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
○ University of Illinois, Urbana IL 61801, USA
Email: {glaguna, murrieta, hector.becerra, rigolpz}@cimat.mx

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 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 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 place within the environment, 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. This algorithm is useful for obstacle avoidance and planning purposes, but it has the disadvantage that it has difficulties to deal with large environments. Another type of environment’s representations are the topological maps in the form of graphs. 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]. 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 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
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 \(\text{lp}\) and \(\text{rp}\). 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 \(\text{rp}\) is called \(\text{rt}\). The direction of the line tangent to the robot’s boundary at \(\text{lp}\) is called \(\text{lt}\) (See Fig. 1). The omnidirectional sensor is also able to track the direction \(\text{lt}\) or \(\text{rt}\) depending whether the sensor is placed over \(\text{lp}\) or \(\text{rp}\). The sensor might be located at \(\text{rp}\) or \(\text{lp}\).

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 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 [4]. The complete GNT built process is defined with just these 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 that statements as \([g_1, ..., g_k] = [g_2, ..., g_k, g_1]\) can be made. 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. In summary the gap sensor is able to detect and order the gap directions, the direction \(\text{rt}\) or \(\text{lt}\) 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 preferential directions \(\text{rt}\) or \(\text{lt}\) and any gap, or between one of the two preferential directions \(\text{lt}\) or \(\text{rt}\) and the wall (\(\partial E\)) that is in contact with the omnidirectional sensor.

![Fig. 1: Representation of the robot’s sensors.](image)

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. Once the landmark was recognized, it is encoded as a special GNT node connected to the corresponding omnidirectional sensor. The landmark is encoded as a special node in the GNT, because it is not a gap, it cannot suffer critical events.

---

\(^1\)A convex vertex is a polygon vertex of an internal angle smaller than \(\pi\).

\(^2\)A reflex vertex is a polygon vertex of an internal angle greater than \(\pi\).

\(^3\)Note that this is the configuration space for a translating disc rather than for a rigid body because of rotational symmetry.
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 degenerate 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 a 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 = {lp, rp, sc, bc, aligned, o}

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

\[ ye_1 = (0, 0, 0, 0, x, x) \]
\[ ye_2 = (0, 1, 1, 0, 1, 1) \]
\[ ye_3 = (1, 0, 1, 0, 1, 0) \]
\[ ye_4 = (x, x, 0, 1, x, 1) \]
\[ ye_5 = (x, x, 0, 1, x, 0) \]
\[ ye_6 = (0, 1, 1, 0, 0, 1) \]
\[ ye_7 = (1, 0, 1, 0, 0, 0) \]
\[ ye_8 = (0, 0, 1, 0, x, 0) \]
\[ ye_9 = (0, 0, 1, 0, x, 1) \]

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 no contact with ∂E is 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, ω_r ∈ {−1, 0, 1}. The robot’s controls are defined by the vector u = {ω_l, ω_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 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 on 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 leaf nodes have a label called primitive). 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 $(\Sigma, S, s_0, \delta, \Gamma, \omega)$, where:

- $\Sigma$ is the input alphabet (a finite, non-empty set of symbols). In $M$, $\Sigma$ 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.
- $\delta$ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$. In $M$, given an observation and the current state, $\delta$ defines which will be the new state. It is important to note that $\delta$ 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.
- $\Gamma$ is the output alphabet (a finite set of symbols), it is defined as the signals given to the motors for executing a given control $u$.
- $\omega$ is the output function: $\omega : S \rightarrow \Gamma$, each state provides a specific output signal defined on $\Gamma$.

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 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 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 \( \gamma \) 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 \( \gamma(\mathbf{ye}_i) = (\omega_l, \omega_r) = u_j \). The motion policy is:

- \( \gamma(\mathbf{ye}_1 \lor \mathbf{ye}_2 \lor \mathbf{ye}_3) = u_1 \)
- \( \gamma(\mathbf{ye}_5 \lor \mathbf{ye}_8) = u_2 \)
- \( \gamma(\mathbf{ye}_4 \lor \mathbf{ye}_9) = u_3 \)
- \( \gamma(\mathbf{ye}_6) = u_4 \)
- \( \gamma(\mathbf{ye}_7) = u_5 \)

In which \( \lor \) 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 \( \partial E \) it is possible to identify the observations that represent the presence of gaps that do not disappear. Those observations are: \( ye_{1}^{L} = (0,1,0,1,0,1) \lor ye_{5}^{L} = (1,0,0,1,0) \). 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 \( \text{the change from cyclic to linear gap ordering when the gap sensor is touching the wall according to the model detailed in [4].} \) Local exploration uses linear lists. Those lists are \( \text{init-list and end-list. init-list} \) contains the current gaps read by the sensor and the preferential direction (\( rt \) or \( lt \) depending if the contact point is \( rp \) or \( lp \)). \( \text{init-list} \) tracks the changes over the gaps due to critical events, and the location of preferential direction (\( rt \) or \( lt \)) in the list. \( \text{end-list} \) contains gaps read by the sensor and the preferential direction after the motion primitive ends. The order of the gaps in \( \text{end-list} \) is different from the order in \( \text{init-list} \), because it is built at a different position. \( \text{init}_{f} \) is the first element of \( \text{init-list} \), \( \text{init}_{l} \) is the last element of \( \text{init-list} \), \( \text{init}_{rt} \) is the element containing the \( rt \) direction and \( \text{init}_{lt} \) is the element containing the \( lt \) direction. The equivalent elements in \( \text{end-list} \) have an analogous nomenclature. \( G_1 \) and \( G_2 \) are auxiliary lists containing specific subsets of \( \text{init-list} \) and \( \text{end-list} \) respectively. \( G_0 \) 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)

```
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_i^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
        end while
    end if
end if

else if lp = true then
    5. init-list ← Current gaps and \( lt \) direction starting from the sensor’s obstructed visibility region following a clockwise order;
    if ye_i = ye_i^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
        end while
    end if
end if

7. 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

8. end-list ← Current gaps and \( lt \) direction starting from the sensor’s obstructed visibility region following a clockwise order;

end if

9. \( G_1 \) ← \( \{ x \in \text{init-list} \mid \text{init}_{f} < x < \text{init}_{l} \} \);
10. \( G_2 \) ← \( \{ x \in \text{end-list} \mid \text{end}_{rt} < x < \text{end}_{l} \} \);
11. \( G_0 \) ← \( G_1 \land G_2 \);

for every gap \( g_i \in G_0 \) do
    12. Label node \( g_i \) in the GNT as a primitive node;
end for
```
at each time that observation $y^R_4$ or $y^L_5$ 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 $\partial E$ with point $r_p$ then the $G_1$ list includes all the gaps belonging to the open interval between the $r_p$ point starting position at the beginning of Algorithm 1 and the $r_t$ direction at the end of Algorithm 1. In this interval the order of the gaps is established in counterclockwise sense. Moreover, $G_2$ list includes all gaps belonging to the open interval between $r_t$ direction at the end of the algorithm and the $r_p$ 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 $r_p$ position and the current one in counterclockwise sense. Those gaps are generated by reflex vertices located within the unreachable region. Observation $y^R_4$ or $y^L_5$ 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 the offspring of them in $G_7$. Each time that observation $y^R_4$ or $y^L_5$ 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 $\partial E$ with point $l_p$ and the omnidirectional sensor is placed at $l_p$ the proof is analogous, it is just the symmetric case.

![Diagram](image)

Fig. 4: $R_{na}$ is shown in white and in region $R_a$ in dark grey. 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 $y^R_4$ or $y^L_5$ 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 on 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 $y^R_4$ or $y^L_5$ 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 $C$ 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 $C$ is simply connected then the landmark is found. If the collision free sub-set of the configuration space $C$ 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.

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 $r_t$. 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). In the multimedia material, we have included a video, in which two exploration simulations are presented. Some snapshots of the first simulation are presented in this section. The second simulation shows the case when the collision-free subset of the robot’s configuration space has several connected components. Fig. 5 shows the robot executing control $u_1$ that yields a straight line motion primitive, its initial position lies in the interior of $E$, it moves forward until a contact with $\partial E$ is detected, it is the only case where the robot does not follow the environment’s boundary. Fig. 6 shows that the landmark is totally visible from the omnidirectional sensor location, hence the landmark is encoded as a node child of the root in the GNT.

![GNT](image1)

Fig. 5: The robot is executing the straight line motion primitive. The corresponding GNT is shown.

![GNT](image2)

Fig. 6: Landmark encoding in the GNT.

Fig. 7 shows the 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).

![GNT](image3)

Fig. 7: GNT at the end of the execution of local exploration algorithm.

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.

REFERENCES