobot is a basic mobot and gives some clues in how to test some good characteristics in mobile robots, such as revertible paths.

If the mobot can move between two poses forward and backward simply by reversing its motors inputs, it is easier to control its pose, since it is always possible to redo and undo movements. This fact is easy to be verified using the motors set relation, as illustrated in the following theorems.

Theorem 1 *For a given mobot, if M_u is an odd function related to the inputs ($M(p,u) = -M(p,-u)$), and u is the solution for the mobot basic problem for $p_r(0) = p_0$ and $p_r(t_f) = p_1$, then $\tilde{u} = -u$ is the input function for the mobot basic problem for $p_r(0) = p_1$ and $p_r(t_f) = p_0$.*

Proof: u is the solution for $p_r(0) = p_0, p_r(t_f) = p_1$ and $\dot{p} = M(p,u) \Rightarrow p_1 = p_0 + \int_0^{t_f} M(p,u)dt \Rightarrow p_0 = p_1 - \int_0^{t_f} M(p,u)dt = p_1 + \int_0^{t_f} M(p,-u)dt \Rightarrow -u$ is the solution for $p_r(0) = p_1, p_r(t_f) = p_0$ □

Theorem 2 *For a given mobot, if M_u is an odd function related to the inputs ($M(p,u) = -M(p,-u)$), the mobot is basic if and only if the mobot is capable of solve the basic problem for at least one final position $\tilde{p}_1$, $\forall p_0 \in W$.*

Proof: ($\Rightarrow$) If the mobot is basic, it solves the basic problem for every p_0, p_1, thus solving in particular for all p_0 and (at least) one p_1.

($\Leftarrow$) Given $p_0, p_1 \in W$, let $\tilde{u}_0$ be the input function so that $p_r(0) = p_0$ and $p_r(t_{f1}) = \tilde{p}_1$ and $\tilde{u}_1$ the input function for $p_r(0) = p_1$ and $p_r(t_{f2}) = \tilde{p}_1$. Defining the input u as:

$$u(t) = \begin{cases} \tilde{u}_0(t), & t \le t_{f1} \\ -\tilde{u}_1(t - t_{f1}), & t > t_{f1} \end{cases}$$

leads $p_0 \to \tilde{p}_1 \to p_1$ (theorem 1), so that $p(t_{f1} + t_{f2}) = p_1$ □

The path provided by theorem 2 is usually non-optimal, since it assumes that the path between any two points pass through one special defined point. This pictures an mobot that every time it is commanded to get to another pose goes back to home ($\tilde{p}_1$) to be able to define the next path. Anyway, the idea that it is always possible to come back home is very comfortable and secure.

No assumption over the workspace is made yet, although it sounds easier to control the pose in "open" spaces, such as spaces without obstacles and no need for avoidance. For example, the AlphaMobot is originally placed in a infinite workspace clean of obstacles.

By the other hand, the workspace can be disconnected (with two separable parts). In this case there is a clear impossibility to get from a point in one part to another point in the other, turning away the interest of the problem. So for now on it is assumed that the workspaces are at least connected.

3.1 Convex workspaces

Convex workspaces are very special workspaces in which every two points can be connected by a straight line and every point in the line is in the workspace (Figure 2(a)). Connected workspaces in general can be non-convex, but since they are connected, there is always a path linking two points.

Definition 5 (Convex space) *A space W is convex if, for all $p_0, p_1 \in W$, $(t \cdot p_0 + (1-t) \cdot p_1) = p \in W, 0 \leq t \leq 1$.*

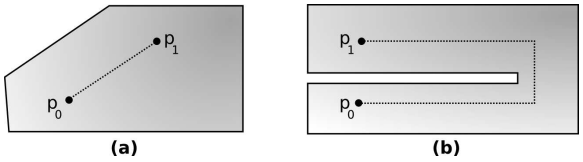

Figure 2: Different workspaces: (a) convex and (b) connected but non-convex workspace

It is easy to see that $\mathbb{R}^n$ is a convex space, giving some clues in how to solve the basic problem for the AlphaMobot and leading to the following result:

Theorem 3 *The sample mobot defined in section 2 as AlphaMobot is a basic mobot.*

Proof: For a given $p_0 = (x_0, y_0, \theta_0)$ and $\tilde{p}_1 = (0, 0, 0)$, define $u(t) = (v(t), w(t))$ as:

$$w(t) = \begin{cases} \pi - \arctan(y_0/x_0) - \theta_0, & 0 \leq t \leq 1 \\ 0, & 1 < t \leq 2 \\ \arctan(y_0/x_0) - \pi, & 2 < t \leq 3 \end{cases}$$

and

$$v(t) = \begin{cases} 0, & 0 \leq t \leq 1 \\ \sqrt{x_0^2 + y_0^2}, & 1 < t \leq 2 \\ 0, & 2 < t \leq 3 \end{cases}$$

If the equation given by motor set definition is integrated, it is possible to prove that $p_r(3) = \tilde{p}_1$, for any p_0. Moreover, the resultant path p_r is an straight line between p_0 and $\tilde{p}_1$. As $W = \mathbb{R}^3$ is convex, the path is valid, since every point in the path is in W. Thus, by theorem 2, the mobot is basic. □

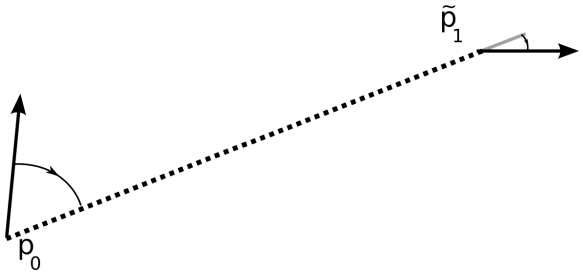

Figure 3: Path p_r produced by the input defined in the proof of theorem 3

It is possible that for some purposes the simple connection between two points provided by the basic mobot are not good enough, requiring the resulting path to have other additional proprieties, such as the smoothness.

Usually in mechanical systems functions of class C^3 are the minimum requirement for smooth movements. This provides that the position, speed and acceleration are all continuous. It is a much harder problem to find inputs that suffices these requirements than the basic problem.

Definition 6 (Mobot C^k problem) *Given a mobot in W with initial position $p_r(0) = p_0$ and a desired final position $p_1 \in W$, a solution of the mobot C^k problem is an input function u of class C^k $(u : t \to U)$ such that $p_r(t_f) = p_1$ with $t_f < \infty$.*

In particular for the AlphaMobot, there is yet a simple solution to the C^k problem, using splines, since splines paths are exactly defined by the initial and final points positions and tangent vectors ("speeds").

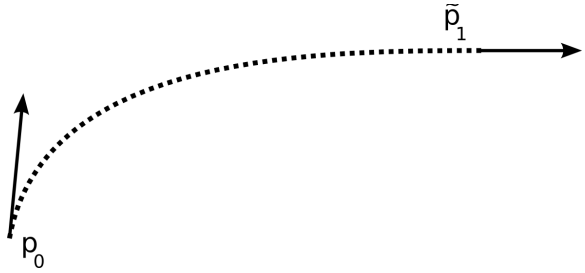

Figure 4: Path constructed using splines

The tangential speed input can be easily calculated from the spline formulation, as well as the rotational speed (defined usually as torsion using spline notation) and the splines definition assures that the start and end points suffices the requirements.

3.2 Non-convex workspaces

The problem for non convex workspaces can be hard to prove in such a general way, but since the workspace must be at least connected, there is always some path between two points. This way, the real problem is to define motors inputs in order to create this path or, instead, finding such path and follow it by using a step by step approach over small convex areas.

A general solution to this problem is to have a map of convex areas and its connections, so that it is possible to get to any pose within the same area as usual and, for other areas, to switch between two areas using special locations in the boundary named as **gates**.

Graphs can be used to model the neighborhood of areas, where the vertexes are the areas and the edges are the gates. So it is possible to use a wide set of graph tools to solve navigation problems, like finding one path between two areas or

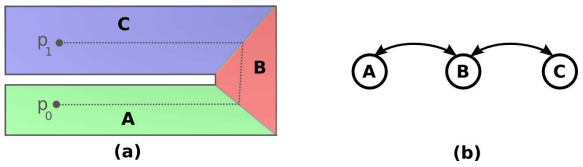

Figure 5: Separating a non convex workspace in three convex workspaces (a) and its neighborhood graph (b)

(as easy as that) the shortest path between areas. (Ferguson and Stentz, 2006; Carsten et al., 2006)

Moreover, if the workspace contais obstacles or regions to avoid, a map will be needed for the mobot to be able to plan the path. This map can be pre-loaded and static, but this is a simplistic approach, since it assumes that the real environment will not change significantly in the future, which is in general false.

Another option is the self constructed map, constructed by the mobot during some exploration process, stored for future exploitation. Although it is a much more complex approach, it works pretty well, mainly because when humans construct a map for the mobot they assume a structure that is obvious to them but not necessary to the mobot, which have different sensorial tools.

4 Physical Implications in Pose Control

In order to solve the basic problem, the mobot is required to define its motors input using some variables that were assumed known. But, in a real implementation, these variables must be measured in order to be known. In fact, a mobot needs at least to be able to know the current pose to define the inputs as a function of it.

Measuring or estimating mobots pose (as in AlphaMobots $p_r = (x, y, \theta)$) is generally problematic. There are many usual solutions to solve this problem, some of which had become very popular among robots designers (Siegwart and Nourbakhsh, 2004). Each one holds its advantages and disadvantages, therefore combinations between them are common. Also, considering the fact that position is always measured in relation to some known point, an origin (or "home pose") $p_r = p_0$ must be defined. The most common two of the approaches are described in this paper.

4.1 Dead Reckoning

With Dead Reckoning (also called path integration), the mobot estimates its position through time by using a initial position and applying all the signals sent to its own motion devices to a mathematical, simplified, model given by the motor set equation (Borenstein and Feng, 1994; Siegwart and Nourbakhsh, 2004). For AlphaMobot, this can be described as:

$$p(t_f) = \begin{bmatrix} x(t_f) \\ y(t_f) \\ \theta(t_f) \end{bmatrix} = p_0 + \int_0^{t_f} \begin{bmatrix} u_v(t) \cdot \cos\theta(t) \\ u_v(t) \cdot \sin\theta(t) \\ u_\omega(t) \end{bmatrix} dt$$

This model will then produce estimatives of the current mobot position in relation to its position at $(t = 0)$ which is known as the starting point or origin.

This approach is simple because no information from the environment is needed. The main disadvantage of this technique relies on model imperfections, since the smallest error in the model will be integrated through time, leading to prohibitive position errors.

The model imperfections come from the assumptions made for the mobot motors dynamics, such as assuming that the output torque and its resulting velocity is constant over time for a given input, temperature and load. This assumption is generally far from reality. Also, tires static friction coefficients with the ground are usually big but cannot completely eliminate skidding, which model is not trivial and thus is generally not considered.

There are two classes of position errors that emerge from dead reckoning, translational and rotational errors. Translational errors are produced by imperfections in the model of $u_v(t)$ and thus would generate the error

$$\begin{bmatrix} \Delta x(t_f) \\ \Delta y(t_f) \\ \theta(t_f) \end{bmatrix} = \int_0^{t_f} \begin{bmatrix} d(u_v(t)) \cdot \cos\theta(t) \\ d(u_v(t)) \cdot \sin\theta(t) \\ u_\omega(t) \end{bmatrix} dt$$

Given that $\Delta x(t_f)$ and $\Delta y(t_f)$ are produced by the same $d(u_v(t))$, they constitute a vector with direction θ_e close, if not equal, to the desired motion vector component θ. This error is therefore easy to overcome.

Rotational errors, in the other hand, are generated by imperfections in the model for $\theta(t_f)$ producing the error

$$\begin{bmatrix} \Delta x(t_f) \\ \Delta y(t_f) \\ \Delta\theta(t_f) \end{bmatrix} = \int_0^{t_f} \begin{bmatrix} u_v(t) \cdot \cos\Delta\theta(t) \\ u_v(t) \cdot \sin\Delta\theta(t) \\ d(u_\omega(t)) \end{bmatrix} dt$$

In this situation, $\Delta x(t_f)$ and $\Delta y(t_f)$ won't form a vector with direction θ_e close to θ, thus generating an ever growing position error(Borenstein and Feng, 1994).

In order to overcome this errors, the use of wheel encoders are very common. Encoders on wheels will act as an internal feedback of both $u_v(t)$ and, considering the difference of two encoder readings on the wheels of a differential drive system, $u_\omega(t)$. Although encoders may suggest something like a "live reckoning", the encoders are

internal to the robot structure and tires will never be perfect (they will skid, bend, deform), so rotational and translational errors will still be present.

4.2 Landmark-based feedback

If fixed features (i.e., landmarks) of the environment are trackable, landmark-based pose measurement is an option. It is achieved when the mobot can measure some of its pose components related to a landmark that can be a wall, light sources, ultrasonic beacon systems readings (Mata et al., 2002; DeSouza and Kak, 2002; de Lima Ottoni and Lages, 2003).

Using the relative measured distance between the mobot and a fixed landmark keeps errors lower because even though distance measurement can carry errors, they are not integrated in time, in fact they can be even attenuated using good sampling procedures.

Consider the simple problem of a mobot that aims to cross a room following a straight line and thus reaching a object with its $\theta = \angle\pi/2$, in relation to the objects angle (mainly for large objects as walls, for example). To reach its goal, the mobot only needs to set its motion vector as close as possible to the gradient of the scalar field formed by $D(p_m)$ of each point of the room, defined as:

$$D(p_m) = D(x_m, y_m, \theta_m) = d_m$$

such that d_m is the Euclidean distance between the mobot and the landmark. If the landmark is a straight wall, the mobot can even estimate its θ.

Splitting (simplifying) this task in two is a very effective way to solve this problem. The path can be planned as was done in theorem 3 for AlphaMobot, so that in the first step the mobot sets its θ pointing to the object, what can be accomplished by applying a constant input to its motion system:

$$u = (u_v, u_\omega) = (0, k_s)$$

, being k_s a small enough rotational speed that allows multiple readings of d_m for a wide range of θ_m, and then taking a θ_o that suffices

$$\begin{cases} \frac{\partial D(p_m)}{\partial \theta_m} = 0 \\ \\ \frac{\partial^2 D(p_m)}{\partial \theta_m^2} > 0 \end{cases}$$

θ_o is the optimal travel direction to reach the mobot goal, since it is the direction that provides the minimum distance to the goal. The second step consists in driving forward

$$u = (u_v, u_\omega) = (v_d, 0)$$

until $D(p_m)$ goes zero. If the drive distance is big, it might be necessary to repeat the first step many times to correct forward driving imperfections.

Landmark-based pose measurement and control is usually the best approach given the fact that errors do not grow over time. Mobot imperfections will be overcomed by the absolute landmark position measurement.

To locate and measure landmarks, the most common approaches are laser, ultrasonic or infrared rangefinders, machine vision systems, stereoscopic machine vision systems, ultrasonic beacon interpretteurs, contact or light sensors (Erdtmann et al., 2005). Some landmark-based systems even work with moving landmarks that have known motion vector through time.

There are some other unusual, but not less useful, types of landmarks. Earth's magnetic field is not exactly a "landmark" but act as one as long as it provides the mobot equipped with a Electronic Compass with absolute θ measurement. Also, the floor can be used as a "continuous landmark", using an Optoelectronic Sensor coupled with a LED, both pointing to the ground, as seen in any commercial optical mouse, can produce external feedback on the motion vector of the mobot. Global Positioning System (GPS) is also a landmark-based pose measurement system.

This approach disadvantage lies on the facts that sometimes it is really problematic and not trivial to measure the landmark position and that sometimes reliable landmarks are not available.

4.3 Hybrid Approaches

As suggested before, many hybrid solutions to pose measurement, and therefore control, are possible.

Providing a mobot with an electronic compass, it is possible to correct rotational errors from time to time. A very simple proportional control system can be created to control the mobot angle:

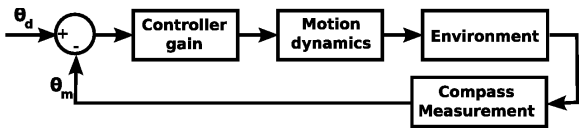

Figure 6: Angle control using an electronic compass

To a given measured angle θ_m and a desired angle θ_d, acting periodically on the motors proportionally to the error $(\theta_d - \theta_m)$ would lead rotational errors to null. Using this system, dead reckoning corrections can become very effective. The integral nature of the system eliminates constant rotational errors while the control is on.

This concept was simulated (as seen in figures 7 and 8) and the control system managed to eliminate rotational errors even on mobots that,

without this system, would suffer from rotational errors as big as 30 degrees on a relative small (10 cm - a third the size of the robot) forward drive.

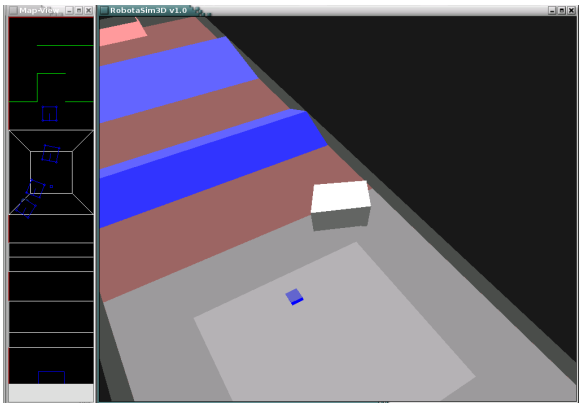

Figure 7: Mobot simulation without any angle control system

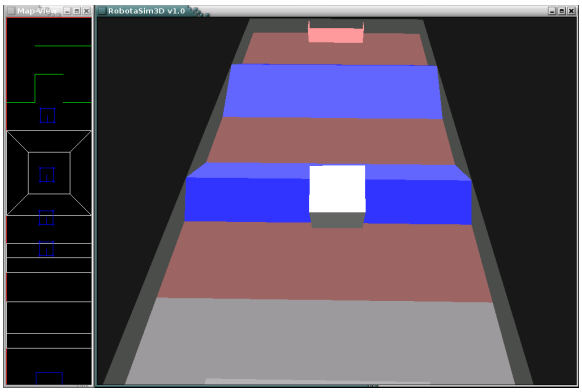

Figure 8: Mobot simulation using the proposed angle control system

This approach uses the magnetic field as continuous landmark, combined with accurate dead reckoning.

5 Conclusions

The process of definition and illustration of potential problems (and its solution) is a good project practice, since it unveils characteristics that were assumed not relevant in a quicker analysis.

Some problems, as the pose measurement, have usual solutions, but they can be pretty hard to implement in confined spaces and mobile environments, where almost nothing can be assumed static.

It is possible to get good results by using basic elements, such as electronic compass and mixed navigation approach with dead reckoning for small displacements and landmarks for area shifting (assuming that a pre-build static map can be used).

With the enhanced portability of modern computing, more general and robust approaches could be implemented by using "bulkier" sensorial systems, such as machine vision and stereo vision, allowing several landmarks to be detected and good pose estimatives, but the software design for object recognition is still a problem to be solved.

References

Borenstein, J. and Feng, L. (1994). *UMBmark - A Method for Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots*, University of Michigan, Michigan.

Carsten, J., Ferguson, D. and Stentz, A. (2006). 3d field d*: Improved path planning and replanning in three dimensions, *2006 IEEE/RSJ, International Conference on Intelligent Robots and Systems*, IEEE, Beijing, China.

de Lima Ottoni, G. and Lages, W. F. (2003). Navegação de robôs móveis em ambientes desconhecidos utilizando sonares de ultra-som, *Sba Controle & Automação* **14**(4).

DeSouza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(2).

Erdtmann, M. J. K., Silvano, C. M. and Stemmer, M. R. (2005). Modelo de visão computacional de baixo nível antropomórfica com aplicação em robótica móvel, *Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI*, Natal.

Ferguson, D. and Stentz, A. (2006). Anytime RRTs, *2006 IEEE/RSJ, International Conference on Intelligent Robots and Systems*, IEEE, Beijing, China.

Mata, M., Armingol, J., de la Escalera, A. and Salichs, M. (2002). Learning visual landmarks for mobile robot navigation, *15th World congress of the International Federation of Automatic Control*, Barcelona, Spain.

Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*, MIT Press.