**Noname manuscript No.**
(will be inserted by the editor)

# A Motion Strategy for Exploration Driven by an Automaton Activating Feedback-based Controllers

**Edgar Martinez · Guillermo Laguna · Rafael Murrieta-Cid · Hector M. Becerra · Rigoberto Lopez-Padilla · Steven M. LaValle**

Received: date / Accepted: date

**Abstract** This paper addresses the problem of exploring an unknown, planar, polygonal and simply connected environment. To explore the environment, the robot follows the environment boundary. In the first part of this paper, we propose a motion policy based on simple sensor feedback and a complete exploration strategy is represented as a Moore machine. The proposed motion policy is based on the paradigm of avoiding the state estimation; there is a direct mapping from observation to control. We present the theoretical conditions guaranteeing that the robot discovers the largest possible region of the environment. In the second part of the paper, we propose an automaton that filters spurious observations to activate feedback-based controllers. We propose a practical control scheme whose objective is to maintain a desired distance between the robot and the boundary of the environment. The approach is able to deal with imprecise robot's observations and controls, and to take into account variations in the robot's velocities. The control scheme switches controllers according to observations obtained from the robots sensor. Our control scheme aims to maintain the continuity of angular and linear velocities of the robot in spite of the switching between controllers. All the proposed techniques have been implemented and both simulations and experiments in a real robot are presented.

**Keywords** Exploration · Combinatorial Filters · Feedback Controllers · Nonholonomic Constraints

This work was partially funded by CONACYT projects 220796 and 264896.

E. Martinez
Centro de Investigación en Matemáticas, CIMAT, Guanajuato, México
E-mail: edgar.martinez@cimat.mx

G. Laguna
Iowa State University,
Ames Iowa, USA
E-mail: glaguna@iastate.edu

R. Murrieta-Cid
E-mail: murrieta@cimat.mx

H. M. Becerra
E-mail: hector.becerra@cimat.mx

R. Lopez-Padilla
CIATEC,
León, México
E-mail: rlopez@ciatec.mx

S.-M. LaValle
University of Illinois,
Urbana IL, USA
E-mail: lavalle@illinois.edu

## 1 Introduction

Our work is related to the problem of planning robot's paths that avoid collision with obstacles [1–3], and particularly with nonholonomic robots [4–6]. This paper is also related to the problem of exploring an unknown environment [7–10] to build a representation of it useful for other tasks, for instance object finding [11,12].

The task of exploring unknown planar environments has been treated in many previous works [13–15,11,16,17]; some of them use a simplified model where a mobile robot is considered as a point. From a theoretical point of view, this approach has allowed one to solve some problems of robot navigation; however, for more realistic tasks, this approach is not sufficient. Modeling the robot as a point ignores the robot's physical dimensions and that assumption may impact the real performance. A natural step forward, and more realistic, is to consider the robot as a nonzero size entity. A disc shape is the simplest one. The robot's size represents additional constraints in the configuration space. This raises the main conceptual difference between a point robot and a disc robot, which makes necessary the design of exploration strategies specific for a disc robot. Indeed, the concept of visibility is equal to the concept of reachability for a point robot. It means that if the robot can see certain place within the environment, that place is also reachable for the robot. However, this property is not necessarily true for a disc robot.

If the original map is totally known then to build the configuration space of a disc robot is easy, this map is expanded by the robot radius. But note that in the exploration problem addressed in this work, initially the map is unknown and the problem is to discover it with the robots sensors. Also note that the configuration space of the disc shaped robot is not observable, the robot cannot directly measure it with a sensor, not even for a local map. To plan collision free paths, the disc shaped robot must infer relevant information of the configuration space from the workspace.Thus, one main contribution of this paper is to determine what relevant information is needed from the workspace, and how to obtain it directly with the robot sensor to infer valid motions in the configuration space. In this paper, we also present a novel exploration strategy of an unknown, planar polygonal environment using a disc robot.

## 1.1 Related Work

Many works have addressed the problem of exploring an unknown environment to build a representation of it [18–21]. It is possible to classify those exploration strategies into two main types: (i) systematic exploration and (ii) strategies in which sensed information is taken into account to define the next sensing location. In systematic explorations (exploration type (i)), the robots follow a predefined motion pattern, for instance following walls, moving in concentric circles [22], and so forth. In non-systematic exploration (type (ii)), information taken by the sensor is frequently used to select an appropriate sensing location. Some exploration strategies of type (ii) use frontier-based exploration, originally proposed in [8]. In frontier-based exploration, the robot goes to the imaginary line that divides the known and unknown parts of the environment. In [18–20], the proposed exploration strategies lead the robot to locations in which maximal information gain is expected; a utility function is defined to maximize the new information that will be obtained in the next sensing location. Several works have proposed to generate random sensing locations for exploration (e.g., [23,24]). The work reported in [24] presents sensor-based exploration techniques. Given strong sensors and good odometry, standard SLAM approaches [25–27] provide a geometric map of the environment. In [28], a method is proposed for building a global geometric map without precise robot localization by registering scans collected by laser range finder. A different map building approach is the occupancy grid [17], which represents the environment as a 2D array, instead of using geometric primitives (e.g., line segments). Another type of environment's representations are the topological maps in the form of graphs [16,11,29]. The problem of exploring an unknown environment for searching of one or more recognizable targets is considered in [16]. That method assumes limited sensing capabilities of the robot and the environment is represented in the so-called boundary place graph, which records the set of landmarks.

A method for robot's navigation without the capacity of sensing orientation but sensing range discontinuities is presented in [11]. In that work, the Gap Navigation Tree (GNT) is proposed, which is a combinatorial structure that encodes information about range discontinuities (gaps) and the relation between them. This original GNT approach was designed for exploration and navigation of a point robot. A probabilistic model for the gaps in the GNT is presented in [15]. This improves robustness given that the model deals with noise in the sensor's measurements. The GNT was also extended to clouds of points models in [14]. A larger family of gap sensors is described in [30]. The GNT approach has been extended to a disc-shaped differential-drive robot placed into a simply connected polygonal region in [31]. The main result in that work is a navigation strategy that drives the robot to optimally navigate toward a landmark in the region. In [13], a wall following approach for exploration of a simply connected environment with a point robot has been proposed. A data structure called cut ordering is proposed in that work. Once the cut ordering representation is built, it is used to address a pursuit/evasion problem.

As it was mentioned above, the problem of finding collision free paths and the problem of exploring an unknown environment with a mobile robot have been two very active topics in the robotics community. But, to our knowledge, the previous works most closely related to our approach are the ones presented in [11,13] and [31,32]. A significant difference with respect to [13,11] is that in this work the robot is no longer a point. We model the robot as a disc shaped differential drive robot (nonholonomic system). In [31,32] a navigation strategy to reach a landmark has been presented, but in [31, 32] an exploration strategy to learn the GNT and encoding a landmark within it has not been developed. In this paper, we propose such exploration strategy.

The proposed exploration strategy is based on *wall following* and not on chasing gaps in contrast to [11].

Different control schemes have been proposed to achieve a wall following behavior, mainly by using a range sensor for feedback information, e.g. [33–35]. The work in [33] focuses on a robust detection and representation of walls. In [34] a switching wall-following control scheme based on odometry and distance information is presented. The control scheme is designed to avoid saturation of the angular robot velocity when dealing with discontinuous contours. The control scheme of [34] has been used for reactive obstacle avoidance by following the contour of the obstacles in [35]. The use of a distance sensor has also been extended for active sensing in [36], where the perception and action systems of a robot are dynamically coupled for reactive wall-following. Another type of sensor has been used for the wall-following task, for instance, a bio-inspired antennae [37], which is a passive tactile sensor.

### 1.2 Main contributions

One main contribution of this paper is a motion policy based on simple sensor feedback and a complete exploration strategy. We present the theoretical conditions guaranteeing that the robot discovers the largest possible region of the environment. The proposed strategy is compact, in such a way that it is represented as a Moore's finite state machine. Furthermore, the proposed exploration strategy does not requiere to localize the robot.

To explore the environment, the robot follows the environment boundary. We propose a practical hybrid control scheme, whose objective is to maintain a desired distance between the robot and the boundary of the environment. This practical control scheme allows the robot's commands to be imperfect, and to deal with the robot dynamics (i.e. velocities variations). Besides, our control scheme aims to maintain the continuity of angular and linear velocities of the robot in spite of the switching between controllers. The main originality of the proposed approach with respect to previous work on wall following is that a sensor observation is directly related to a given controller, and in this approach, an automaton constraints the possible states transitions filtering spurious observation due to noisy sensor readings.

A preliminary version of a portion of this work appeared in [38]. The main distinguishing features of our current work compared with our previous research in [38] are: 1) We propose an automaton that filters spurious observations to activate feedback-based controllers. 2) We propose a practical hybrid control scheme, whose objective is to maintain a desired distance between the robot and the boundary of the environment. 3) The work presented in [38] did not include any experiments on a real robot at all. In this new version, experiments on a real robotic set-up were included showing the practical viability of the approach.

The remainder of this paper is organized as follows: Section 2 presents the problem statement, Section 3 presents concepts defined in [11], which are used in this work. Sections 4 and 5 present the robot's model including sensing and motion capabilities. Section 6 presents the motion strategy that is modeled as a Moore machine and a feedback motion policy that maps observations to robot commands. In Section 7, we determine several sensor measurements. These measurements are used as feedback information in a hybrid control scheme, and they are transformed to binary observations that determine the transitions between states in a state machine. In Section 8, we propose another Moore's machine whose objective is to move the robot to a desired distance from the environment boundary. In this case, some observations activate a specific controller. In Section 9 feedback based controllers are proposed, these controlles are able to deal with noisy measurements and take into account robot's dynamics. Section 10 presents simulations and experiments in a real robot. Finally, Section 11 concludes the paper.

## 2 Problem Statement

The robot has the shape of a disc with radius $r$ moving in an unknown, planar, polygonal, and simply connected environment, which could be any compact set $E \subset \mathbb{R}^2$ for which the interior of $E$ is simply connected. The boundary $\partial E$ of $E$ is the image of a piecewise-analytic closed curve. However, it is assumed that the collision-free subset of the robot's configuration space $\mathcal{C}$ is simply connected or it might have several connected components. $\mathcal{C}$-space obstacle corresponds to that of a translating disc, that is, the extended boundary of $E$ which is due to the robot's radius[1]. A salient object (i.e. a landmark) is located in the environment. The robot is unable to localize itself in *a global reference frame.*

*The main objective is to explore an environment.* That is, while the robot moves the visibility region of the robot's sensor must cover the environment $E$ at least once, or in the worst case the largest possible region of $E$. Consequently, the robot will find the landmark or declare that an exploration strategy to find it does not exist.

[1] Note that this is the configuration space for a translating disc rather than for a rigid body because of rotational symmetry.

## 3 Preliminaries

We use a topological map called the Gap Navigation Tree (GNT) [11,31] to represent the environment. A depth discontinuity in the boundary of the environment is called a gap. The gaps are the borders between the known and unknown environment. The robot sensor (a laser range finder) is used to detect these discontinuities (gaps) from the current position of the robot (see Fig. 1). The GNT is an efficient data structure that dynamically changes according to some critical events in the gaps until the whole environment has been discovered.

The GNT can be constructed incrementally as the robot moves along a path $\tau$. Initially, the GNT consists of a root node that is connected to one leaf node for every gap in $G(\tau(0))$. Each time $t$ at which a change in $G(\tau(t))$ occurs corresponds to a critical event. This requires updating the GNT. There are four different kinds of critical events:

- A new gap $g$ *appears*: A node $g$ is added as a child of the root, while preserving the cyclic ordering from the gap sensor (see Fig. 1 a)). For a description of the gap sensor see Section 4.
- A gap $g$ *disappears*: The node $g$, which must be a leaf, is removed (see Fig. 1 b)).
- Gaps $g_1$ and $g_2$ *merge* into $g$: Nodes $g_1$ and $g_2$ become children of a new node, $g$, which is added as a child of the root and preserving the ordering of gaps (see Fig. 1 c)).
- Gap $g_1$ *splits* into $g_2$ and $g_3$: If $g_1$ is a leaf node, then $g_2$ and $g_3$ become new nodes; otherwise, they already exist as children of $g_1$. Both $g_2$ and $g_3$ are connected to the root, preserving the ordering of gaps and removing $g_1$ (see Fig. 1 d)).

If any leaf vertex has the potential to split, then the GNT is incomplete because it could expand, some gaps split and other gaps simply disappear. The gaps that disappear are called primitive (their corresponding nodes in the GNT are also called primitive). If all the leaf nodes of the GNT are primitive, then the GNT is said to be complete. Indeed, the following Lemma in [11] guarantees the termination of the GNT's construction.

**Lemma 1** *([11]) The procedure of iteratively chasing non-primitive leaves terminates with a resulting complete GNT.*

For the proof please see [11].

In [11], chasing a gap means to move the robot's sensor until it touches the vertex that generates the gap observing the portion of the environment occluded by that vertex and hence making the corresponding gap disappear. The key observation to understand the termination condition of the exploration task given by the GNT is that, any time that a new gap appears in the GNT, the portion occluded from the robot sensor has been already sensed (it is a primitive gap). Therefore, the only gaps that contribute to the incompleteness of the GNT are ones that either appeared at the beginning of the exploration or were formed by a sequence of splits of these gaps. Hence by chasing those initial gaps the exploration will be finished when all those gaps have been explored making them become primitive gaps.

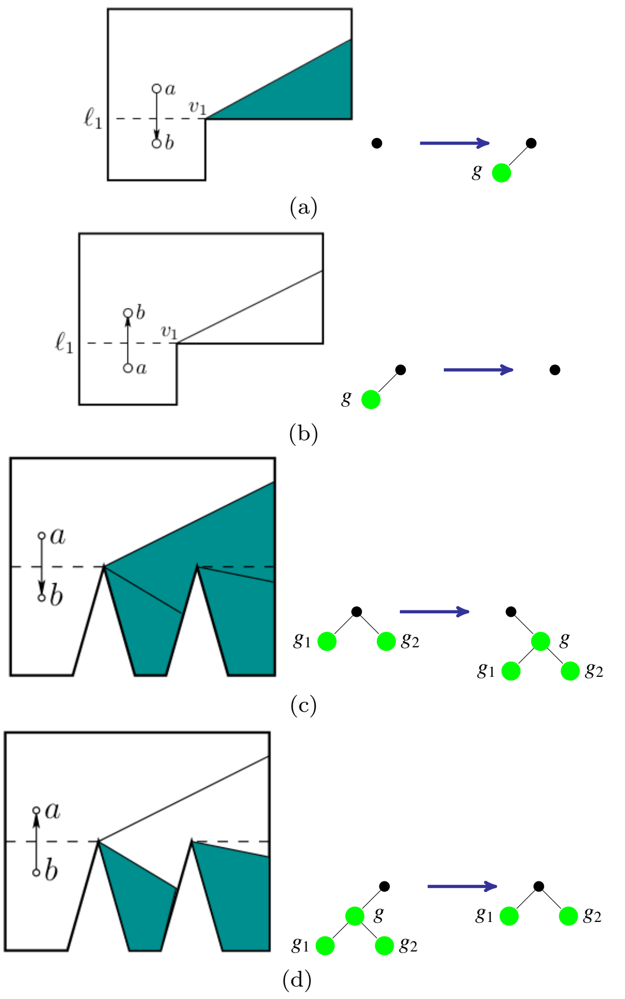


Fig. 1: Critical events: a) Gap appears, b) Gap disappears, c) Gap merge, d) Gap splits.

In [11], the angles of the gaps are unknown due to the limited sensor's capabilities, but the sensor is able to maintain a cyclic angular order of them. If a gap first disappears and then it appears again, it will appear in the same angular other with respect to the others gaps (this has been proved in [11]). Let $G(x) = [g_1, ..., g_k]$ denote the sequence of gaps as they appear in the gap sensor, when it is placed at $x \in E$, if $x$ lies in the interior of $E$ there is a cyclic order such that statements as $[g_1, ..., g_k] = [g_2, ..., g_k, g_1]$ can be made (see Figs. 2 a) and b)). If $x$ lies in $\partial E$ then part of the sensor's view is obstructed by the boundary, and a linear ordering of gaps is obtained (see Figs. 2 c) and d)).

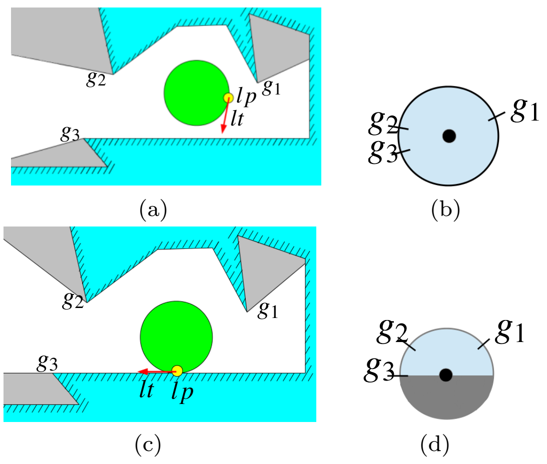


Fig. 2: a) Gap sensor at $x \in E$, b) Cyclic order, c) Gap sensor at $x \in \partial E$, d) Linear order.

## 4 Sensing Model

### 4.1 Robot's sensors and landmark

The differential drive robot has a defined forward heading. The extremal left and right side robot's points are respectively called $lp$ and $rp$. The robot has an omnidirectional sensor, which is used to discover the environment. The sensor might be located at $rp$ or $lp$. The direction of the line tangent to the robot's boundary at $rp$ is called $rt$. The direction of the line tangent to the robot's boundary at $lp$ is called $lt$ (see Fig. 3). The omnidirectional sensor is also able to track the direction $lt$ or $rt$ depending whether the sensor is placed over $lp$ or $rp$.

The omnidirectional sensor is also able to detect and track discontinuities in depth information (gaps). Hence, over the omnidirectional sensor, it is possible to build a gap detector, further referred as the gap sensor. The gap sensor is also able to identify any of the four possible critical events related to the gaps: gap appears, disappears, merges and splits.

In summary the gap sensor is able to detect and order the gap directions, the direction $rt$ or $lt$ and a visibility obstruction if the sensor is in contact with $\partial E$. This behavior allows the sensor to detect events such as alignments between the directions $rt$ or $lt$ and any gap, or between one of the two directions $lt$ or $rt$ and the wall ($\partial E$) that is in contact with the omnidirectional sensor.

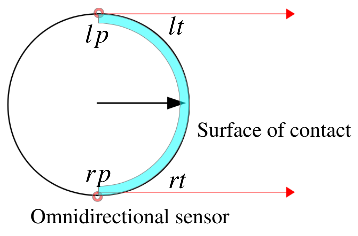


Fig. 3: Representation of the robot's sensors.

Let $\Lambda$ be a static disc-shaped landmark with the same radius as the robot lying on the interior of $E$. The landmark is said to be recognized if $\Lambda$ is visible at least partially from the location of the omnidirectional sensor.

Let assume that $\Lambda$ is painted in the ground and can be detected, hence it does not have volume and it does not produce distance discontinuities (gaps). This assumption is made for a further navigation task [31] in which once the environment has been explored the robot would have as goal to park on the landmark.

The assumption about that the landmark is a disc with the same radius as the robot allows one to establish that whenever the collision-free subset of the robot's configuration space $\mathcal{C}$ is simply connected then the landmark will be always reachable. Moreover, it also allows one to establish that in this case the whole landmark will be totally visible at some moment during the exploration.

Regarding the landmark encoding, if the whole landmark is visible from the current position then it is directly connected to the root of the GNT.

Let $rp_\Lambda$ be an extremal point on the landmark such that whenever $rt$ is aligned to $rp_\Lambda$ the body of the landmark is to the left of direction $rt$. There is an analogous definition for point $lp_\Lambda$.

The landmark $\Lambda$ can be encoded at most with two gaps. If a reflex vertex [2] occludes point $rp_\Lambda$ then $\Lambda$ is encoded with the gap generated by the vertex. Similarly, if a reflex vertex occludes point $lp_\Lambda$ then it is also encoded with the gap generated by the vertex. In particular, in the case when the robot and $\Lambda$ are located in different connected components of the collision-free

[2] A reflex vertex is a polygon vertex of an internal angle greater than $\pi$.

subset of the configuration space $\mathcal{C}$, then $\Lambda$ might be encoded at most with two gaps, one that is generated by a vertex that occludes point $rp_\Lambda$ and other that occludes point $lp_\Lambda$.

We assume that the robot can distinguish whether there exists contact on a single point or more than one. That is, the robot is able to detect whether its frontal periphery is in contact with an obstacle. The frontal periphery is called surface of contact (See Fig. 3). The sensor also distinguishes whether the point $rp$ or $lp$ is in contact with a wall. This information can be obtained with different sensor, e.g. a tactile bumber or a omnidirectional laser range finder. The particular case of both points $rp$ and $lp$ being simultaneously in contact with a wall is not considered, it only would happen in a narrow corridor of exactly the same width as the robot.

## 4.2 The observation vector

With the sensor capabilities defined above, it is possible to define an observation vector which includes all the possible observations that trigger a specific control.

Six binary sensor observations constitute the observation vector: (1: $lp$) the robot is touching $\partial E$ with point $lp$. (2: $rp$) the robot is touching $\partial E$ with point $rp$. (3: $sc$) the robot is touching $\partial E$ with a single point within the surface of contact (this point might be either $lp$, $rp$ or any other point within the surface of contact). (4: $bc$) the robot is touching $\partial E$ with two or more points within the surface of contact (one of them can be either $lp$ or $rp$). (5: $aligned$) in the case that point $rp$ is touching $\partial E$: if direction $rt$ is aligned with the polygonal edge that point $rp$ is touching, or, if point $rp$ is touching a reflex vertex and the direction $rt$ is aligned with the first polygonal edge measured in clockwise sense starting from direction $rt$ then $aligned = 1$. In the case that point $lp$ is touching $\partial E$: if direction $lt$ is aligned with the polygonal edge that point $lp$ is touching, or, if point $lp$ is touching a reflex vertex and the direction $rt$ is aligned with the first polygonal edge measured in counterclockwise sense starting from direction $lt$ then $aligned = 1$. (6: $o$) the omnidirectional sensor is located at point $lp$ (0) or the omnidirectional sensor is located at point $rp$ (1). Thus, the observation vector is $ye_i = \{lp, rp, sc, bc, aligned, o\}$

The set of all 64 observation vectors can be partitioned by letting $x$ denote any value to obtain:

$$\begin{aligned}
ye_1 &= (0,0,0,0,x,x)\\
ye_2 &= (0,1,1,0,1,1)\\
ye_3 &= (1,0,1,0,1,0)\\
ye_4 &= (x,x,0,1,x,1)\\
ye_5 &= (x,x,0,1,x,0)\\
ye_6 &= (0,1,1,0,0,1)\\
ye_7 &= (1,0,1,0,0,0)\\
ye_8 &= (0,0,1,0,x,0)\\
ye_9 &= (0,0,1,0,x,1)
\end{aligned}$$

The meaning of each observation vector is the following:

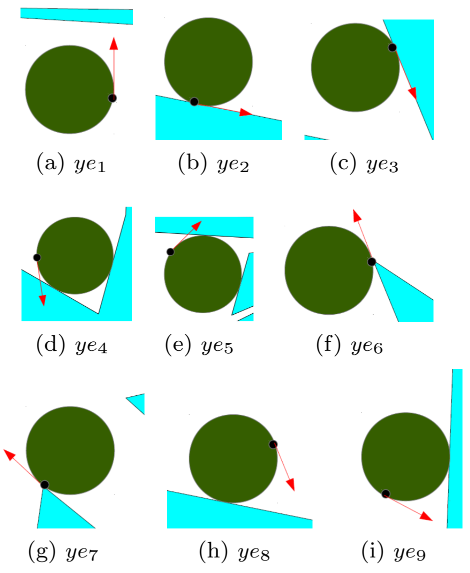

(a) $ye_1$ (b) $ye_2$ (c) $ye_3$ (d) $ye_4$ (e) $ye_5$ (f) $ye_6$ (g) $ye_7$ (h) $ye_8$ (i) $ye_9$

Fig. 4: Examples of Observation Vectors.

- $ye_1$ *No contact*: This observation might only happen at the beginning of the exploration if the robot lies completely in the interior of $E$, such that no contact with $\partial E$ is sensed (see Fig. 4a).
- $ye_2$ *Single contact* with $rp$: The omnidirectional sensor is positioned at $rp$, there is single contact detected at that point, and the direction $rt$ is aligned with the polygonal edge that point $rp$ is touching (see Fig. 4b).
- $ye_3$ *Single contact* with $lp$: This observation is analogous to Single contact with $rp$, it is the left symmetric case (see Fig. 4c).
- $ye_4$ *Multicontact, sensor at* $rp$: The omnidirectional sensor is located at point $rp$ and there is a multicontact detected ($rp$ might be a contact point), while the omnidirectional sensor is placed at $rp$. The robot's surface of contact is touching more than one point of $\partial E$, the contact might be with any combination of edges or reflex vertices of $E$ (see Fig. 4d).

- $ye_5$ *Multicontact, sensor at lp*: This observation is analogous to Multicontact $rp$, it is the left symmetric case (see Fig. 4e).
- $ye_6$ *Reflex vertex rp*: The omnidirectional sensor is located at point $rp$, there is single contact between point $rp$ and a reflex vertex of the polygonal environment, and the direction $rt$ is not aligned with the first polygonal edge, measured in clockwise sense starting from direction $rt$ (see Fig. 4f).
- $ye_7$ *Reflex vertex lp*: The omnidirectional sensor is located at point $lp$, there is single contact between point $lp$ and a reflex vertex of the polygonal environment, and the direction $lt$ is not aligned with the first polygonal edge, measured in counterclockwise sense starting from the reflex vertex (see Fig. 4g).
- $ye_8$ *No-single contact* at $lp$: The omnidirectional sensor is positioned over $lp$ and the robot is touching an edge or a reflex vertex of $\partial E$ with a single point different to $lp$ (see Fig. 4h).
- $ye_9$ *No-single contact* at $rp$: This observation is analogous to No-single contact at $lp$, it is the right symmetric case with the omnidirectional sensor positioned at $rp$ (see Fig. 4i).

## 5 Motion Model

The differential drive robot has two independent wheels, each one with its own motor. The robot is allowed to execute five motion primitives as shown in Fig. 5. Let the angular velocity of the right and left wheels be $\omega_l$ and $\omega_r$ respectively, with $\omega_l$, $\omega_r \in \{-1, 0, 1\}$. The robot's controls are defined by the vector $u = \{\omega_l, \omega_r\}$. Five motion primitives are generated by the following controls:

| | |
|---|---|
| $u_1 = (1, 1)$ | forward straight line motion |
| $u_2 = (1, -1)$ | clockwise rotation in place |
| $u_3 = (-1, 1)$ | counterclockwise rotation in place |
| $u_4 = (1, 0)$ | clockwise rotation w.r.t. point $rp$ |
| $u_5 = (0, 1)$ | counterclockwise rotation w.r.t. point $lp$ |

Executing the controls defined above, the robot explores the environment through moving in contact with the wall. If the omnidirectional sensor is placed at $rp$ then the robot follows the environment's boundary $\partial E$ in counterclockwise sense, and if the sensor is placed at $lp$ then the robot follows $\partial E$ in clockwise sense.

## 6 The Exploration Automaton

A finite-state machine (FSM) is defined as a mathematical model of computation, it is conceived as an

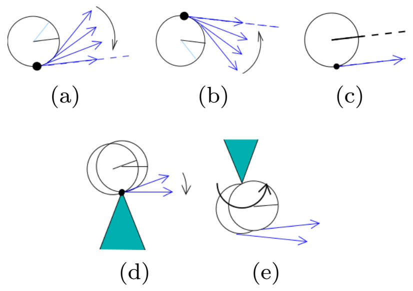


Fig. 5: The motion primitives: a) Clockwise rotation in place, b) Counterclockwise rotation in place, c) Straight line motion, d) Clockwise rotation w.r.t. $rp$, e) Counterclockwise rotation w.r.t. $lp$.

abstract machine that can be in one of a finite number of states [39]. The machine is in only one state at a time, it can change from one state to another by a triggering event or condition called *transition*. A FSM is defined by a list of its states, and the triggering condition for each transition. A special kind of FSM is the Moore machine which includes outputs associated with every state. According to the presented definition, *it is possible to represent the whole exploration strategy as a Moore machine.*

The FSM $M$ represents the robot's planner or exploration strategy. $M$ includes a motion policy and manages GNT queries and updates. The motion policy is a mapping from observations to controls (see Section 6.1). Note that the motion policy is only a part of the whole exploration strategy.

The task is not finished until a stop condition for exploration is met, this condition is not included in the motion policy because it requires topological information of the environment that is not given by the current sensor readings. This information is given by the GNT built during the robot's motion. As it is detailed in [11], the exploration task for a point robot ends when all the environment has been seen, it happens when *all the leaf nodes of the GNT are labeled as primitive ones* (the leaf nodes have a label called primitive). The condition to stop the exploration for a disc robot is similar to the one for a point robot, but includes the additional issue of gaps that never disappear. Note that due to the robot's dimensions, there may be some unreachable environment's regions yielding those gaps. Consequently, an algorithm called *local exploration* has been developed for dealing with this issue. See Algorithm 1, which is part of the exploration strategy.

A graphical representation of $M$ is shown in Fig. 6. There are seven states, one of them is the initial state when no motion primitive has been executed, there is an end state which establishes the GNT completeness,

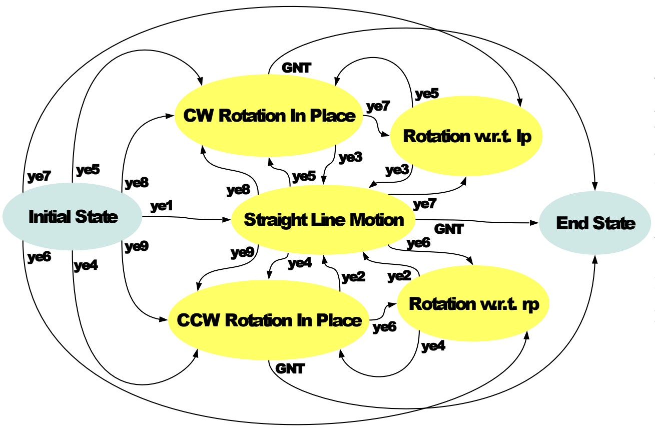


Fig. 6: The finite-state machine that represents the exploration strategy.

the task has been finished, so no motion primitive is applied and the robot stops its movement. The other states represent the execution of the motion primitives defined in Section 5. All the links in Fig. 6 are labeled with the corresponding observations defined in Section 4, with the exception of the links to the GNT. These links to the GNT represent queries to the GNT asking whether all the leaf nodes are marked as primitive ones.

GNT queries are done in states CCW Rotation in Place, CW Rotation in Place, and Straight Line Motion (see Fig. 6). This is because, the GNT might change because the occurrence of critical events while the robot is executing one of these motions. The queries are required to decide whether or not the exploration is terminated (i.e. the stop condition is met). Local exploration algorithm might be triggered in states CCW Rotation in Place or CW Rotation in Place. Local exploration algorithm also updates the labels of the gaps in the GNT.

The methodology to design the automaton consists in first defining the motion primitives needed to perform the task, which in general are equivalent to the states in the automata (in some tasks there may be additional states corresponding to the acquisition of some information with the robots sensors). Second, it is required to find the controls to execute them and the observations to activate a given controller. The controllers are synthesized according to the sensor's measurements available at each state in the automaton, from which both a feedback error is proposed and a controller to drive it to zero. This methodology might be used to build an automaton for other robotic tasks.

## 6.1 Motion Policy

The motion policy is based on the paradigm of avoiding the state estimation to carry out two consecutive mappings: $y \rightarrow x \rightarrow u$, that is from observation $y$ to state $x$ and then to control $u$, but instead of that there is a direct mapping $y \rightarrow u$.

Let $\gamma$ be a mapping function, the motion policy can be established by: $\gamma : \{0,1\}^6 \rightarrow \{-1,0,1\}^2$, then the function is expressed as $\gamma(ye_i) = (\omega_l, \omega_r) = u_j$. The motion policy is:

- $\gamma(ye_1 \vee ye_2 \vee ye_3) = u_1$
- $\gamma(ye_5 \vee ye_8) = u_2$
- $\gamma(ye_4 \vee ye_9) = u_3$
- $\gamma(ye_6) = u_4$
- $\gamma(ye_7) = u_5$

In which $\vee$ means "or".

The previous list summarizes the complete relationship between the controls and the observations given by the sensors.

## 6.2 The Local Exploration Algorithm

The configuration space restrictions for a disc robot might cause the presence of unreachable environment places. Those places might yield gaps that cannot disappear regardless of the robot motion. Once the point $lp$ or $rp$ lies on $\partial E$ it is possible to identify the observations that represent the presence of gaps that do not disappear. Those observations are: $ye_4^R = (0,1,0,1,x,1)$ or $ye_5^L = (1,0,0,1,x,0)$ (see Figs. 7 a) and b)).

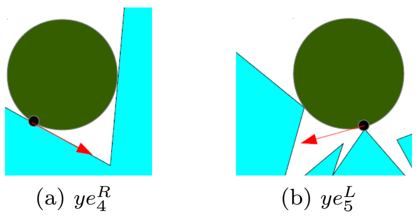


Fig. 7: Observations $ye_4^R$ and $ye_5^L$.

$ye_4^R$ means that the omnidirectional sensor is placed at $rp$, there is multi-contact between the robot and $\partial E$ and the point $rp$ is touching $\partial E$. Analogously, $ye_5^L$ means that the omnidirectional sensor is placed at $lp$, there is multi-contact between the robot and $\partial E$ and the point $lp$ is touching $\partial E$. They are special cases of $ye_4$ and $ye_5$ observations respectively, when any of these observations happen the local exploration algorithm is triggered (whose pseudocode is presented in Algorithm 1). The algorithm uses information from the

GNT, the algorithm ends after the nodes encoding gaps generated by vertices within an unreachable region are labeled as primitives.

The pseudocode of local exploration algorithm is presented in Algorithm 1, and it is described below for the case when the robot touches the environment border with point $rp$, the case when the robot touches the environment border with point $lp$ is just the symmetric one.

The main idea of this algorithm is the following. The gaps are ordered by angle in two lists, one starting from the direction of the first contact point (before the robot starts the rotation in place) and other starting from the direction of the second contact point (after the robot finishes the rotation in place). Additionally the direction of line $rt$ is used as common reference direction in both lists. Based on these three directions (direction of the first contact point, direction of the second contact point and direction of line $rt$) is possible to determine which gaps are generated by vertices that are not accesible to the robot.

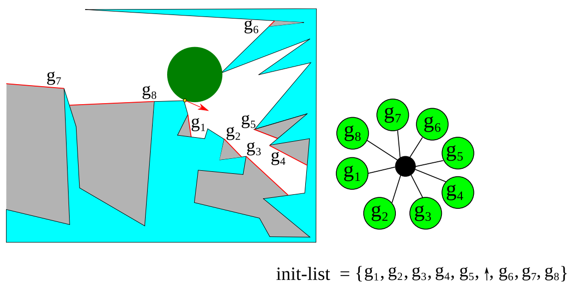


Fig. 8: list *init-list* and gaps before the robot's rotation in place.

Before the robot executes the rotation in place, the gaps are ordered by angle starting from the direction defined by the center of the robot and the location of point $rp$. The gaps are stored in a linear list called *init-list*. See Fig. 8. The changes (splits, merges, appearances and disappearances) in the gaps, due to the robot's rotation in place, are updated in list *init-list* until the sensor touches the second vertex.

After the robot finished the rotation in place a list called *end-list* is created, in this list the gaps are ordered by angle starting from the direction defined by the center of the robot and the location of second vertex that is touched by point $rp$. Lists *init-list* and *end-list* have different elements because the lists start and end at different angular directions. See Fig. 9. Note that gap $g_{13}$ has not been included in the lists, given that this gap has appeared during the robot's rotation in place and

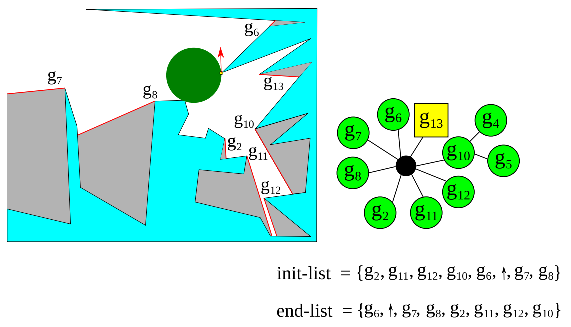

g6
g7
g8
g13
g10
g2
g11
g12
g4
g5
init-list = {g2, g11, g12, g10, g6, ↑, g7, g8}
end-list = {g6, ↑, g7, g8, g2, g11, g12, g10}

Fig. 9: lists *init-list* and *end-list*, and gaps after the robot's rotation in place.

as it was mentioned before, any time that a new gap appears in the GNT, it must be primitive.

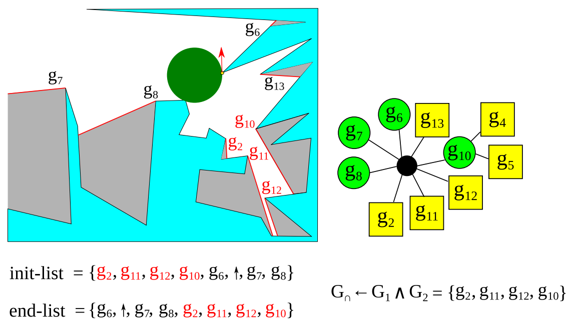

g6
g7
g8
g13
g10
g2
g11
g12
g4
g5
init-list = {g2, g11, g12, g10, g6, ↑, g7, g8}
end-list = {g6, ↑, g7, g8, g2, g11, g12, g10}
G∩ ← G1 ∧ G2 = {g2, g11, g12, g10}

Fig. 10: Lists $G_1$, $G_2$ and $G_\cap$

At the end of the rotation in place, the first gap on *init-list* is the first gap (in counterclockwise angular order) generated by a vertice that is not accesible, this gap is stored in $init_f$. The last gap in *end-list* is the last gap (in counterclockwise angular order) generated by a vertice that is not accesible, this gap is stored in $end_l$. As we have said above, the direction of line $rt$ is used as common reference direction in both lists and is stored in elements $init_{rt}$ and $end_{rt}$.

$G_1$ and $G_2$ are auxiliary lists containing specific subsets of *init-list* and *end-list* respectively. $G_1$ contains from $init_f$ until the element corresponding to direction $rt$. $G_2$ contains from the element corresponding to direction $rt$ until $end_l$. $G_\cap$ includes the elements that are common in both $G_1$ and $G_2$. Algorithm 1 finds $G_\cap$ and gives as output the updated GNT. In fact $G_\cap$ contains the common elements in $G_1$ and $G_2$ that correspond to the gaps within the unreachable region. See Fig. 10.

Those gaps are the ones that must propagate the primitive label to their offspring on the GNT.

**Algorithm 1** Local Exploration Algorithm

```
Input: GNT, current observation: $ye_i$.
Output: updated GNT.
if $rp$ =true then
  1. init-list ← Current gaps and $rt$ direction starting from the sensor's obstructed visibility region following a counterclockwise order;
  if $ye_i = ye_4^R$ ($u_3$ is executed) then
    while ($ye_i \neq ye_2$) and ($ye_i \neq ye_6$) do
      if GNT-event = true then
        if critical-event ≠ gap-appear then
          2. Apply the update suffered by the root's child nodes of the GNT to the corresponding gaps in init-list;
        end if
      end if
      3. Update the position of the $rt$ direction (due to the sensor's motion) in init-list according to the current angular counterclockwise order in the sensor reading;
    end while
    4. end-list ← Current gaps and $rt$ direction starting from the sensor's obstructed visibility region following a counterclockwise order;
    5. $G_1 \leftarrow \{x \in \textit{init-list} \mid init_f \leq x < init_{rt}\}$;
    6. $G_2 \leftarrow \{x \in \textit{end-list} \mid end_{rt} < x \leq end_l\}$;
  end if
end if
8. $G_\cap \leftarrow G_1 \wedge G_2$;
for every gap $g_i \in G_\cap$ do
  9. Label node $g_i$ in the GNT as a primitive node;
  10. Propagate the primitive label to the offspring of $g_i$;
end for
```

## 6.3 Proving some properties of the motion strategy

The following lemma corresponds to the case when the robot touches the environment border with point $rp$, the case when the robot touches the environment border with point $lp$ is just the symmetric one.

**Lemma 2** *The exploration strategy guarantees that all leaf gaps (i.e. gaps encoded as leaf nodes in the GNT) are labeled as primitive gaps.*

*Proof* The gaps that do not disappear are handled by Algorithm 1. If the robot is touching $\partial E$ with point $rp$ then the $G_1$ list includes all the gaps belonging to the angular interval between the direction defined by the center of the robot and the location of point $rp$ before the rotation in place and the $rt$ direction after the rotation in place. In this interval the order of the gaps is established in counterclockwise sense. Moreover, $G_2$ list includes all gaps belonging to the angular interval between direction $rt$ after the rotation in place and the direction defined by the center of the robot and the location of point $rp$ after the rotation in place. In the second angular interval, the gaps are also ordered in counterclockwise sense. The intersection $G_\cap$ between $G_1$ and $G_2$ includes only the gaps, ordered in counterclockwise sense, that lie between the direction defined by the center of the robot and the location of point $rp$ before the rotation in place and the direction defined by the center of the robot and the location of point $rp$ after the rotation in place. Those gaps are generated by reflex vertices located within the unreachable region. Observation $ye_4^R$ detects an unreachable region. The region is unreachable because the robot's bumper has touched $\partial E$ at two points. During the robot rotation in place, the omnidirectional sensor moves from a point touching $\partial E$ to the other, hence all gaps within the unreachable region are considered. Due to the possible split and merge critical events between these gaps, the primitive label of such gaps is propagated to all of the offspring of them in $G_\cap$. Each time that observation $ye_4^R$ occurs the local exploration algorithm is executed. Hence, all gaps encoded as leaf nodes (called leaf gaps) in the GNT are labeled as primitive gaps. □

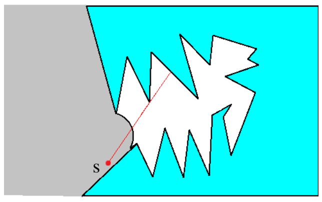


Fig. 11: $R_{na}$ is shown in white and in region $R_a$ in dark grey. The regions are divided by the arc of circle trajectory followed by the omnidirectional sensor during a robot's rotation in place. The figure also shows a source $s$ with a ray of light which goes from $R_a$ to $R_{na}$.

**Lemma 3** *The robot covers (observes) the largest possible portion of the environment with the omnidirectional sensor's visibility region.*

*Proof* The omnidirectional sensor trajectory during the rotation in place motion is an arc of circle, which divides the environment's interior in two regions, named accessible region $R_a$ and unaccessible region $R_{na}$, such that $R_a \cap R_{na} = \emptyset$. The boundary between those regions depends on the robot's radius. The omnidirectional sensor is unable of penetrating deeper in the unreachable region due to the configuration space restrictions, therefore, the arc of circle determined by the robot radius is the boundary between both regions. Refer to Fig. 11. It is clear that every ray of light emerging from any source $s \in R_a$ which touches $R_{na}$ must cross the regions' boundary as seen in Fig. 11. If the visibility polygon of $s$ includes a portion of $R_{na}$ then every ray of light emerging from $R_a$ to $R_{na}$ must cross the regions' boundary. Therefore every single ray of light traveling from any point $x \in R_a$ to $R_{na}$ must cross the regions' boundary. Hence, an omnidirectional sensor following the arc of circle trajectory guarantees observing the largest possible region of $R_{na}$. Observation $ye_4^R$ or $ye_5^L$ indicate that there is an unreachable region, each time that observation $ye_4^R$ or $ye_5^L$ occurs the omnidirectional sensor is moved over the boundary between the accessible and unaccessible regions. □

Supported by Lemmas 2 and 3 the following theorem states one of the main results of the paper.

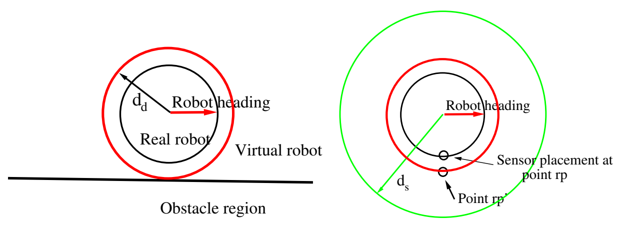
**Theorem 1** *For the class of environments described in Section 2, the exploration strategy modeled as the Moore machine $M$ presented in Section 6 (graphically represented in Fig 6) terminates. Upon termination, the robot has either covered with the omnidirectional sensor visibility region all of the environment or has covered its maximal geometrically possible portion. Consequently, the robot finds the landmark or declares that an exploration strategy to find it does not exist.*

*Proof* Since the environment is simply connected, a wall following strategy is enough for exploring all the environment for a point robot due to the absence of internal obstacles (generating more than one class of homotopic paths). For a disc robot, the gaps that are generated by reflex vertices located in reachable regions are labeled as primitive gaps, since the robot is able to reach the reflex vertices generating those gaps, then these gaps disappear. If there are unreachable regions, where some gaps do not disappear regardless the sensor's motions, then *local exploration* algorithm is executed. Lemma 2 guarantees that all leaf gaps are labeled as primitive ones, that is the stop condition for the exploration task. Hence, the exploration task terminates. Lemma 3 guarantees that the robot discovers the largest possible region of the environment. Hence, if the collision free subset of the configuration space $\mathcal{C}$ is simply connected then the landmark is found. If the collision free subset of the configuration space $\mathcal{C}$ has several connected components then the landmark might or might not be found. Again, by Lemma 3 the robot observes (discovers) the largest possible part of the environment, therefore when the landmark is not found, there does not exist a robot exploration strategy to find the landmark, for the connected component of the configuration space where the robot lies. □

## 7 Imperfect observations

In the sections above, we have presented the theoretical conditions guaranteeing the robot to discover the largest possible region of the environment. However, we assume that the controls are perfect and the robot moves in contact with the environment boundary (see Section 5). In the sequel of this work, our goal is to permit the robot's observations and commands to be imperfect, and to deal with the robot dynamics (i.e. velocities variations).

To achieve this goal, the robot follows the environment boundary to a desired distance different to zero. A *main idea* is to maintain a virtual circular robot of radius $d_d > r$, where $r$ is the radius of the real robot, in contact with $\partial E$. Refer to Fig. 12a.



(a) Virtual robot (b) Circle to reduce linear velocity

Fig. 12: Two discs

In general, it is more flexible and practical to follow the environment boundary to a small distance, instead of moving in contact with it, since the controls yielding the robot motion can be imprecise. Hence, it is less likely that the robot collides with the obstacles. Furthermore, all the required feedback information can be obtained directly from a laser range finder.

Additionally, we will use a circle of radius $d_s > d_d$ to detect an obstacle that is close to the robot to reduce the robot's linear velocity, refer to Fig. 12b. We call $rp'$ to the point at distance $d_d$ from the robot center, in the direction perpendicular to the robot heading and at the right of the heading.

In Section 4, an observation is defined by a vector with six binary elements and only nine useful observations. Below, we show that the binary elements of the observation vector representing abstract information can be obtained using a laser range finder, which implements the omnidirectional sensor referred along the previous sections.

For simplicity, in the sequel of this work, we will assume that the omnidirectional sensor is located at $rp$. The case when the omnidirectional sensor is placed at $lp$ is just the symmetric case. For this reason, the binary elements $lp$ and $o$ are not used. Thus, we only use four of the six binary elements defined in Subsection 4.2.

Besides, we add two new binary elements which are described below.

Omnidirectional sensor measurements are used to compute several angles and distances in local reference frames attached to the robot. These angles and distances themselves are used in two ways: 1) They are transformed to binary observations that determine the transitions between states in a state machine which represents the complete robot exploration motion strategy. 2) They are used as feedback information in a hybrid control scheme, whose objective is to navigate the robot maintaining a desired distance between the robot and the boundary of the environment. This objective is achieved enforcing convergence of some errors over the measured distances or angles.

To detect the features in the environment (corners that delimit walls), we use a local and simple line fitting technique that find convex and concave corners. First, the closest laser point from the laser sensor is detected. Second, the angles between the ray from the laser sensor to the closest point and the rays between the closest point and the next 10 sensed points (in counterclockwise sense) are measured. These 10 angles are averaged and the resulting angle is called reference angle. Third, the angle between the ray from the laser sensor to the closest point and the ray between the closest point and a given sensed point is measured, this angle is called angle of the point. If the angle of the point is smaller than the reference angle plus a given threshold, then a concave corner is detected. Analogously, if this angle of the point is larger than the reference angle plus a given threshold then a convex corner is detected. For more complex environments, other well know algorithms exist to fit lines based on points [23,40].

Below, we describe how to obtain the binary elements of an observation vector $yc_i$ based on measurements of the laser range finder, we also describe the relevant feedback measurements (in terms of angles and distances) used in each motion primitive executed by the robot.

## 7.1 Observations for straight line motion

Refer to Fig. 13. The line passing over points $rp$ and $rp'$ is called line $rp - rp'$. The ray pointing to the closest point obstacle over the line segment that the robot follows is called $r_{min}$. Let $\theta_1$ be the angle from line $rp - rp'$ to the ray $r_{min}$ measured in counterclockwise sense, $\theta_1 \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Let $d_1$ be the smallest distance from the robot's center to the line segment that the robot is following.

The robot reduces its linear velocity whenever an obstacle different to the wall that the robot is following

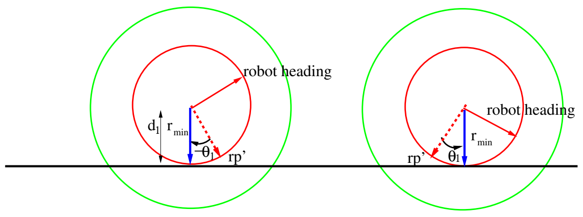
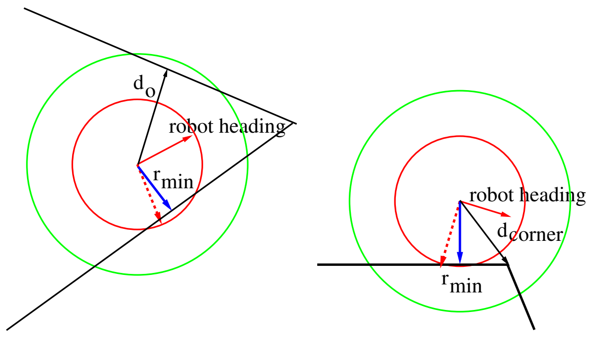


Fig. 13: Angle $\theta_1$ and ray $r_{min}$

is closer than distance $d_s$, the robot also reduces its linear velocity if there is a visible convex corner (also called a reflex vertex) closer to it than distance $d_s$. One of the two new bits, that we use in the observation vector is called $st$, this bit is set to 1 if there is an obstacle or convex corner closer to the robot center than distance $d_s$, see Table 1.

The omnidirectional laser range finder is use to measure distance $d_o$, which is the distance between the robot center and the closest obstacle that does not belong to the wall that the robot is following. Distance $d_{corner}$ is also measured using the laser, $d_{corner}$ is the distance between the robot center and the closest visible convex corner. See Figs. 14a and 14b.



(a) $d_o$ (b) $d_{corner}$

Fig. 14: Distances $d_o$ and $d_{corner}$.

The simple line fitting method described at the beginning of this section is used to build segments and detect convex and concave corners. Alternatively, a convex corner can also be located since it generates a distance discontinuity (a gap), over two angular consecutive sensor readings.

It is said that point $rp'$ is in contact with an obstacle, if there is an obstacle point (sensor reading) closer to $rp'$ than a given threshold $\epsilon_1$. The bit $rp'$ is set to 1 in Table 1. See Fig. 15 a). It is said that there is single contact at point $rp'$ if there is a single circular sector of radius $d_d$ (representing the virtual robot) that intersect an obstacle and the point $rp'$ belongs to that circular sector. The bit $sc$ is set to 1 in Table 1, see Fig. 15 b).

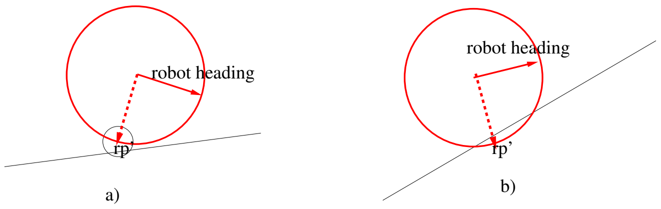


Fig. 15: Single contact at $rp'$

It is said that the robot heading is aligned with the wall that the robot is following if $|\theta_1| < \epsilon_2$ and there are not obstacle points that do not belong to the wall (line segment) that the robot is following, closer to the robot center than the robot radius $d_d$. If the above condition holds then bit *aligned* is set to 1 in Table 1.

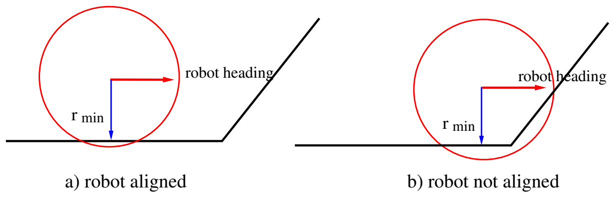


Fig. 16: Robot aligned and robot not aligned

The controllers $SLW$ and $SLWD$ presented in Section 9 use distance $d_1$ and angle $\theta_1$ as feedback information to keep the robot following the wall and aligned with it. Additionally, controller $SLWD$ uses distances $d_o$ or distance $d_{corner}$ as feedback information to reduce the robot linear velocity whenever the robot gets close to an obstacle or convex corner, see Section 9.

## 7.2 Observations for rotation in place

If the robot is in contact with an obstacle and the robot is not aligned with it, or if there is a bicontact between the robot and the obstacle region then robot rotates in place.

It is said that the robot is in single contact with an obstacle if a single circular sector of the virtual disc robot intersects the obstacle region. The bit $sc$ is set to 1 in Table 1. Fig. 17 a) shows the case when the robot is in contact with a segment of the polygonal environment and it is not aligned with that segment. The bit *aligned* in Table 1 is set to 0. This case happens when the robot moves from the interior of the polygonal environment to reach the boundary of the obstacle region.

Fig. 17 b) shows the case when there is a bicontact between the polygonal region and the environment. It is said that there is a bicontact if two different circular sectors of the virtual disc robot intersect the obstacle region. The bit $bc$ in Table 1 is set to 1.

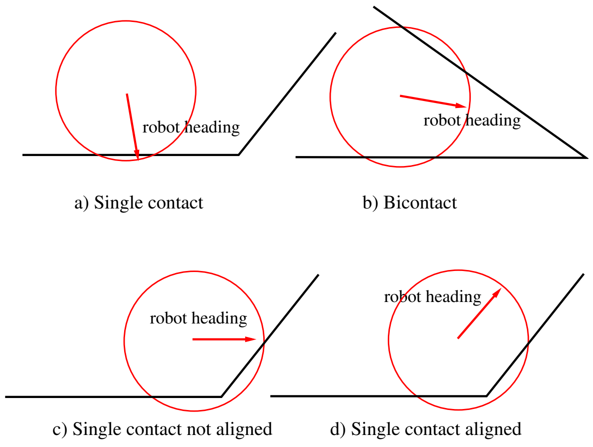


Fig. 17: Single contact, robot not aligned and bicontact

It is said that the robot is in single contact with an obstacle and it is not aligned with it, if the virtual disc robot intersects more than one polygonal segments of the polygonal environment, and the robot heading is not aligned with the last segment in counterclockwise sense, see Fig. 17 c). The bit $sc$ is set to 1 and the bit *aligned* is set to 0 in Table 1. This case happens when the robot is following a segment and it encounters a concave corner. In contrast, it is said that the robot is in single contact with an obstacle and the robot is aligned with it, if the disc robot intersects more than one segments of the polygonal environment, and the robot heading is aligned with the last segment in counterclockwise sense, see Fig. 17 d). Bit $sc$ is set to 1 and bit *aligned* is also set to 1 in Table 1.

If the disc shaped robot intersects more than one polygonal segment then two or more consecutive rotations in place might be executed. The robot might be aligned with a segment, however there might be other obstacle blocking the robot.

We use the line fitting method described above to find the last segment in counterclockwise sense of the obstacle region boundary to align the robot heading with it. The line fitting algorithm that we use considers that the robot (and hence the omnidirectional sensor) is inside the polygonal environment and that it does not have access to the whole map, but only to the laser points which are visible, this is equivalent to reason over the visibility polygon, see Fig. 18. In the figure, the blue segments are visible segments and the red segments are gaps (also called free segments).

Recall that, the line passing over points $rp$ and $rp'$ is called line $rp - rp'$ and the ray pointing to the closest

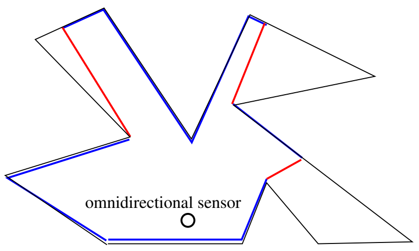


Fig. 18: Visibility polygon

point obstacle is called $r_{min}$. $\theta_2$ is the angle between the line $rp - rp'$ and the ray $r_{min}$, $\theta_2 \in (0, \pi)$. See Fig. 19.

The controller $RP$ that will be presented in Section 9 use angle $\theta_2$ as feedback information to make the robot to rotate in place.

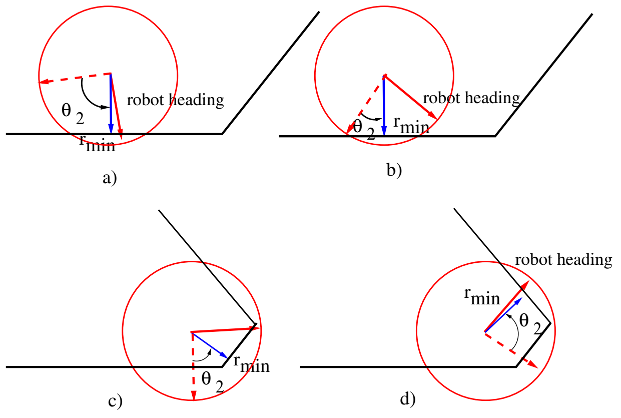


Fig. 19: Angle $\theta_2$ and ray $r_{min}$

If the robot heading is aligned (i.e. $|\theta_2| < \epsilon_2$) with the last polygonal segment, in counterclockwise sense, of the obstacle region boundary then the rotation in place terminates with success, see Fig. 19 d).

The other new bit is called $rp' - e$ (see Table 1), this bit is set to 1 if point $rp'$ is closer to a convex corner than a given threshold $\epsilon_1$. A rotation in place also terminates if point $rp'$ is closer to a convex corner than a given threshold $\epsilon_1$. See Fig. 20 d).

If the disc shaped robot is in bicontact with the polygonal region, then only the last circular sector of the disc robot in counterclockwise order is considered to terminate the rotation in place motion.

## 7.3 Observations for rotation with respect to a convex corner

The robot rotates with respect to a convex corner or point $rp'$, following an arc or circle, whenever the point $rp'$ is closer to a convex corner than a given threshold $\epsilon_1$. The above condition set the bit $rp' - e$ equals to 1. The ray from the robot center pointing to that convex corner is called $r_{corner}$.

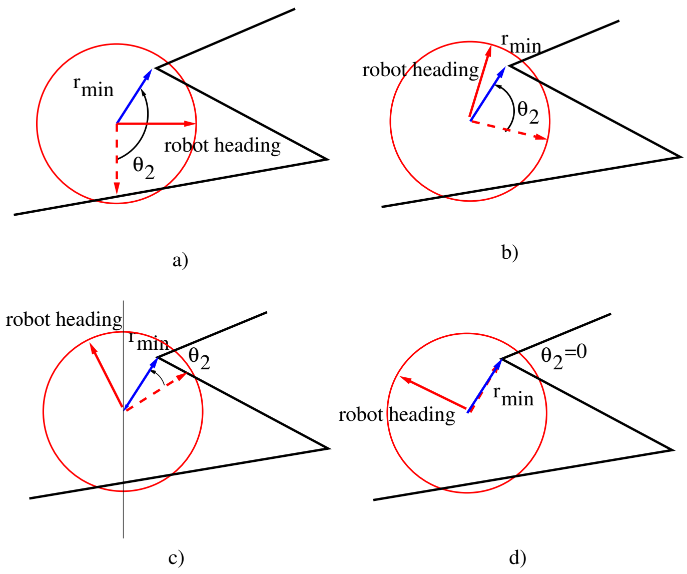


Fig. 20: Angle $\theta_2$ and ray $r_{min}$, touching a convex corner

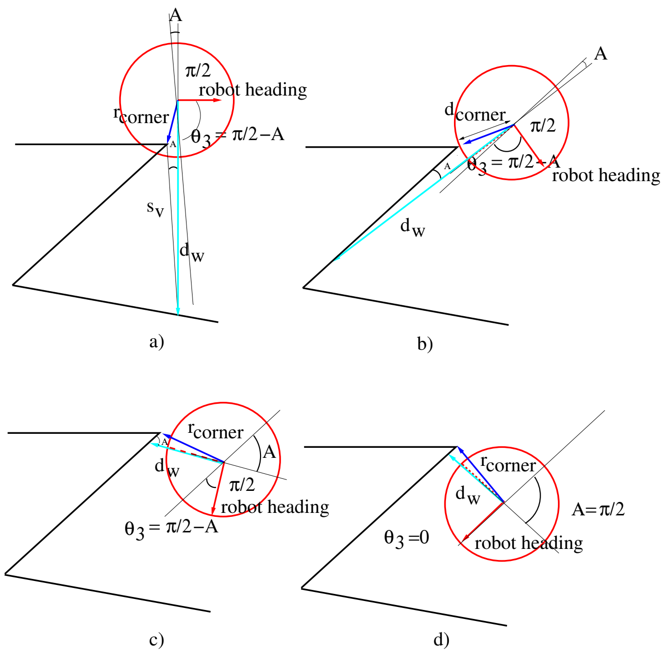


Fig. 21: Angle $\theta_3$, line $rp - rp'$ and ray $r_{corner}$ are not colinear, rotation with respect to point $rp'$

Angle $\theta_3$ is used as feedback information during this type of rotation. Here, we describe the general case, in which the line $rp - rp'$ is not colinear with the ray $r_{corner}$, this happens because of an imprecision on the robot motion that occurs when the robot is approaching the convex corner. See Fig. 21. Two distances are used to compute angle $\theta_3$, the distance to the corner $d_{corner}$ and the distance to an obstacle in the direction of the line $rp - rp'$, this second distance is called $d_w$. The cosines law is used to compute an auxiliar angle called $A$, angle $\theta_3 = \frac{\pi}{2} - A$. Note that the arc of circle that the robot executes is centered in point $rp'$ and not in the convex corner. Note also that some times the robot is aligning its heading to a virtual line segment (denoted $s_v$) , see Figs. 21 a) and b). However, as the line passing over point $rp$ and $rp'$ is getting perpendicular to that line segment, the correct segment shall be sensed and considered, see Fig. 21 c). The arc of circle terminates when the angle $\theta_3$ is smaller than a threshold $\epsilon_2$, this is equivalent to have the robot heading aligned with the segment after the corner, in counterclockwise sense, see Fig. 21 d).

An additional complication might happend to compute the angle that the robot must rotate, when it travels an arc of circle around a convex corner. The complication corresponds to the following fact, it might be an obstacle that the robot touches before its heading is aligned with the segment after the corner. We use a simple line fitting technique to detect the potential collision points that do not belong to the line segment after the convex corner.

Thus, if there exist potential collision points closer to the robot than distance $d_s$ then an auxiliar angle $\theta_4$ is computed to determine the angle that the robot must rotate. The computation of angle $\theta_4$ is based on the following observation: when the robot rotates following an arc of circle, all the points over the periferia of the disc shaped robot rotate with respect to a given point and they rotate the same angle. For simplicity, we describe the procedure assuming that the line $rp - rp'$ and the ray $r_{corner}$ are colinear, then the robot rotates following an arc of circle centered at the corner. However, in general when the line $rp - rp'$ and the ray $r_{corner}$ are not colinear, hence the robot rotates an arc of circle centered at point $rp'$.

To compute $\theta_4$, distances $d_{corner}$ and $d_o$ are used, $d_o$ is the distance to the closest obstacle point. The law of cosines can also be used to compute the distance between the convex corner and the point at distance $d_o$ from the robot center. This distance is called $d_{co}$. To find the point that collides with the closest obstacle (from the robot center), a circle centered at the convex corner (center of rotation) of radius $d_{co}$ is used. This circle centered at the corner and the circle representing the disc shaped robot are intersected. The intersection point closer to the closest obstacle (from the robot center) is called point $IC$. $\theta_4$ is the angle between the ray from the corner to the point $IC$ and the ray from the corner to the point obstacle closest to the robot center. The Fig. 22 a) shows the case when the obstacle is a corner and Fig. 22 b) when the obstacle is a segment. Finally, the angle that the robot must rotate around the corner is $\min\{\theta_3, \theta_4\}$. Angle $\theta_4$ is found at each iteration of the method, note that is possible to decide which angle $\theta_3$ or $\theta_4$ is smaller, at every time instance.

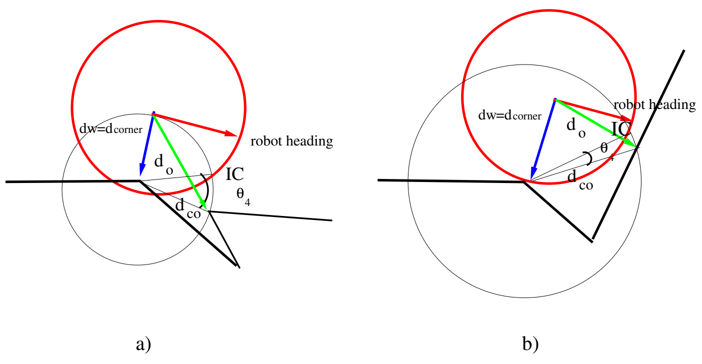


Fig. 22: Angle $\theta_4$, line $rp - rp'$ and ray $r_{corner}$ are colinear, rotation with respect to the convex corner.

The controller $AC$ that will be presented in Section 9 use either angle $\theta_3$ or angle $\theta_4$ as feedback information to make the robot to rotate around a convex corner.

### 7.4 Allowing some approximations

We stress the fact that some approximations are allowed, 2 thresholds are used. Threshold $\epsilon_1$ is the radius of the circle modeling point $rp'$ that determines whether or not a convex corner is touching point $rp'$ or whether or not the robot is touching a wall with point $rp'$. Threshold $\epsilon_2$ is an angular threshold that determines whether or not the robot is aligned, the same threshold is used for angles $\theta_1$, $\theta_2$, $\theta_3$ and $\theta_4$. The tuning of these parameters is a compromise between precise control action and robustness against imperfect sensor readings.

## 8 Finite State Machine and Switching Control Scheme

Analogously to the case in which the robot moves in contact with the environment boundary, it is possible to obtain a new Moore machine, for the case where the robot has as objective to move to a desired distance from the environment boundary.

In this last case, one or more observations activate a specific controller. Table 1 presents the observation that activates each controller.

| $yc_i =$ | $(rp',$ | $sc,$ | $bc,$ | $st,$ | $rp'-e,$ | $aligned)$ | Control |
|---|---|---|---|---|---|---|---|
| $yc_1 =$ | (0, | 0, | 0, | 0, | X, | X) | *SLI* |
| $yc_2 =$ | (0, | 0, | 0, | 1, | X, | X) | *SLID* |
| $yc_3 =$ | (1, | X, | X, | 0, | X, | 1) | *SLW* |
| $yc_4 =$ | (1, | X, | X, | 1, | X, | 1) | *SLWD* |
| $yc_5 =$ | (X, | 1, | 0, | X, | 0, | 0) | *RP* |
| $yc_6 =$ | (X, | 0, | 1, | X, | X, | 0) | *RP* |
| $yc_7 =$ | (1, | X, | X, | X, | 1, | 0) | *AC* |

Table 1: Observations $yc_i$

Recall that the controllers use angles $\theta_1$, $\theta_2$, $\theta_3$ and $\theta_4$, and distances $d_1$, $d_o$ and $d_{corner}$ as feedback information to execute the robot motions.

The relation between an observation and the activation of a given controller is given by:

- $yc_1 \rightarrow SLI$
- $yc_2 \rightarrow SLID$
- $yc_3 \rightarrow SLW$
- $yc_4 \rightarrow SLWD$
- $yc_5 \vee yc_6 \rightarrow RP$
- $yc_7 \rightarrow AC$

In which $\vee$ means "or".

The robot still executes basic motion primitives: straight line, rotation in place and arc of circle, however, sensed information is used to correct a possible deviation from the motion primitives.

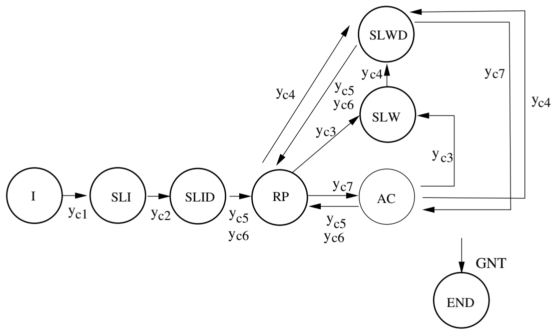


Fig. 23: The finite-state machine.

The Moore machine defines the possible transitions between states given by one or more observations, and gives the exploration's termination condition. A graphical representation of this Moore machine is shown in Fig. 23.

It is important to stress that the activation of a given controller depends on both the observation and the state in the finite state machine, thus the automaton constraints the possible states transitions filtering spurious observation due to noisy sensor readings. In this approach the planning stage corresponds to the design of the FSM and it is done prior to execution, once this is done, for any execution instance and for any different environment, the method is reactive, it just relates observations to controls.

The GNT gives the termnation condition for the exploration task. A GNT link represents a query to the GNT asking whether all the leaf nodes are marked as primitive ones. The exploration task might terminate in any state where the sensor is moving.

The local exploration algorithm presented in Subsection 6.2 is designed for the case in which the robot moves in contact with the environment boundary. The algorithm is used to detect gaps, which are generated by reflex vertices located within an unreachable region.

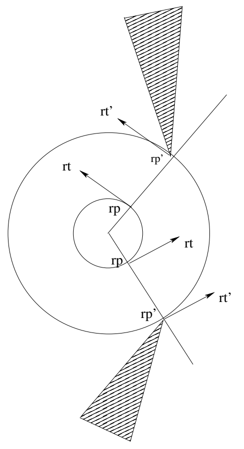


Fig. 24: A virtual robot of radius $d_d$

When the robot does not move in contact with the environment boundary, since the sensor is not located at point $rp'$ then the sensor will not discover the same portion of the environment, compared with the case in which the sensor is placed at point $rp'$. However, local exploration algorithm can still be used to label observed gaps generated by reflex vertices located within an unreachable region as primitive gaps. Now, the unreachable region means that a robot of radius $d_d$ cannot reach the region in instead of a robot of radius $r$.

Recall that local exploration algorithm is based on the direction of the first contact point, the direction of the second contact point and direction of line $rt$. Lines $rt$ and $rt'$ have the same direction, and also the direction of the line from the center of the robot to point $rp$ is the same that the direction of the line passing by points $rp$ and $rp'$, see Fig. 24. Therefore, the angular order of the direction of these lines with respect to the gaps will not change regarless of whether the environment border is touched with point $rp$ or $rp'$. Hence, local exploration algorithm can still be used and the exploration strategy terminates.

The only difference is that a new observation will start the execution of the algorithm. This observation is a particular case of observation $yc_6$ in which point $rp'$ is in contact with the environment boundary. While robot rotates in place the GNT is updated according the gaps' critical events until point $rp'$ is again in contact with the environment.

Unfortunately, since the robot does not move in contact with the environment, the portion of an unreachable region that will be covered with the omnidirectional sensor's visibility region is in general smaller compared with a robot that moves in contact with the environment.

## 9 Dealing with imperfect actions: Feedback based controllers

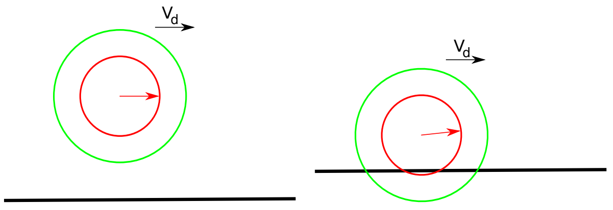
In this section, we detail the proposed switching control scheme. The activation of a given controller depends on both the observation and the state in the finite state machine. In the control scheme $V_c$ is the current velocity at the moment when a critical event happens, either a controller is activated (because an observation changes) or while executing a controller a feedback measurement changes.

### 9.1 SL in interior ($SLI$)

This controller is activated when observation $yc_1$ is detected. In this case, the robot is in the interior of the environment as shown in Fig. 25a. The controller has to drive the robot in straight line towards the environment boundary. To do so, the angular velocity is set to zero and the linear velocity is smoothly increased from $V_c$ (it might be zero) to $V_d$ the maximal desired linear velocity. Note that $V_c$ is the current linear velocity at the moment when the controller is activated because the change of observation; this velocity is denoted $V_{c|t=0}$. Thus, the controller looks for reaching velocity $V_d$ in open loop since there is no information available for feedback. $SLI$ is the only controller that works in open loop. Such behavior is achieved using the following controller:

$$\begin{aligned} V &= \frac{V_d - V_{c|t=0}}{2}(1 + \tanh(\alpha(t - \beta))) + V_{c|t=0} \\ \omega &= 0 \end{aligned} \tag{1}$$

where $\alpha$ is a scale factor and $\beta$ is a time shifting. These parameters are related to the duration of the transition from $V_{c|t=0}$ to $V_d$, they are computed such that the maximum robot acceleration is not exceeded.



(a) $SL$ in interior (b) $SL$ following a wall

Fig. 25: Straight line ($SL$) motion primitive.

### 9.2 SL in interior with deceleration ($SLID$)

The observation that triggers this controller is $yc_2$. In this case, the robot is in the interior of the environment and it is moving in straight line (zero angular velocity). An obstacle is detected at a distance smaller than $d_s$ as shown in Fig. 26.

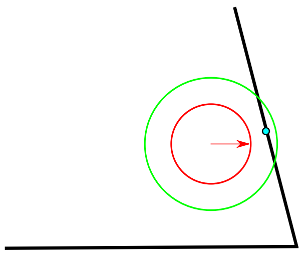


Fig. 26: Straight line in interior with deceleration.

The robot must reduce its linear velocity until it stops when the distance to the obstacle is equal to $d_d$. To achieve that, the controller is define as follows:

$$\begin{aligned} V &= k_1 e_o \\ \omega &= 0 \end{aligned} \tag{2}$$

where $e_o = d_d - d_o$, $k_1$ is a control gain given by $k_1 = \frac{V_{c|t=to}}{d_d - d_s}$. Here $V_{c|t=to}$ is the robot speed at the moment that an obstacle gets closer to the robot than distance $d_s$. This particular gain provides continuity in the linear velocity at the switching time from controller $SLI$ to $SLID$.

### 9.3 SL following a wall ($SLW$)

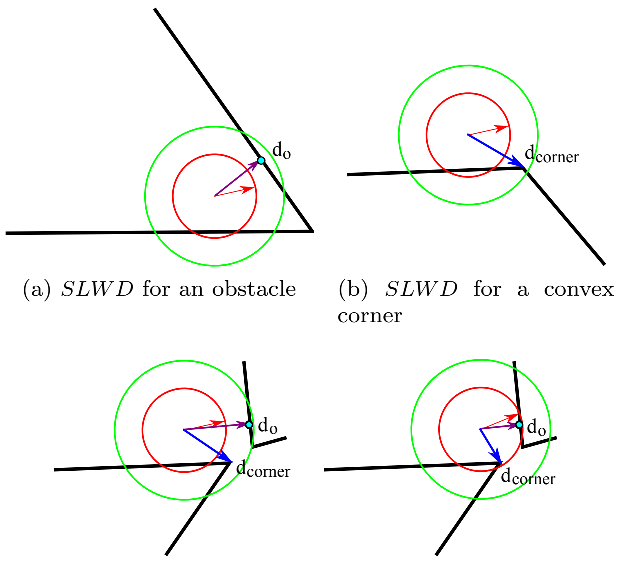
The observation that triggers this controller is $yc_3$. Fig. 25b shows a case where the $SLW$ controller is used. Similar to the $SLI$ controller, this one looks for reaching the desired linear velocity $V_d$ using a smooth transition from the current velocity $V_{c|t=0}$, here $V_{c|t=0}$ is the robot speed at the moment when the controller is activated. Besides, the robot orientation must be controlled to be aligned with a wall, which is achieved by using a controller with two feedback components in terms of distance and angular deviation. The controller is given by:

$$V = \frac{V_d - V_{c|t=0}}{2}(1 + \tanh(\alpha(t - \beta))) + V_{c|t=0} \qquad (3)$$
$$\omega = k_2 e_d + k_3 \theta_1$$

where $e_d = d_d - d_1$, $k_2$, $k_3$ are control gains. Notice that this controller starts when the robot is stopped and aligned with a wall. Thus, the angular velocity is close to zero while this controller is active and continuity of the angular and linear velocities is achieved. To obtain gains $k_2$ and $k_3$ we proceed as follows. These gains are manually tuned, we start with small positive gains and then the values are increased to obtain a faster converge of the errors to zero. This process is repeated while no oscillation in the robot's trajectory appears.

### 9.4 SL following a wall with deceleration ($SLWD$)

In this case the robot is following a wall correcting its orientation through its angular velocity as in the $SLW$ controller. Differently to the $SLW$ controller, if an obstacle or a convex corner are detected at a distance smaller than $d_s$ then the robot must reduce its linear velocity from the current value to zero. See Figs. 27a and 27b. The robot stops when the distance to the obstacle or to the convex corner is equal to $d_d$. If both an obstacle and a convex corner are detected at a distance smaller than $d_s$ then the robot must slow down until it stops when the $\min\{d_o, d_{corner}\}$ is equal to $d_d$. See Figs. 27c and 27d. The observation $yc_4$ activates the $SLWD$ controller, which is defined as:



(a) $SLWD$ for an obstacle (b) $SLWD$ for a convex corner

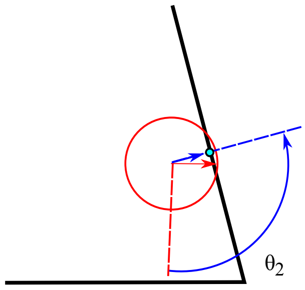
(c) $SLWD$ $\min\{d_o, d_{corner}\}$ is $d_{corner}$ (d) $SLWD$ $\min\{d_o, d_{corner}\}$ is $d_o$

Fig. 27: Straight line following a wall with deceleration ($SLWD$).

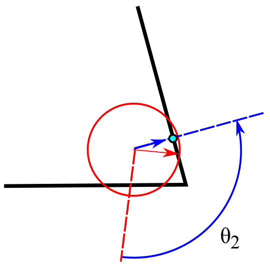
$$V = k_4 e_p \qquad (4)$$
$$\omega = k_2 e_d + k_3 \theta_1$$

where $e_p = \bar{e}_d - e_o$, $e_o = d_d - \min\{d_o, d_{corner}\}$ and $k_2$, $k_3$ and $k_4$ are control gains. The reference signal $\bar{e}_d$ is set depending on the previous state of the FSM. If the previous state is $SLW$ then $\bar{e}_d = 0$ and the control gain must be $k_4 = \frac{V_{c|t=to}}{d_d - d_s}$. Here $V_{c|t=to}$ is the robot speed at the moment that an obstacle gets closer to the robot than distance $d_s$. Gain $k_4$ is adjusted using the equation above to avoid a discontinuity on the robot speed.

If the previous state is $RP$ or $AC$ then the $SLWD$ controller initiates when the robot is stopped. In this case, the reference signal is set as the time-varying profile $\bar{e}_d = \frac{e_{o|t=0}}{2}(1 + \cos(\frac{\pi t}{\tau_1}))$. This reference tracking controller generates a smooth linear velocity that starts in zero and finishes in zero at time $\tau_1$. Thus, continuity in the linear velocity is achieved even if the $SLWD$ controller is launched and there are obstacles closer to the robot than distance $d_s$. If the obstacle closest to the robot changes (from a corner to a wall or viceversa) then the gain $k_4$ is modified to $k_4 = \frac{V_{c|t=ts}}{d_d - \min\{d_o, d_{corner}\}}$ to maintain the continuity of the linear velocity. Here $V_{c|t=ts}$ is the robot speed at the moment that the closest obstacle to the robot changes provided that the obstacle is closer to the robot than distance $d_s$. This controller also maintains continuity of the angular velocity.

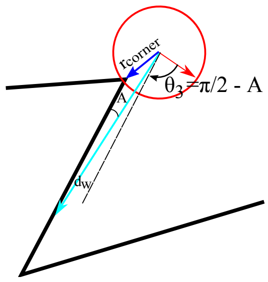


(a) $RP$ with a single contact over the disc of radius $d_d$

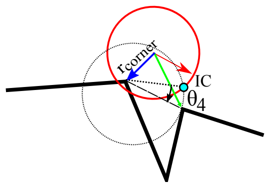


(b) $RP$ with multiple contacts over the disc of radius $d_d$

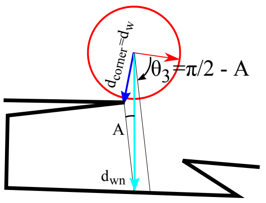
Fig. 28: Rotation in place ($RP$) motion primitive.



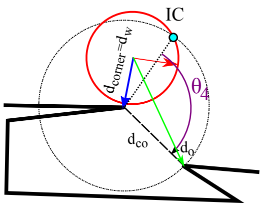
(a) $AC$ using $\theta_3$ for feedback



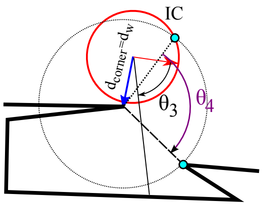
(b) $AC$ using $\theta_4$ for feedback



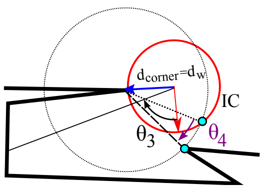
(c) $AC$ measuring $\theta_3$



(d) $AC$ measuring $\theta_4$



(e) $AC$ where the $\min\{\theta_3, \theta_4\}$ is $\theta_3$



(f) $AC$ where the $\min\{\theta_3, \theta_4\}$ is $\theta_4$

Fig. 29: Arc of circle (AC) motion primitive.

## 9.5 Rotation in place ($RP$)

This controller is activated with two different observations. A first case occurs with the observation $ye_5$, which happens when the robot is at distance $d_d$ of the environment boundary but it is not aligned with it. See Fig. 28a. A second case occurs when the observation $yc_6$ is measured. This happens when there is a multi-contact between the robot and the environment boundary, see Fig. 28b. In both cases the angle $\theta_2$ must be taken from its initial value to zero smoothly. To achieve that, the following trajectory tracking controller is proposed:

$$
\begin{aligned}
V &= 0 \\
\omega &= k_5 e_{\theta_2}
\end{aligned}
\tag{5}
$$

where $e_{\theta_2} = \theta_d - \theta_2$, $\theta_d = \frac{\theta_{2|t=0}}{2}\left(1 + \cos\left(\frac{\pi t}{\tau_2}\right)\right)$ and $k_5$ is a control gain, which is tune manually following a similar procedure that the case of gains $k_2$ and $k_3$. Notice that observations $yc_5$ and $yc_6$ occurs when the robot is stopped (the previous states of the FSM can be $SLID$, $SLWD$ or $AC$, which terminate with a motionless robot).

The trajectory tracking controller generates a smooth angular velocity that aligns the robot to the wall, the duration of the motion generated by the controller is $\tau_2$, starting and finishing with $\omega = 0$.

## 9.6 Arc of circle ($AC$)

This controller is activated by the observation $yc_7$. The goal is to move the robot along an arc of circle of radius $d_d$ and to align its heading to the next edge of the polygonal environment after the corner. The center of this rotational motion is a convex corner or point $rp'$. However, the robot cannot get aligned with that polygonal edge after the corner if there is an obstacle that intersects the disc of radius $d_d$ during the arc of circle robot's motion.

If the robot does not detect obstacles that prevent its alignment with the next edge of the environment then the measure $\theta_3$ is used for feedback (see Fig 29a).

If the robot detects obstacles that prevent its alignment (see in Fig. 29b) then the measure $\theta_4$ is used for feedback. If both angles $\theta_3$ and $\theta_4$ can be measured during the motion then the minimum of $\theta_3$ and $\theta_4$ is used for feedback, $\theta_{AC}$ is set to $\min\{\theta_3, \theta_4\}$. See Figs. 29c and 29d.

We propose the following $AC$ controller, which also aims to achieve continuity in the robot velocities:

$$
\begin{aligned}
V &= k_t \theta_{AC}, \\
\omega &= \omega_n + k_6 e_c + k_7 \frac{de_c}{dt}
\end{aligned}
\tag{6}
$$

where $\omega_n = -V/d_d$ represents a nominal angular velocity to achieve that the robot moves along an arc of circle of radius $d_d$ and the error $e_c = d_d - d_{corner}$. Any deviation is corrected by the proportional and derivative terms of the error $e_c$, which are weighted by control gains $k_6$ and $k_7$. Similar to the case of the straight line controllers, to tune gains $k_6$ and $k_7$, one starts with small positive gains and then the values are increased to obtain a faster converge of the errors to zero. This is repeated while no oscillation in the robots trajectory appears.

The FSM makes the controller $AC$ to start when the robot is motionless, in this controller a variable gain is used as follows:

$$k_t = \frac{V_d}{2\theta_{AC|t=0}}(1-\cos(\frac{\pi t}{\tau_3})) \text{ if } t < \tau_3 \tag{7}$$

$$k_t = \frac{V_{c|t=\tau_3}}{\theta_{AC|t=\tau_3}} \text{ if } t \geq \tau_3$$

This variable gain allows us to start the arc of circle at zero initial velocity, then the robot has to speed up trying to reach its maximum allowable speed $V_d$ and finally the linear velocity returns to zero when $\theta_{AC}$ is zero. $\tau_3$ is a parameter that determines the time needed to reach the maximum value of the gain.

In Fig. 29e, we present a case in which is possible to measure both angles $\theta_3$ and $\theta_4$ at the same time. Fig. 29e shows a case where $\theta_3 < \theta_4$ and $\theta_3$ is used for feedback. Subsequently, during the robot's motion tracing an arc of circle, it happens that $\theta_4 < \theta_3$, which is shown in Fig. 29f. Consequently, the angle for feedback changes from $\theta_3$ to $\theta_4$, which might yield an undesired discontinuity in the robot velocities. To alleviate this issue, we propose to adapt the control gain $k_t$ making $k_t = \frac{V_{c|t=t_s}}{\min\{\theta_{3|t=t_s},\theta_{4|t=t_s}\}}$, where $t_s$ is the time when the minimal angle changes from $\theta_3$ to $\theta_4$ or viceversa according to the minimum value. Here $V_{c|t=t_s}$ is the robot speed at the moment when $\min\{\theta_{3|t=t_s},\theta_{4|t=t_s}\}$ changes from $\theta_3$ to $\theta_4$ or viceversa.

## 10 Implementation

The simulation and experiments presented in this section have the following main objectives: They show that the robot does not need to travel all the environment boundary to finish the exploration. Thus, the experiments validate the use of the GNT to detect the termination of the exploration task, as soon as all the leaf nodes in the GNT are primitive the exploration is finished. The experiments also empirically show the pertinence of the automaton. One can observe that the automation diminishes the change of controllers producing smoother robots velocities compared with a control scheme, in which the controller change based only in the observations. Finally, our experiments verified that the controllers work properly, the robot is able to follow the walls and robot's velocities did not present discontinuities.

### 10.1 Exploration's simulations

The whole method presented in Section 6 has been implemented and simulations' results are included. All our simulation experiments were run on a 2.2GHz Intel Core i7-2670QM quad-core processor PC, equipped with 8 GB of RAM, running Linux, and were programmed in C++ using the computational geometry library LEDA. Our software implementation exactly emulates the FSM presented in Fig. 6.

The already explored environment is shown in white. The current visibility robot's region is shown in light gray (yellow), the environment regions which have not seen yet are shown in dark gray. The obstacles are shown in medium gray (blue). The robot is represented with a black disc, the omnidirectional sensor is a point over the robot's boundary. A small arrow over the robot is used to show the sensor direction $rt$. The landmark is represented by a medium gray disc (green). In the GNT, the primitive leaf nodes are shown as squares (yellow), the landmark node is a triangle (blue), and the non-primitive nodes are shown as circles (green).

Some snapshots of a simulation are presented in this section. Fig. 30 shows the robot executing a straight line motion primitive until a contact with $\partial E$ is detected, Fig. 31 shows when the landmark is totally visible from the omnidirectional sensor location.

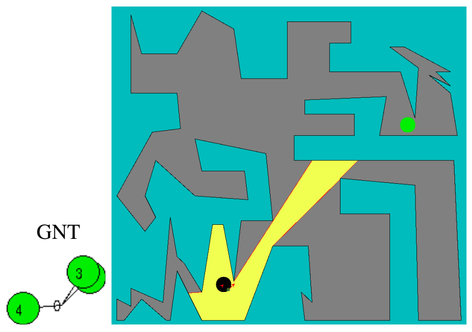


Fig. 30: The robot is executing the straight line motion primitive until a contact with $\partial E$ is detected. The corresponding GNT is shown.

Fig. 32 shows the GNT at the end of the execution of local exploration algorithm. Fig. 33 shows the moment when the robot has finished to explore the environment.

### 10.2 Experiments in a real robot

In all the experiments, we have used a Pioneer P3-DX robot, a differential drive system. The robot is modeled

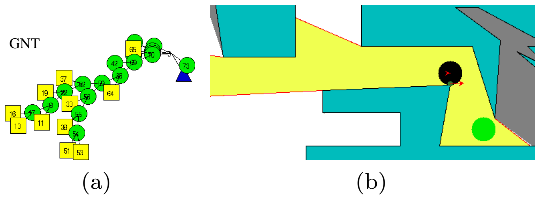


Fig. 31: The landmark is totally visible from the omnidirectional sensor location, hence it is encoded as a node child of the root in the GNT.

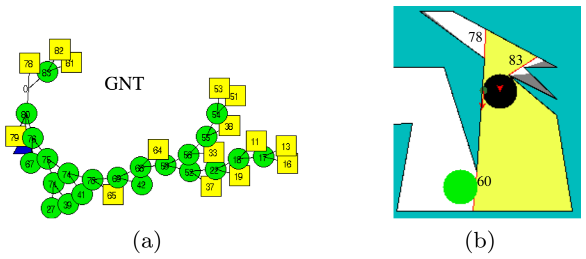


Fig. 32: GNT at the end of the execution of local exploration algorithm. The gaps 78 and 83 receive the primitive label, and node 83 propagates it to its offspring (leaf nodes 81 and 82).

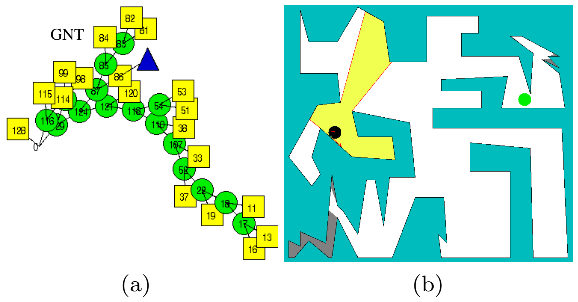


Fig. 33: The exploration task is finished, all the leave nodes in the GNT are primitives. Note that the robot has finished to explore the environment before traveling the whole environment boundary.

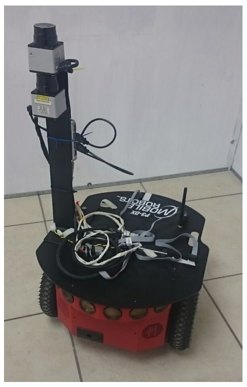
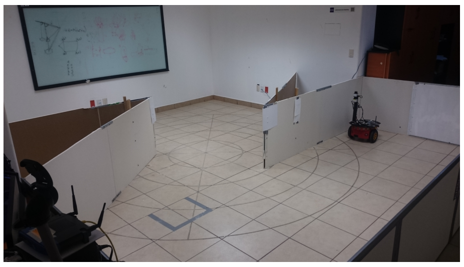


Fig. 34: The robot and the lasers.



(a) Lab: This environment is a drywall made structure in the CIMAT robotics lab, it is composed of 6 straight line segments, 8 concave corners and 2 convex corners

(b) Office: This environment is a CIMAT's office with regular furniture

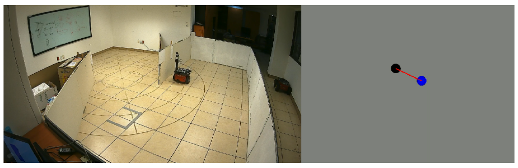
Fig. 35: Environments used for the experiments.

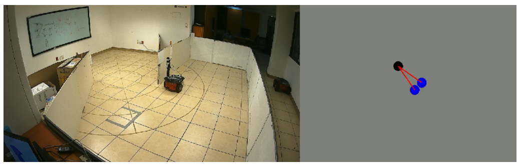
as a disc of radius of 0.2 $m$, it has a maximum translational velocity of 1.2 $m/s$ and a maximum rotational velocity of 5.236 $rad/s$. For the experiments, the maximum desired linear velocity $V_d$, was set to 0.33 $m/s$. The desired distance $d_d$ between the robot's center and the environment's boundary is set to 0.4 $m$. Thus, the distance between the robot's boundary and the environment is controlled to be 0.2 $m$.

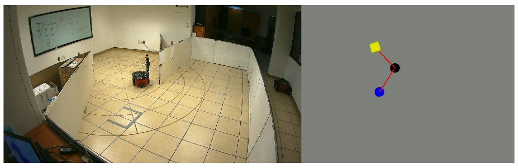
In our implementation all the algorithms run directly on the robot computer, which is a Pentium M at 1.8 Ghz with 1 GB of RAM. The operating system is Linux using some ROS functionalities, the control cycle runs to 12.5 Hz. The software is programmed in C++.

The ominidirectional sensor was implemented using two laser range finders Hokuyo model URG-04LX, which were mounted on the robot in opposite directions, see Fig. 34. Our current implementation of the gaps' detector is simple, it uses directly the raw data obtained with the lasers, to detect two consecutive angular measurements with a difference in distance larger than a given threshold, it was coded to test the whole method: automaton and controllers in the real robot.

The experiments were done in two different environments, see Figs. 35a and 35b. The main goal of the first experiment is to show the evolution of the GNT, a relevant aspect of this experiment is that it shows that the robot does not need to travel all the environment boundary to finish the exploration, as soon as all the leaf nodes in the GNT are primitive the exploration is finished. Indeed, as it was mentioned in Section 6, the main objective of the GNT in this approach is to indicate that the exploration task is finished, without the need to localize the robot. Fig. 36 shows the evolution of the GNT during the exploration task and the corresponding robot position in the environment.


(a) The robot starting the exploration and the initial GNT


(b) A split critical event


(c) A new gap appears as a primitive gap

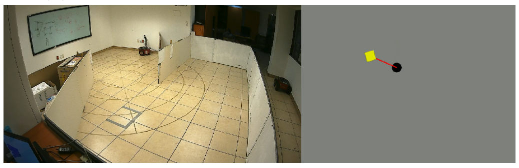
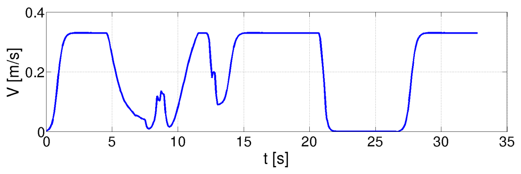

(d) The exploration ends, the only gap in the GNT is primitive

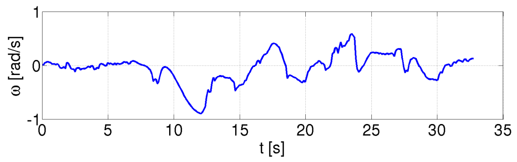
Fig. 36: Evolution of the GNT during the exploration.

Fig. 37 shows the linear and angular robot's velocities and the distance to the wall, while the robot was following the environment boundary until the stop condition given by the GNT was obtained. Fig. 37c indicates the distance to the wall and which controller is activated in each time interval. Notice that the robot's velocities are continous in spite of the switching between controllers.

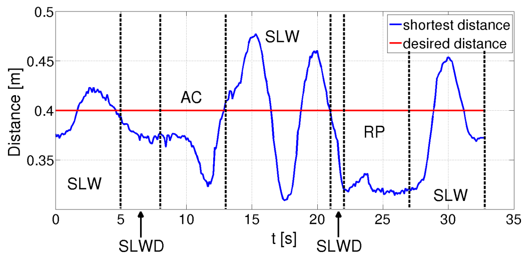
This second set of experiments (see Fig. 35b) had as a main objective to test the feedback-based controllers and the resulting wall following capability in a typical CIMAT's indoor environment. The resulting statistics are shown in Table 2. These statistics show the performance of the robot to follow the environment's boundary using the proposed strategy for 3 laps. For each lap



(a) Linear robot's velocity



(b) Angular robot's velocity



(c) Distance to the wall and activated controller. Time interval 0-5 seconds corresponds to SLW controller, time interval 5-8 sec. controller SLWD, time interval 8-13 sec. controller AC, time interval 13-21 sec. controller SLW, time interval 21-22 sec. controller SLWD, time interval 22-27 sec. controller RP and time interval 27-33 sec. corresponds to controller SLW.

Fig. 37: Linear and angular robot's velocities, the distance to the wall and the activated controller.

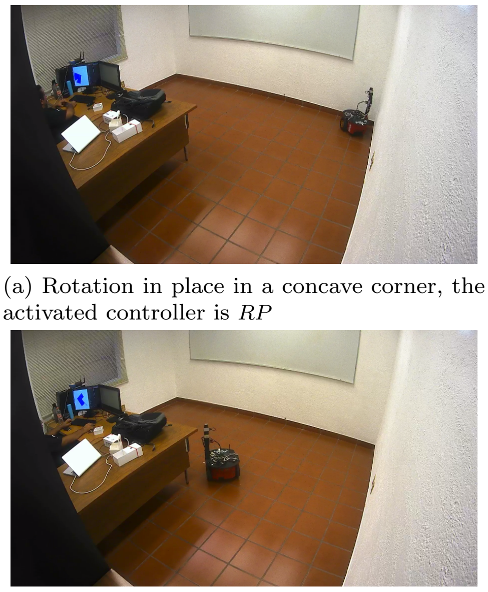
that the robot executed, we present the average distance between the robot's center and the environment's boundary, the corresponding standard deviation, the maximum and minimum values of this distance, as well as the time per lap taken by the robot to traveled the environment's boundary. In average, the robot follows the the environment boundary at a distance of 0.36 $m$, which implies an error of 4 $cm$ with respect to the setpoint distance $d_d = 0.4$ $m$. The maximum measured error during the whole motion was 13 $cm$. In average the time to complete a lap was 59.61 $s$ and the total time to travel the 3 laps was 178.84 $s$ and the perimenter of the enviroment is 16.7 m.

Fig. 38 shows an experiment in a CIMAT's office. Figs. 39, 40 and 41 show the robot linear and angular velocities while it was traveling the office.

*In the multi-media material, we have included a video*, in which two simulations and two experiments in the real robot are presented. The first experiment was done

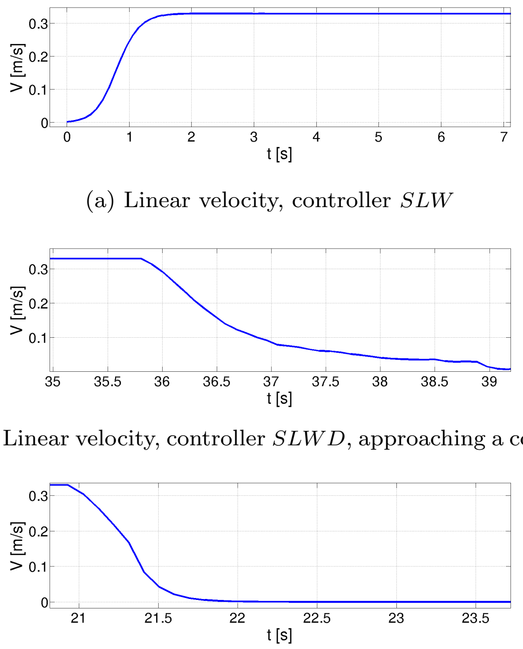
| Lap Number | Average Distance [m] | Std Dev [m] | Max. Distance [m] | Min. Distance [m] | Lap Time [sec] |
|---|---|---|---|---|---|
| 1 | 0.364 | 0.034 | 0.447 | 0.291 | 59.04 |
| 2 | 0.359 | 0.038 | 0.441 | 0.293 | 58.368 |
| 3 | 0.358 | 0.041 | 0.431 | 0.27 | 61.44 |

Table 2: Statistics of shortest distance measured from the robot center to the boundary of the environment: a CIMAT's office


(a) Rotation in place in a concave corner, the activated controller is $RP$

(b) Arc of circle around a convex corner, the activated controller is $AC$

Fig. 38: Experiments in the office.



(a) Linear velocity, controller $SLW$

(b) Linear velocity, controller $SLWD$, approaching a corner

(c) Linear velocity, controller $SLWD$, approaching an obstacle

Fig. 39: Linear velocities: Controllers $SLW$ and $SLWD$.

in the CIMAT's robotics lab and second one in a CIMAT's office.

Our experiments verified that the robot's velocities did not present discontinuities, in general the angular velocity is more noisy than the linear one, because of errors in the sensor reading that makes to vary the direction to the closest obstacle.

In order to test the pertinence of the automaton, we have implemented an alternative control scheme in which each controller was activated only depending on the observation, without considering the state in the machine. We have observed that the working controller changes very frequently given as a result discontinuous and noisy robot velocities. Even though that implementation was able to accomplish the task, to explore the same environments the robot took 10 times longer than when the states in automaton are used. The automaton reduces velocities discontinuities yielding smoother velocities because spurious observations are often rejected, in other words to change from and state to another–and consequently from a controller to another–, it is required that given the current state a specific observation (or set of observations) occurs.

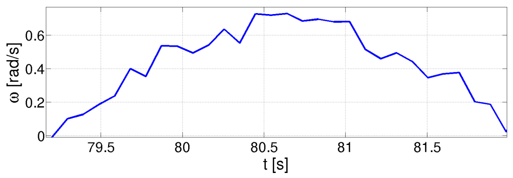
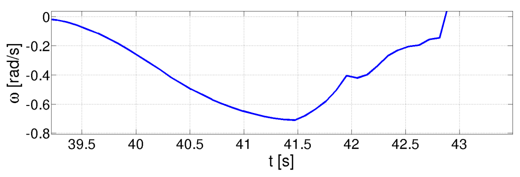
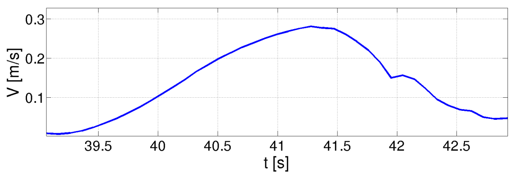


Fig. 40: Robot's angular velocity while the robot was rotating in place in a concave corner, controller $RP$.

Thus, it is less likely that both conditions are accomplished, the state and the observation. Hence, the automaton diminishes the change of controllers producing smoother robots velocities, because spurious observations are more frequently rejected, compared with a control scheme that changes the controller only based on the observation without taking into account the automaton state (internal robot state).



(a) Robot's angular velocity generated by controller $AC$, while the robot was executing an arc of circle rotating around a corner



(b) Robot linear velocity for the same controller, the robot was executing an arc of circle

Fig. 41: Angular and linear velocities: Controller $AC$.

Some limitations of our current implementation are the following: If the maximum desired linear velocity is changed then the controllers gains must be changed accordingly. A possibility to improve this limitation is to use adaptable gains depending on the maximum desired linear velocity. Spurious gaps sometimes instantaneously appear, a possible way to alleviate this problem is to filter the raw data. However, this issue has never prevented the robot to finish the exploration. Nevertheless, to deal with more complex environments, our current implementation of the gaps' detector should be improved either using well known filtering techniques on the raw laser data or even using robust line fitting methods to detect convex corners, making it more robust.

Based on the experimental results, we conclude that the theoretical modeling presented in the first part of the paper can be adapted to deal with imperfect real laser readings and imperfect execution of motion primitives. Our current implementation works properly for the tested environments.

## 11 Conclusions

This paper addressed the problem of exploring an unknown environment, using a differential drive robot with the shape of a disc. To explore the environment, the robot follows the environment boundary. The robot is equipped with typic robotic sensors and the proposed exploration strategy does not requiere to localize the robot.

The exploration problem addressed in this paper is more challenging than the case of a point robot because visibility information does not provide collision free paths in the configuration space. In this paper an exploration strategy is proposed. This exploration strategy is modeled as a Moore machine, and it guarantees exploring all the environment or the largest possible region of it. The robot is able to find a landmark or declare that an exploration strategy for this objective does not exist. A motion policy based on sensor feedback is also proposed.

We have proposed a practical hybrid control scheme that allows the robot's commands to be imperfect, and to deal with the robot dynamics (i.e. velocities variations). Besides, our control scheme aims to maintain the continuity of angular and linear velocities of the robot in spite of the switching between controllers. The main originality of the proposed approach with respect to previous work on wall following is that in this approach, the FSM constraints the possible states transitions filtering spurious observation due to noisy sensor readings. We underline that in this approach the planning stage corresponds to the design of the FSM and it is done prior to execution, once this is done, for any execution instance and for any different environment, the method is reactive, it just relates observations to controls. All the proposed algorithms have been implemented and both simulations and experiments in a real robot are presented to validate the approach. The experimental results in a real robot have matched with the proposed modeling.

In this work, we only considered polygonal environments and sensors with not limited range. Since a gap encodes the frontier between known and unknown space, we believe that a gap can also be used to encode the frontier generated by the sensor range, at the moment that a wall enters within the sensor range the gap would disappear. Therefore, we think that the gaps modeling can be extended to consider a sensor with limited range. We also think that the wall following capability can be extended to other types of environments different from polygons. Perhaps some approximations of the environments can be done using simple curves or even line segments. If so, it would be possible to find some controls to follow such curves. Nevertheless, such extensions would need a detailed analysis to obtain a formal modeling with certain guarantees, these extensions are left for future work. We would also like to extend the approach for the execution based on feedback of any type of trajectories and not just to wall following.

## References

1. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.
2. J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.
3. J. Minguez and L. Montano. Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.
4. J.-P. Laumond, P.E. Jacobs, M. Taïx, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. on Robotics and Automation*, 10(5):577–593, 1994.
5. A. Bicchi, G. Casalino, and C. Santilli. Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. *J. of Intelligent Robots Systems*, 16(4):387–405, 1996.
6. J.-B.. Hayet, H. Carlos, C. Esteves, and R. Murrieta-Cid. Motion planning for maintaining landmarks visibility with a differential drive robot. *Robotics and Autonomous Systems*, 4(62):456–473, 2014.
7. B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47–63, 1991.
8. B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, USA, 1997.
9. F. Amigoni and V. Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.
10. M. Juliá, A. Gil, and O. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.
11. B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.
12. A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. *Advanced Robotics*, 23(12-13):1533–1560, 2009.
13. M. Katsev, A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle. Mapping and pursuit-evasion strategies for a simple wall-following robot. *IEEE Transactions on Robotics*, 27(1):113–128, 2011.
14. Y. Landa and R. Tsai. Visibility of point clouds and exploratory path planning in unknown environments. *Communications in Mathematical Sciences*, 6(4):881–913, 2008.
15. L. Murphy and P. Newman. Using incomplete online metric maps for topological exploration with the gap navigation tree. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 2792–2797, 2008.
16. C.J. Taylor and D. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Transactions on Robotics and Automation*, 14(3):417–426, 1998.
17. A. Elfes. Sonar-based real world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–264, 1987.
18. R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 661–666, Barcelona, Spain, 2005.
19. H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, 1999.
20. A. Makarenko, B. Williams, F. Bourgault, and H. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2002, Lausanne, Switzerland*, pages 534–539, 2002.
21. Y.-A.. Gidhar and G. Dudek. Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring. *Autonomous Robots*, 40(7):1267–1278, 2016.
22. R. Sim and G. Dudek. Effective exploration strategies for the construction of visual maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2003*, pages 3224–3231, Las Vegas, Nevada, USA, 2003.
23. H. González-Banos and J.-C. Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21(10-11):829–848, 2002.
24. G. Oriolo, M. Vendittelli, L. Freda, and G. Troso. The srt method: randomized strategies for exploration. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2004*, pages 4688–4694 Vol.5, New Orleans, LA, USA, 2004.
25. S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 1998*, pages 1546–1551, Leuven, Belgium, 1998.
26. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
27. H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.
28. F. Amigoni, S. Gasparini, and M. Gini. Building segment-based maps without pose information. *Proceedings of the IEEE*, 94(7):1340–1359, 2006.
29. A. Kolling and S. Carpin. Extracting surveillance graphs from robot maps. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 11–19, 2008.
30. S. M. LaValle. Sensing and filtering: A fresh perspective based on preimages and information spaces. In *Foundations and Trends in Robotics Series*. Now Publishers, Delft, The Netherlands, 2012.
31. R. Lopez-Padilla, R. Murrieta-Cid, and S.M. LaValle. Optimal gap navigation for a disc robot. In E. Frazzoli et al., editor, *Proc. of the Tenth Workshop on the Algorithmic Foundations of Robotics: Springer Tracts in Advanced Robotics*, pages 123–138. Springer, 2013.
32. R. Lopez-Padilla, R. Murrieta-Cid, I. Becerra, G. Laguna, and S. M. LaValle. Optimal navigation for a differential drive disc robot: A game against the polygonal environment. *J. of Intelligent Robotic Systems*, 89(1-2):211–250, 2018.
33. E. Bicho. Detecting, representing and following walls based on low-level distance sensors. In *Proc. of the Int. Symposium on Neural Computation*, Berlin, Germany, 2000.
34. M. Toibero, F. Roberti, and R. Carelli. Stable contour-following control of wheeled mobile robots. *Robotica*, 27(1):1–12, 2009.
35. M. Toibero, F. Roberti, R. Carelli, and P. Fiorini. Switching control approach for stable navigation of mobile

robots in unknown environments. *Robotics and Computer-Integrated Manufacturing*, 27(2):558–568, 2011.

36. A. De and D. E. Koditschek. Toward dynamical sensor management for reactive wall-following. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2013*, pages 2400–2406, Karlsruhe, Germany, 2013.
37. A. G. Lamperski, O. Y. Loh, B. L. Kutscher, and N. J. Cowan. Dynamical wall following for a wheeled robot using a passive tactile sensor. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 3838–3843, Barcelona, Spain, 2005.
38. G. Laguna, R. Murrieta-Cid, H.M. Becerra, R. Lopez-Padilla, and S.M. LaValle. Exploration of an unknown environment with a differential drive disc robot. In *Proc of IEEE Int. Conf. on Robotics and Automation*, pages 2527–2533, 2014.
39. J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Pearson Education, 2000.
40. W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1994.