Exploration of an Unknown Environment with a Differential Drive Disc Robot

Guillermo Laguna†, Rafael Murrieta-Cid†, Hector M. Becerra†
Rigoberto Lopez-Padilla† and Steven M. LaValle○
† Centro de Investigación en Matemáticas (CIMAT), Guanajuato 36000, Mexico
Email: {glaguna, murrieta, hector.becerra, rigolpz}@cimat.mx
○ University of Illinois, Urbana IL 61801, USA
Email: lavalle@uiuc.edu

***Abstract*—This paper addresses the problem of exploring an unknown, planar, polygonal and simply connected environment. A saliency object (i.e. a landmark) is located in the environment. The collision-free subset of the robot's configuration space is simply connected or it might have several connected components. The robot is a differential drive system shaped as a disc. The robot has limited sensing, namely it is incapable of measuring any distance or angle, or performing self localization. The exploration problem consists in discovering the environment with the robot's sensor. To solve this problem, a motion policy is developed based on simple sensor feedback and a complete exploration strategy is represented as a Moore Machine. The proposed exploration strategy guarantees that the robot will discover the largest possible region of the environment. Consequently, the robot will find the landmark or declare that an exploration strategy to find it does not exist.**

I. Introduction

The task of exploring unknown planar environments has been treated in many previous works [1]–[6]; 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 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 true performance. A natural step forward, and more realistic, is to consider the robot as a nonzero size entity. A disc shape is the most simple one. The robot's size represents additional constraints in the configuration space, specifically, a growth of the obstacle's size in a measure related to the robot's radius. 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: the concept of visibility is equal to the concept of reachability for a point robot. It means that if the robot can see certain environment place, that place is also reachable for the robot. However, this property is not necessarily true for a disc robot. Indeed, the configuration space is not observable, the robot cannot measure it. To solve an exploration problem, the disc robot must be able to infer information of the configuration space from the workspace. In this paper, we present a novel exploration strategy of an unknown, planar polygonal environment using a disc robot.

A. Related Work

Many works have addressed the problem of exploring an unknown environment to build a representation of it. Given strong sensors and good odometry, standard SLAM approaches [7], [8] provide a geometric map of the environment. In [9], a method is proposed for building a global geometric map without precise robot localization by registering scans collected by laser range. A different map building approach is the occupancy grid [6], which represents the environment as a 2D array, instead of using geometric primitives (e.g. line segments). This algorithm is useful for obstacle avoidance and planning purposes, but it has the disadvantage of being difficult to handle large environments. Another type of environment's representations are the topological maps in the form of graphs. In these graphs the nodes represent environment places and the edges represent adjacency. The problem of exploring an unknown environment for searching of one or more recognizable targets is considered in [5]. This method uses 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 [4]. 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. The GNT is dynamically built based on critical events detected with the robot's sensors. 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 [3]. This improves robustness given that the model deals with noise presence in the sensor's measurements. The GNT was also extended to clouds of points models in [2]. A larger family of gap sensors is described in [10]. The GNT approach has been extended to a disc-shaped differential-drive robot placed into an unknown, simply connected polygonal region in [11]. The main result in that work is a navigation strategy that drives the robot to optimally navigate toward a landmark in the region. However, in [11], an exploration strategy to learn the GNT and encoding a landmark within it, has not been developed. In this paper, we address that ex-

ploration problem. We propose an exploration strategy where we do not allow the robot to localize itself or to build a geometric map, the strategy is based on *wall following* and not on chasing gaps in contrast to [4]. A wall following approach has been proposed for exploration of a simply connected environment with a point robot in [1]. A data structure called cut ordering is proposed in that work. The point robot is able to identify whether the robot at its current location is touching an environment wall, a convex vertex[1], a reflex vertex[2] or whether it lies on the interior of the environment. Once the cut ordering representation is built, it is used to address a pursuit/evasion problem. A difference with respect to [1] is that in this work the robot is no longer a point.

The main contribution of this paper is a complete exploration strategy that reports whether all the environment has been seen or the largest possible region has been seen. A strategy is proposed to deal with cases where no accessible places are found. These cases represent a challenge given that a portion of the environment might not be visible to the robot and the strategy must ensure to see as much as possible. The environment is represented in an efficient data structure, the GNT. Additionally, the proposed strategy is relatively easy to implement and compact, in such a way that it is represented as a Moore machine.

II. Problem Statement

The robot has the shape of a disc with radius $\boldsymbol{r}$ 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 and the boundary ∂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[3]. A saliency object (i.e. a landmark) is located in the environment. The robot is unable to localize itself at *any reference frame*, and has limited sensing capabilities, namely it is incapable of *measuring any distance or angle*.

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.

III. Sensing Model

We use the Gap Navigation Tree [4], [11] to represent the environment. The GNT is a efficient data structure that dynamically changes according to some critical events until the whole environment has been discovered.

[1]A convex vertex is a polygon vertex of an internal angle smaller than π.

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

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

A. 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 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. 1). The omnidirectional sensor is also able to track the direction lt or rt depending whether the sensor is placed over lp or rp. The sensor might be located at rp or lp.

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 refereed 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 [4]. The complete GNT built process is defined with just this four events. 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. 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 as statements $[g_1, ..., g_k] = [g_2, ..., g_k, g_1]$ can be made. If x lies in ∂E then part of the sensor's view is obstructed by the boundary, and a linear ordering of gaps is obtained. In summary the gap sensor is able to detect and order the gap directions, the preferential direction and a visibility obstruction if the sensor is in contact with ∂E. This behavior allows the sensor to detect events such as alignments between the preferential directions rt or lt and any gap, or between one of the two preferential directions lt or rt and the wall (∂E) that is in contact with the omnidirectional sensor.

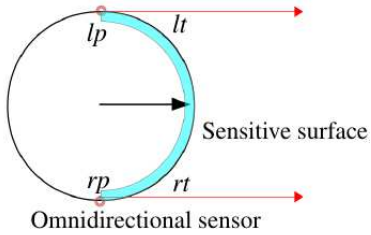

Fig. 1: Representation of the robot's sensors.

Let Λ 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 Λ is visible at least partially from the location of the omnidirection sensor. Once the landmark was recognized, it is encoded as a special GNT node connected to the corresponding node associated with the gap occluding it, or if the whole landmark is visible from the current position then it is directly connected to the root. The landmark is encoded as a special node because it is not encoding a gap, it cannot suffer the GNT critical events.

B. A Tactile Bumper

The robot's frontal periphery is contact sensitive (See Fig. 1). In a real robot it could be implemented, for instance with a piezoelectric sensor. The sensitive surface model is able to distinguish whether there exists contact on a single point or more than one, the sensor also distinguishes whether the point rp or lp is in contact with a wall. The particular case of both points rp and lp being simultaneously in contact is not considered, it only would happen in a narrow corridor, of exactly the same width as the robot, and that scenario is considered a degenerated case.

C. The observation vector

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

Six binary sensor observations constitute the observation vector: (1: lp) the robot is touching ∂E with point lp. (2: rp) the robot is touching ∂E with point rp. (3: sc) the robot is touching ∂E with a single point within the sensitive surface (this point might be either lp, rp or any other point within the sensitive surface). (4: bc) the robot is touching ∂E with two or more points within the sensitive surface (one of them can be either lp or rp). (5: $aligned$) direction rt is aligned with the edge of the polygonal region that point rp is touching, or point rp is touching a reflex vertex and the preferential direction rt is aligned with the first polygonal edge, measured in clockwise sense starting from direction rt; or direction lt is aligned with the edge of the polygonal region that point lp is touching, or point lp is touching a reflex vertex and the preferential direction lt is aligned with the first polygonal edge, measured in counterclockwise sense starting from direction lt; (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 possible 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:

- 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 there is not contact sensed.
- ye_2 *Single contact* with rp: The omnidirectional sensor is positioned at rp, there is single contact detected at that point, and the preferential direction rt is aligned with the polygonal edge that point rp is touching.
- ye_3 *Single contact* with lp: This observation is analogous to Single contact rp, it is the left symmetric case.
- 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 sensitive surface is touching more than one point of ∂E, the contact might be with any combination of edges or reflex vertices of E.
- ye_5 *Multicontact, sensor at* lp: This observation is analogous to Multicontact rp, it is the left symmetric case.
- 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 preferential direction rt is not aligned with the first polygonal edge, measured in clockwise sense starting from direction rt.
- 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 preferential direction lt is not aligned with the first polygonal edge, measured in counterclockwise sense starting from the reflex vertex.
- 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 ∂E with a single point different to lp.
- 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.

IV. 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. 2.

Let the angular velocity of the right and left wheels be ω_l and ω_r respectively, with ω_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 wall following. If the omnidirectional sensor is placed at rp then the robot follows the environment's boundary ∂E in counterclockwise sense, and if the sensor is placed at lp then the robot follows ∂E in clockwise sense.

V. The Exploration Automaton

A finite-state machine (FSM) is defined as a mathematical model of computation, it is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time, it can change from one state

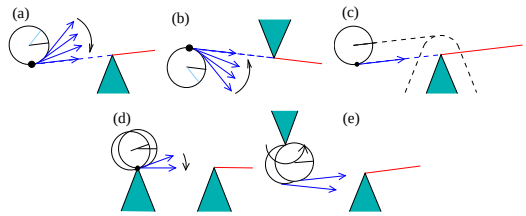

Fig. 2: The motion primitives: (a) Clockwise rotation in place, (b) Counterclockwise rotation in place, (c) Straight line motion, (d) Clockwise rotation w.r.t. rp, (e) Counterclockwise rotation w.r.t. lp.

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, these outputs depend exclusively of the current state and they do not take into account the input. 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 V-A). 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 [4], 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 stop exploration condition 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, this algorithm is part of the exploration strategy.

M is formally defined as a sextuple (Σ,S,s_0,δ,Γ,ω), where:

- Σ is the input alphabet (a finite, non-empty set of symbols). In M, Σ is defined by both the observation vector ye_i and an additional query input given by the GNT, needed for determining whether the stop condition is met.
- S is a finite, non-empty set of states, every state represents the selection and execution of a robot's motion primitive with the exception of two states: the initial state when the robot is not executing a primitive yet and the end state, in which the robot has finished the exploration task.
- s_0 is the initial state, in which the exploration task begins.
- δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$. In M, given an observation and the current state, δ defines which will be the new state. It is important to note that δ is a partial function, for example, $\delta(q, x)$ does not have to be defined for every combination of $q \in S$ and $x \in \Sigma$. Actually the set of allowed combinations is well established in the motion policy of Section V-A.
- Γ is the output alphabet (a finite set of symbols), it is defined as the signals given to the motors for executing a given control u.
- ω is the output function: $\omega : S \rightarrow \Gamma$, each state provides an specific output signal defined on Γ.

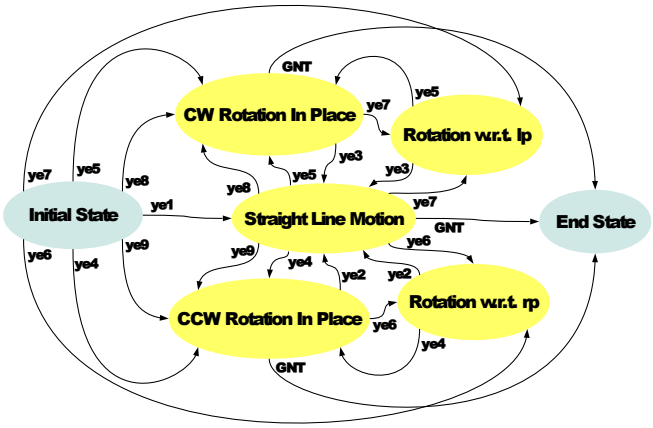

Fig. 3: The finite-state machine that represents the exploration strategy.

A graphical representation of M is shown in Fig. 3. 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, the task has been achieved, 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 IV. All the links in Fig. 3 are labeled with the corresponding observations defined in Section III, with the exception of the GNT link, it represents a query 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. 3). Given that, the GNT might change because the occurrence of critical events, while the robot is execution 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 updates the labels of the gaps in the GNT.

A. 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 γ 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 $\gamma\left(ye_i\right) = (\omega_l, \omega_r) = u_j$. 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.

B. 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 ∂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 $ye_5^L = (1,0,0,1,x,0)$. They are special cases of ye_4 and ye_5 observations respectively, when the corresponding observation happens (depending if the point is rp or lp) the algorithm is triggered. 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 algorithm uses the property that states *the change from cyclic to linear gap ordering when the gap sensor is touching the wall according to the model detailed in [4]*. *Local exploration* uses linear lists. Those lists are *init-list* and *end-list*. *init-list* contains the current gaps read by the sensor and the preferential direction (rt or lt depending if the contact point is rp or lp). *init-list* tracks the changes over the gaps due to critical events, and the location of preferential direction (rt or lt) in the list. *end-list* contains gaps read by the sensor and the preferential direction after the motion primitive ends. The order of the gaps in *end-list* is different from the order in *init-list*, because it is built at a different position. $init_f$ is the first element of *init-list*, $init_l$ is the last element of *init-list*, $init_{rt}$ is the element containing the rt direction and $init_{lt}$ is the element containing the lt direction. The equivalent elements in *end-list* have an analogous nomenclature. G_1 and G_2 are auxiliary lists containing specific subsets of *init-list* and *end-list* respectively. $G_\cap$ includes the gaps that must propagate the primitive label to their offspring on the GNT.

Lemma 1: The exploration strategy modeled by the Moore Machine M guarantees that all leaf gaps (i.e. gaps encoded as leaf nodes in the GNT) are labeled as primitive gaps, executing Algorithm 1 (local exploration algorithm) at each time that observation y_4^R or y_5^L occurs. Algorithm 1 labels as primitive gaps, the gaps that do not disappear, given that those gaps are generated by reflex vertices located within an unreachable region.

Proof: The gaps that do not disappear are handled by Algorithm 1. If the robot is touching ∂E with point rp then

Algorithm 1 Local Exploration Algorithm

Input: GNT, current observation: ye_i.
Output: updated GNT.
if $rp =$**true then**
  1. *init-list* $\leftarrow$ 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 $(ye_i \neq ye_2)$ **or** $(ye_i \neq ye_6)$ **do**
      if *GNT-event* $=$ **true then**
        if *critic-event* $\neq$ *gap-appear* **then**
          2. Apply the update suffered by the root's children nodes of the GNT to the corresponding gaps in *init-list*;
        end if
      end if
      3. Update the position of the rt direction in *init-list* according to the current angular counterclockwise order in the sensor reading;
    end while
  end if
  4. *end-list* $\leftarrow$ Current gaps and rt direction starting from the sensor's obstructed visibility region following a counterclockwise order;
else if $lp =$**true then**
  5. *init-list* $\leftarrow$ Current gaps and lt direction starting from the sensor's obstructed visibility region following a clockwise order;
  if $ye_i = ye_5^L$ (u_2 is executed) **then**
    while $(ye_i \neq ye_3)$ **or** $(ye_i \neq ye_7)$ **do**
      if *GNT-event* $=$ **true then**
        if *critic-event* $\neq$ *gap-appear* **then**
          6. Apply the update experienced by the root's children nodes of the GNT to the corresponding gaps in *init-list*;
        end if
      end if
      7. Update the position of the lt direction in *init-list* according to the current angular clockwise order in the sensor reading;
    end while
  end if
  8. *end-list* $\leftarrow$ Current gaps and lt direction starting from the sensor's obstructed visibility region following a clockwise order;
end if
9. $G_1 \leftarrow \{x \in$ *init-list* $\mid init_f < x < init_{rt}\}$;
10. $G_2 \leftarrow \{x \in$ *end-list* $\mid end_{rt} < x < end_l\}$;
11. $G_\cap \leftarrow G_1 \wedge G_2$;
for every gap $g_i \in G_\cap$ **do**
  12. Label node g_i in the GNT as a primitive node;
  13. Propagate the primitive label to the offspring of g_i;
end for

the G_1 list includes all the gaps belonging to the open interval between the rp point starting position at the beginning of Algorithm 1 and the rt direction at the end of Algorithm 1. In this interval the order of the gaps is established in counter-clockwise sense. Moreover, G_2 list includes all gaps belonging to the open interval between rt direction at the end of the algorithm and the rp position at the end of the algorithm. In this other interval the order of the gaps is also established in counterclockwise sense. The intersection between G_1 and G_2 includes only the gaps that lie between the original rp position and the current one in counterclockwise sense. Those gaps are generated by reflex vertices located within the unreachable region. Observation y_4^R or y_5^L detects an unreachable region. The region is unreachable because the robot's bumper has touched ∂E at two points. During the robot rotation in place, the omnidirectional sensor moves from a point touching ∂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 the offspring of them in $G_\cap$. Each time that observation y_4^R or y_5^L 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. ■

When the robot is touching ∂E with point lp and the omnidirectional sensor is placed at lp the proof is analogous, it is just the symmetric case.

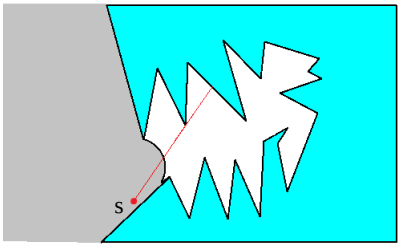

Fig. 4: R_{na} is shown in white and in region R_a in dark grey. The regions are divided by the arc of circle trajectory followed by the omnidirectional sensor during a robot's rotation in place. The figure also shows a source s with a ray of light which goes from R_a to R_{na}

Lemma 2: The robot is capable of covering (observing) the largest possible portion of the environment, by executing local exploration algorithm at each time that observation y_4^R or y_5^L occurs.

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 is the arc of circle described by the rotation, which depends of the robot's radius. The omnidirectional sensor is unable of penetrating deeper in the unreachable region due to the space configuration restrictions, therefore, the arc of circle is clearly the boundary between both regions. Refer to Fig. 4. It is clear that every ray of light emerging from any source $s \in R_a$ which is able to intersect R_{na} must cross the regions' boundary as seen in Fig. 4. 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}. Each time that observation y_4^R or y_5^L occurs the local exploration algorithm is executed. The result follows. ■

Theorem 1: The exploration strategy modeled as a Moore Machine M guarantees exploring all the environment or the largest possible region of it, and it also guarantees that the exploration task terminates. Additionally, the robot is able to find the landmark or to declare that an exploration strategy to find the landmark does not exist, for the connected component of the collision-free subset of the configuration space $\mathcal{C}$ where the robot lies.

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 1 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 2 guarantees that the robot discovers the largest possible region of the environment. Hence, if the collision free sub-set of the configuration space $\mathcal{C}$ is simply connected then the landmark is found. If the collision free sub-set of the configuration space $\mathcal{C}$ has several connected components then the landmark might or might not be found. Again, by Lemma 2 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. ■

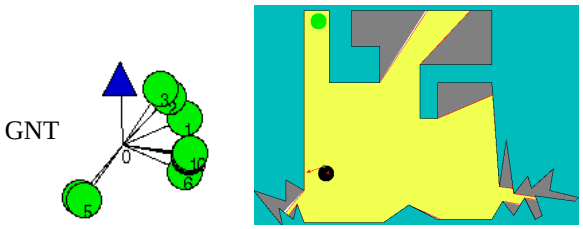

Fig. 5: The robot is executing the straight line motion primitive. The corresponding GNT is shown, the circles represent non primitive nodes (maintaining the same orientation that the gaps they represent) and the triangle is the encoded landmark.

VI. Implementation

The whole method has been implemented and simulations' results are included. 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 preferential 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). Fig. 5 shows the robot executing control u_1 that yields a straight line motion primitive, the robot initial position lies in the interior of E, it moves forward until a contact with ∂E is detected, it is the only case where the robot does not follow the environment's boundary. Fig. 6 shows a partial landmark occlusion. It occurs when the robot is following an environment's edge, then the landmark is encoded to the corresponding gap in the GNT. In Appendix A, we present the case in which the local exploration algorithm is applied with the omnidirectional sensor positioned over rp, the robot is facing a narrow gate impossible to cross, so there are gaps that do not disappear but its corresponding GNT nodes must be properly labeled as primitive nodes at the end of local exploration. In the multi-media material, we have included a video, in which a complete exploration simulation is presented. This simulation shows a larger and more complex map than the one presented in the paper.

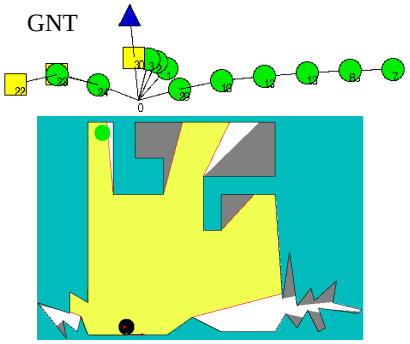

Fig. 6: The landmark is partially occluded, the current GNT is presented and it shows that the landmark is encoded in the GNT (the primitive leaf nodes are shown like squares).

VII. Conclusions

This paper addressed the problem of exploring an unknown environment, using a differential drive robot with the shape of a disc. The robot is equipped with simple sensors and it is unable to build precise geometric maps or localize itself in any Euclidean frame. 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 than an exploration strategy for this objective does not exist. A motion policy based on sensor feedback is also proposed. All the proposed algorithms have been implemented and simulation results are presented.

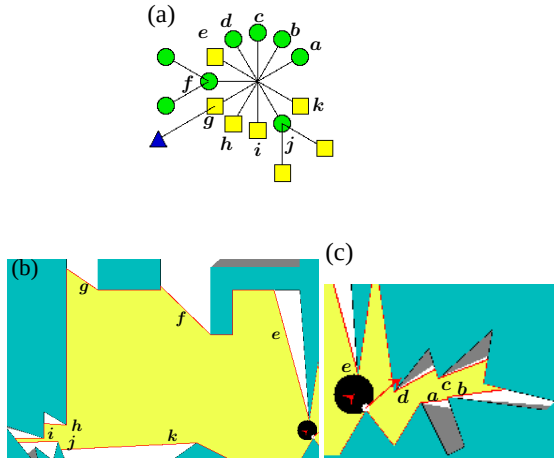

Fig. 7: (a) Current GNT at the beginning of local exploration, (b) gaps out of the unreachable region, (c) gaps within the unreachable region, those gaps must be labeled as primitive ones, after executing local exploration algorithm.

Appendix A
Local exploration example

To clarify the local exploration algorithm, we illustrate the method with an example. Fig. 7 shows the time when the local exploration algorithm starts after observation ye_4^R is met. For labeling gaps, we use consecutive characters (alphabetic order). First, the initial linear list is created, *init-list*= $\{a, b, c, d, \uparrow, e, f, g, h, i, j, k\}$, where the characters represent the current gaps and $\uparrow$ represents the preferential direction rt. Due to lack of space, only some of the critical events are described in detail while others are just mentioned.

Fig. 8 shows the first critical event during the motion primitive execution, it is a merge between gaps c and d yielding the new gap l [4]. The linear list is then, *init-list*= $\{a, b, l, \uparrow, e, f, g, h, i, j, k\}$.

After the first merge, three other merges happen: gaps b and l merge into gap m, gaps e and f (which are outside the unreachable region) merge into gap n and gaps a and m merge into gap o. Thus, after these three merging events, the linear list is *init-list*= $\{o, \uparrow, n, g, h, i, j, k\}$.

Fig. 9 shows the next critical event, it is a split of gap n [5], the current linear list is *init-list*= $\{o, e, f, g, \uparrow, h, i, j, k\}$,

[4] According to the GNT evolution, nodes c and d become children nodes of the new node l which is connected to the root, all merge events have the same behavior.

[5] When a split event happens to a no leaf GNT node, the resulting gaps are its children nodes, so the gaps obtained are e and f.

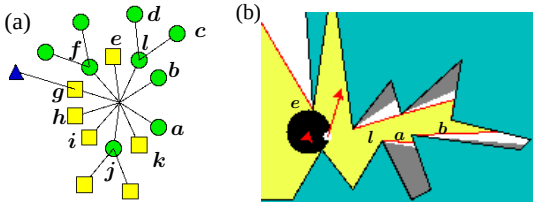

Fig. 8: (a) GNT with c and d as the children nodes of l due to the merge event, (b) the new gap l.

it is important to notice that the linear order between the preferential direction and the gaps also changed.

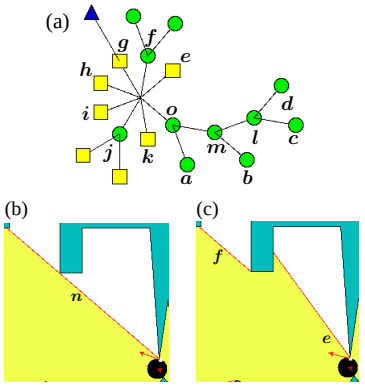

Fig. 9: (a) GNT with resulting gaps e and f after gap n splits (they were its children nodes), (b) the instant before n splits, (c) the resulting gaps e and f after the split event.

Fig. 10 shows when gap e disappears, the current linear list is *init-list*$= \{o, f, g, \uparrow, h, i, j, k\}$.

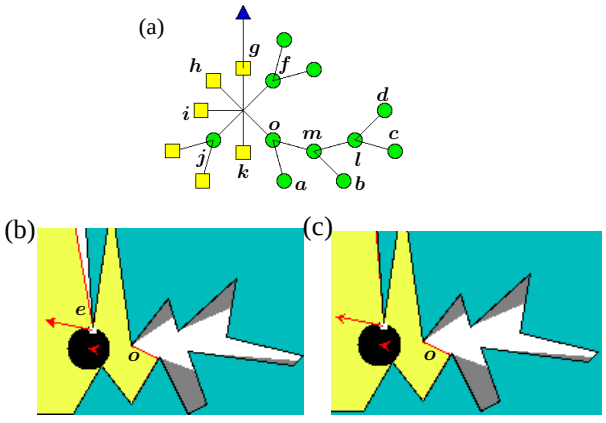

Fig. 10: (a) The GNT after gap e has disappeared, (b) the gaps within the unreachable region and gap e before disappearing, (c) the gaps after the disappear event.

Then a new gap p appears, *init-list* remains unchanged during gap appear events (recall that appearances of gaps are not considered because they already have the primitive label, therefore they do not represent an issue).

Finally, Fig. 11 shows the gaps when the algorithm 1 ends and the GNT leaves are correctly labeled, the final list is *end-list*$= \{f, g, \uparrow, h, i, j, k, o, p\}$. According to local exploration algorithm the auxiliary lists are: $G_1 = \{o, f, g\}$ and $G_2 = \{h, i, j, k, o, p\}$, the intersection list is then $G_\cap = \{o\}$, so the gap o receives the *primitive* label and propagates it to its offspring (leaf nodes a,b,c and d).

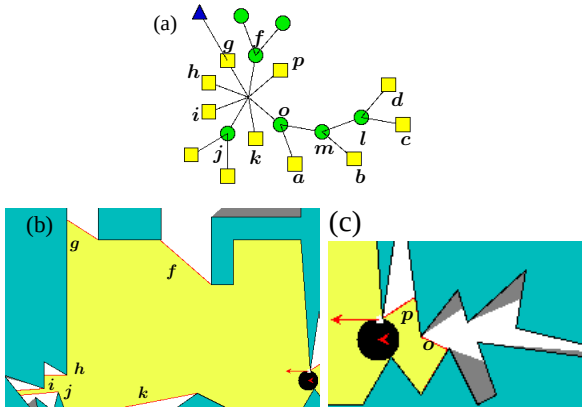

Fig. 11: (a) The GNT at the end of local exploration when the primitive label has been propagated to the leaf nodes representing unreachable gaps, (b) the gaps outside of the unreachable region, (c) the gaps within the unreachable region that have been correctly labeled as primitive ones.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] A. Elfes. Sonar-based real world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–264, 1987.

[7] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[9] F. Amigoni, S. Gasparini, and M. Gini. Building segment-based maps without pose information. *Proceedings of the IEEE*, 94(7):1340–1359, 2006.

[10] 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.

[11] 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. 2013.