Intelligent Control and Automation, 2015, 6, 229-240
Published Online November 2015 in SciRes. http://www.scirp.org/journal/ica
http://dx.doi.org/10.4236/ica.2015.64022

Gap Navigation Trees for Discovering Unknown Environments

Reem Nasir, Ashraf Elnagar

Computational Intelligence Center, Department of Computer Science, University of Sharjah, Sharjah, UAE
Email: ashraf@sharjah.ac.ae

Received 27 July 2015; accepted 6 November 2015; published 9 November 2015

Copyright © 2015 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

We propose a motion planning gap-based algorithms for mobile robots in an unknown environment for exploration purposes. The results are locally optimal and sufficient to navigate and explore the environment. In contrast with the traditional roadmap-based algorithms, our proposed algorithm is designed to use minimal sensory data instead of costly ones. Therefore, we adopt a dynamic data structure called Gap Navigation Trees (GNT), which keeps track of the depth discontinuities (gaps) of the local environment. It is incrementally constructed as the robot which navigates the environment. Upon exploring the whole environment, the resulting final data structure exemplifies the roadmap required for further processing. To avoid infinite cycles, we propose to use landmarks. Similar to traditional roadmap techniques, the resulting algorithm can serve key applications such as exploration and target finding. The simulation results endorse this conclusion. However, our solution is cost effective, when compared to traditional roadmap systems, which makes it more attractive to use in some applications such as search and rescue in hazardous environments.

Keywords

Motion Planning, Gap-Navigation Trees, Roadmap, Robotics, Local Environments

1. Introduction

The problem of "exploring the environment surrounding a robot" can be divided into different sub-categories based on three parameters. The first parameter describes the environment itself and it is divided into simple and multiply connected. Simple environments are environments that have no holes (*i.e.*, obstacles) or discontinuity in its walls while multiply connected environments include holes. The second parameter is status of the obstacles whether it is static or dynamic. As more information is made available to the robot, the problem becomes

more challenging [1]-[3]. The third parameter is knowledge about the robot's environment which is classified into known, partially known or unknown to a robot [1] [3].

When a point robot is placed in a given (known) polygonal region, the focus of path planning research would be finding the path a robot would follow. The tasks required of a robot would determine which path should be considered, the fastest, shortest, safest, or less mechanical movements from the robot. When the task involves search and rescue, then the shortest path is considered, and in known environments, computing shortest paths is a straightforward task. The most common approach is to compute a visibility graph which is accomplished in O(n log n) time by a radial sweeping algorithm [4] where n is the number of visible points for the robot.

When the environment is unknown (or partially unknown), sensors are used to explore and map the environment to develop navigation strategies. In this case, the robot is equipped with sensors that measure distances, able to identify walls from objects and track them. The information reported from these sensors is used by the robot to build a map for the environment that will be used later to navigate through. This approach needs the robot to build an exact model of the environment with all of its walls, edges and objects with their exact coordination and measurements. This leads to raise the question of whether this is practical or not and how accurate these maps are. That leads to many proposed solutions [1], which of course will keep increasing as new issues with the current algorithms arise [5]. On the other hand, some research has been conducted on how probabilistic techniques would be applied to the current algorithms [6] and how that would affect their performance [7].

Our work focused on a relatively new approach where researchers studied the sensory data of a robot [8]. We studied which of these data were essential to the robot's ability to explore and navigate an environment [9]. We further detailed the algorithms required to explore an unknown environment using minimal information stored in the Gap Navigation Trees (GNT) data structure.

2. Problem Statement

The problem tackled in this paper is of two fold. The first is to explore a local environment cluttered with static obstacles. The second is to navigate such environment for carrying out a specific task.

Given an unknown, multiply connected environment with static obstacles, and a robot; the goal is (for the robot) to explore such environment and build a data structure (roadmap) while using the least amount of sensors possible. This data structure shall to be sufficient to achieve the robot's goals (finding an object, tracking an object or even just learning the environment).

We assume that the robot is modeled as a point moving in static, unknown and multiply connected environment, to achieve its goal. The robot is equipped with depth discontinuities sensor. The robot is expected to use the sensor's feedback to build a data structure which is used along with building local grid map as virtual landmarks, to learn and navigate among obstacles in the environment.

3. Proposed Algorithm

3.1. Gap-Navigation Trees (GNT)

GNT is a dynamic data structure constructed over direction of depth discontinuities [8]. The robot is modeled as a point moving in an unknown planar environment. The robot is assumed to have an abstract sensor (in the sense of [10]) that reports the order of depth discontinuities of the boundary, from the current position of the robot. These discontinuities are called gaps, and the abstract sensor may be implemented in a number of ways, using a directional camera or a low-cost laser scanner. Once GNT is constructed, it encodes paths from the current position of the robot to any place in the environment. As the robot moves, the GNT is updated to maintain shortest-path information from the current position of the robot. These paths are globally optimal in Euclidean distance. GNT showed that the robot can perform optimal navigation without trying to resolve ambiguities derived from different environments which result in having same GNT [8].

More research have been done on GNT in simple, closed, piecewise-smooth curved environment [11].Then it has been extended to multiply-connected environments (where the environment has holes, obstacles, jagged holes and edges) and new modification been added to handle the new events that have occurred [12] [13].

We assume that the robot is equipped with a depth sensor. This sensor has the ability to report discontinuities which are referred to as gaps.

Figure 1 shows how a robot detects gaps, where the nodes A and B represent starting points of the visibility edges of the robot's view. Beyond these edges, the robot has no knowledge of the environment; hence these

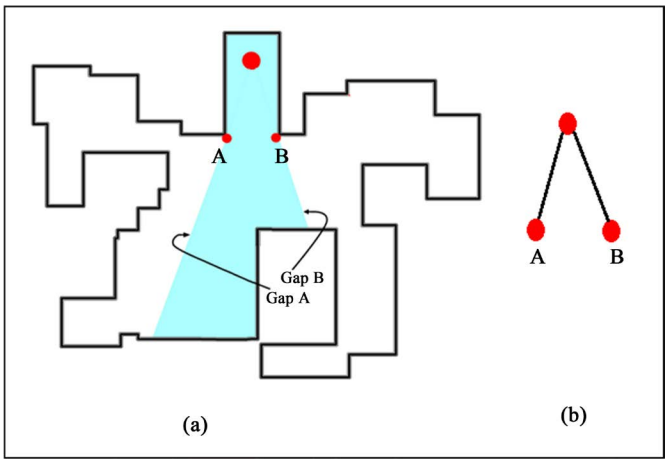

Figure 1. (a) Visibility region of a robot with depth sensor, (b) GNT.

nodes are added as children of the robot's current position. These nodes are called gaps. By adding children to each current position of the robot, and exploring them, the robot navigates and explores the whole environment until the whole environment is visible from these gaps.

There are three events that can occur while the robot investigates gaps:

- Addition of New Gaps: Each discontinuity in the robot's visibility field from its current position will be added as child gap node of the robot's node. This takes place while preserving the cyclic ordering from the gap sensor.
- Merger of Gaps: Two or more gaps could be merged into one gap, if they are the children of the same parent robot position and cover same area when investigated.
- Deletion of Gaps: If a gap becomes redundant, by being covered in the visibility range of the robot while examining another gap, and not a child for the same robot position parent node, then it will be deleted. When gaps belong to the same parent, but are in different directions (cannot be merged), the current gap will be kept, and the ones been seen (in the visibility range of the current gap) but not visited yet will be deleted.

Figure 2 is a detailed example of a robot that is building the GNT of a simple environment at different time steps. The highlighted area shows the visibility region from the current position of the robot. Edges instituted at the vertices are produced by the depth-discontinuity sensor (indicate gaps). Each gap shall be identified with a unique tag (a, b, a.1 ...) for the purposes of illustration. **Figure 2(a)**, shows three gaps, first one is (a), second one is (b), and the last one is (c). The robot will investigate the gaps in the order of detection (counterclockwise). Each black solid disc represents the position of the robot (Rc).

The GNT starts by constructing the Rc (the root is the initial robot's position) and 3 edges to the 3 newly identified gaps. **Figure 9(b)** demonstrates that while the robot is chasing the (a) gap, it encountered five new gaps, which are added as (a.1, a.2, a.3, a.4 and a.5). Notice that (a.1) and (a.2) share the same area but are represented as 2 gaps, same goes for (c), (a.3), and (a.4). There is also the merger between (a) and (b) in (a). That is because (b) is in the visibility range of (a) and they share the same parent. The robot has to explore these gaps in order to recognize that both cover the same area. That is because of its limited sensory data and its partial knowledge of the environment.

In (c) as the robot navigates toward (a.1), 2 events occur. First one is merging (a.1) and (a.2) as both children belong to the same parent. The second event is the deletion of (a.4) as it is completely in the robot's visibility range and therefore become redundant; (a.4) is deleted from GNT.

Applying the depth-first order, next gap to be investigated is (a.3). Once the robot reaches its new position, both of (c) and (a.5) are in the new Rc's visibility range. Therefore, both are deleted. As there are no further gaps to inspect in GNT, the robot stops exploration and declares that GNT is complete.

3.2. Landmarks Strategy

GNT works perfectly in simple planner environments and gives optimal results, but when it is used in more complex environments, it is not guaranteed to work. The depth discontinuity sensor is no longer sufficient; at

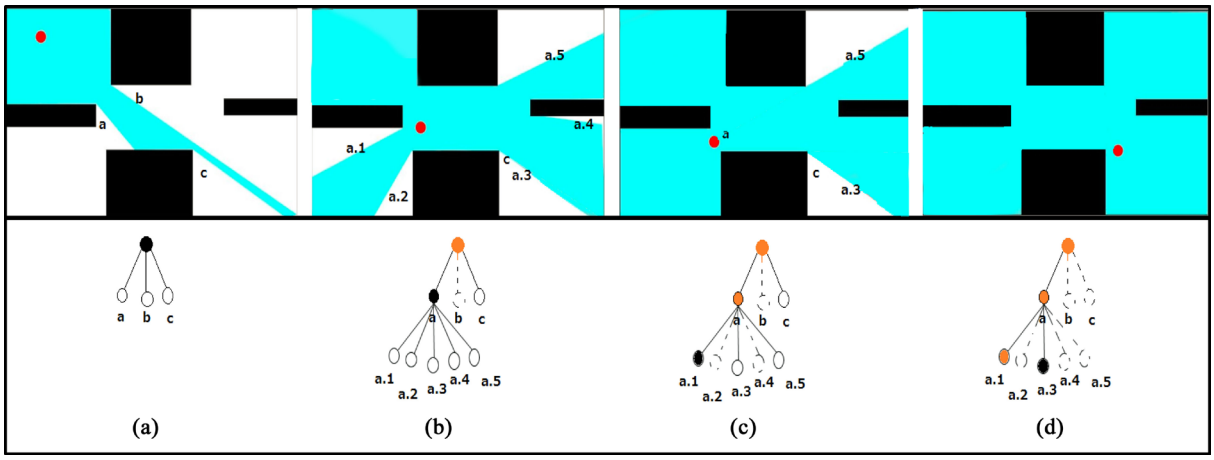

Figure 2. A robot navigating in simple environment and GNT building.

least another piece of data is required; because the robot needs to be able to recognize a gap that was previously visited. In simple environments each gap can be reached from one way only while in multiply connected environments, a gap could be reached from different sides. Therefore, the need to label such gaps becomes necessary. Previous works and suggestions had been made about combining GNT and landmarks, such as simple color landmark models [14], range measurements with respect to distinct landmarks, such as color transitions, corners, junctions, line intersections [15], scale-invariant image features as natural landmarks [16], and matching technology-RANSAC (Random Sample Consensus) [17].

3.3. Exploration Strategy

Before we present the proposed algorithm, we explain our landmark technique. The robot's initial configuration is assumed at location (0, 0) and as a result the environment is divided into 4 quadrants. As the robot moves, we keep track of new positions (x, y) as well as the number of steps.

The robot will always keep track of its original position, and all of the movements it made, with respect to its origin and how it will know (x, y) from any other gap, at any position or time. Our proposed landmark technique, the robot will locally build grid map incrementally for (x, y) values, this is only used to assign different values for each gap. Which will be used to identify each of them, therefore, the robot will know if it's the same gap or not and it will not get stuck around an obstacle, since each gap will be identified by the localized grid. The robot will know which gaps to merge and which to delete. Each gap is identified by the virtual gird map configuration. Next we describe the proposed algorithm and the classical A* search algorithm.

Algorithm 1: Modified Gap Navigation Tree
Input: sensory data (discontinuity in the environment from the depth sensor)
Output: complete *T*

```
1.  T = ϕ
2.  T = {G0}
3.  Let Gi = G0
4.  Update gridmap
5.  While (Gi is not cleared)
6.  ∀j where j<i
7.                          If (any Gi is redundant)
8.  then delete Gi
9.                  If (Gi is not visited)
10. then CHASE(Gi)
11.                 else
12.                         if (CGi is not cleared)
13. then Gi = next in order of unvisited children nodes
14.                         else
15.                                 mark Gi cleared
16.                                 make Gi a parent node.
17. end
18. report T
```

Algorithm 2: *CHASE*(G_i)
Input: a pointer to G_i in T
Output: perform clearing, deletion or addition

```
1.  Mark Gi visited.
2.  If ( there are discontinuities)
3.          For (each Gi _ discontinuities)
4.                  If (can't be seen by other previous gaps)
5.                          then add it and update gridmap
6.                  else
7.                          If ( current discontinuities and a previous gap have same parent)
8.                                  then merge it
9.                          else
10.                                 delete it
11. Gi = first discontinuities added as a child
12. else
13.         gap cleared
14. Gi = parent node
```

A* search algorithm will be used for navigation applications, where the robot will demonstrate how it can move from one point to another using the constructed GNT which was sufficient to represent the environment. A* is a complete search algorithm and will always find a solution if one exists. If the heuristic function h is admissible, meaning that it never overestimates the actual minimal cost of reaching the goal, then A* is itself admissible (or optimal) if we do not use a closed set. If a closed set is used, then h must also be monotonic (or consistent) for A* to be optimal. This means that for any pair of adjacent nodes x and y, where d(x, y) denotes the length of the edge between them, we must have: $\boldsymbol{h}(x) \le \mathrm{d}(x, y) + \boldsymbol{h}(y)$.

Algorithm2: *A* search algorithm*
Input: *SN*, *GN*, *Q*.
Output: best path from start node to goal node, if any

```
1. initialize the OL
2. initialize the CL
3. add SN to OL (you can leave its f at zero)
4. while (OL not empty)
5.          find the node with the least f on the open list, call it "q"
6.           pop q off OL
7.          set q's N parents to q
8.          for (N_i)
9.                  if (N_i is the GN)
10.                         then stop the search
11.                 N_i.g = q.g + distance between N_i and q
12.                 N_i.h = distance from GN to N_i
13.                 N_i.f = N_i.g + N_i.h
14.                 if (Q_i's position = N_i  is in OL and  Q_i.f<N_i.f )
15.                         then skip N_i
16.                 if (Q_i's position = N_i  is in CL and  Q_i.f<N_i.f)
17.                         then skip N_i
18.                 else add N_i to OL
19.         end
20.         push q in the CL
30. end
```

3.4. Complexity Analysis

The following is an analysis for the time and space complexities of the proposed algorithm. The complexity stems from two main phases, which are the buildup phase (constructing GNT) and the query phase (applying A* search algorithm).The time complexity of the first phase is heavily affected by the chasing phase, were the robot needs to check each gap, which takes $O(\mathrm{n})$. Then for each of these gaps, the robot checks the GNT before deciding to add new gaps, which is also another $O(\mathrm{n})$ task. The result is:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \approx O(n^2)$$

The storage space required to maintain the vertices of the resulted GNT is equal to the number of gaps n. Each

gap will also maintain pointers to its parent and its children too. Therefore, there will be no need to store the information of the edges as well. This leads to $O(n)$ storage complexity.

On the other hand, the time complexity of A* depends on the heuristic used. In the worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path), but it is polynomial when the search space is a tree, there is a single goal state, and the heuristic function h meets the following condition:

$$\left|h(x)-h^*(x)\right| = O\left(\log h^*(x)\right).$$

where h^* is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true distance from x to the goal.

4. Results and Analysis

In our simulations we classified the environments into six categories based on the size of obstacles and their distribution. The first four classifications are determined from the obstacles' sizes and the areas they occupy in the environment. The simulation started with sparse environments (*i.e.*, large free space) with uniform-size obstacles followed by sparse environments with variable-size obstacles. We repeated both simulations but in cluttered environments. At last, we simulated two popular problematic environments in the literature of the path planning field, namely, the Narrow-Passages problem and Indoor Environments. The following observations help appreciate the simulation results:

- Blue rectangles represent the obstacles in the environment.
- Red discs signify the initial robot position (Rs) or the current robot's position in the environment (Ri).
- Light orange discs indicate already visited gaps.
- Yellow discs represent the newly encountered gaps, which were added as children from the current robot position.
- Green discs indicate the next gap to be inspected.
- Green/black lines demonstrate the roadmap.
- **Maximum number of Gaps** (MNG): This refers to the theoretical maximum number of gaps, in the given multiply connected environment. The upper limit of the number of gaps is a function of the number of obstacles and their locations. For example, one rectangular obstacle would produce four gaps. If we consider having two obstacles in the environment, there will be a maximum number of eight gaps because the obstacles could be sharing a gap or more. Therefore we might end up with less than eight gaps, but it will not exceed eight. This info is reported by the simulator.
- **Unique Number of Gaps** (UNG): For validation purposes, this is obtained manually, by counting the number of gaps in a given environment. This number is less or equal to the maximum number of gaps. As the number of obstacles increase, in a given environment, this measure tends to be far less than the MNG.
- **Algorithm-computed Number of Gaps** (ANG): This refers to the total number of gaps encountered while constructing the GNT by the proposed algorithm. In an optimal scenario, ANG is the same as UNG. However, in a worse-case scenario it is close to MNG.
- **Gap Redundancy Rate Reduction** (GRR): This refers to the gap reduction in redundant gaps. The best-case scenario would yield testing UNG gaps only. Therefore, redundancy would be MNG-UNG. Therefore, we compute two reduction rates: optimal gap redundancy rate (OGRR) and algorithm-computed gap redundancy rate (AGRR), which are measured by:
  - OGRR= UNG/MNG.
  - AGRR= ANG/MNG.

Our objective is to minimize (AGRR-OGRR). That is the closer AGRR to OGRR is, the less is the redundancy in gaps.

We will start our simulation results with sparse environments which would have obstacles that occupy a small area of the environment. Here is an example of a uniform-size obstacles distribution. The environment has four small obstacles. Each 2 adjacent obstacles share a common gap.

As shown in **Figure 3**, the robot was able to identify each unique gap. Redundant gaps are well dealt with. The red node is the original robot's position and the root of the GNT (Rs). The light orange nodes represent the other gaps in the GNT. In this example, all of the gaps are the children of the root, which are sufficient to cover the whole environment. There were no further gaps to be added from any of its children.

Table 1 summarizes the observations on **Figure 3**. There were four obstacles in the environment. The final GNT consists of 6 nodes. In this example AGRR was 62.5%, which signifies a very good reduction of redundant gaps; it is close to the OGRR as well (68.7%).

Now we show how a robot explores a cluttered environment, in which the obstacles cover a large area of the environment. This example has 20 obstacles of uniform-size and uniform distribution. These obstacles cover a large area of the environment and are aligned as a grid. Rs is represented in red color.

Figure 4 depicts the resulting GNT. The light orange vertices correspond to the visited gaps. The four gaps in the first row of obstacles cannot cover the whole area and therefore another set of gaps has been added among the second row of obstacles. Although there are many obstacles, the final GNT was small because obstacles share significant part of the free-space, which is inversely proportional to the number of the required gaps in GNT.

Table 1. Observations on **Figure 2**.

Performance criterion	Measurement
Number of obstacles in the environment	4
MNG	16
UNG	5
ANG	6
OGRR	68.7%
AGRR	62.5%

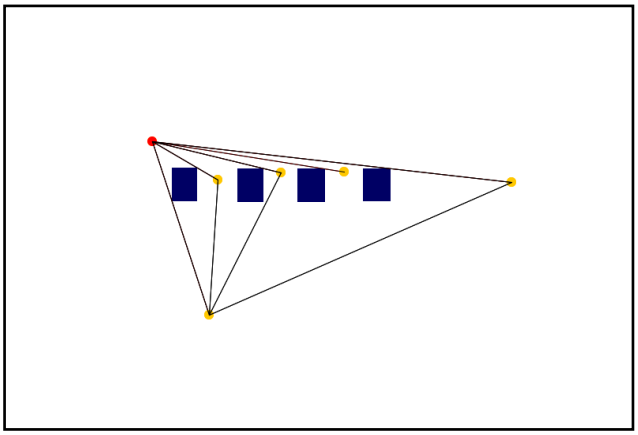

Figure 3. Sparse environment with four obstacles, complete GNT.

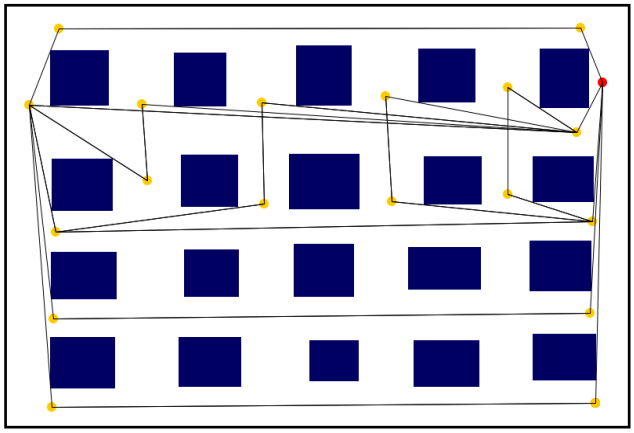

Figure 4. Cluttered environment with 20 obstacles, complete GNT.

Table 2 summarizes the observations on **Figure 4**. There are 20 obstacles in the environment with 80 unique vertices but only 19 gaps were added to the GNT. The final GNT was sufficient for this environment. Although the redundancy reduction rate (76.3%) is high, GNT provides a full coverage of the environment. The measure AGRR is not far from OGRR.

We also tested the performance in environments with variable-size obstacles. The obstacles are few; the free-space is sparse. The next example has only two obstacles.

The performance of the proposed algorithm is expected as when the number of obstacles decreases, the overlapping gaps are few and therefore GNT would yield a higher number of gaps. In practice, such environments are scarce.

Obstacle sizes and distribution in any environment are key parameters to construct the GNT. The algorithm computed a close number of gaps to the MNG. However, the actual number of gaps that are sufficient for this environment (3 gaps, see **Table 3**). This is the result of a global-based computation of the environment.

The following examples have environments with different obstacles' sizes that consume a large area of the environments. Some of these obstacles hide part of the environment (such part is not visible/shared with other obstacles) which increases ANG. Of course, it would decrease the AGRR as well.

Figure 6 represents an example of 5 obstacles that cover a large area of the environment and they differ in their sizes and distribution. One of these obstacles is a polygonal shape of 20 vertices.

Table 2. Observations on **Figure 4**.

Performance criterion	Measurement
Number of obstacles in the environment	20
MNG	80
UNG	11
ANG	19
OGRR	86.3%
AGRR	76.3%

Table 3. Observations on **Figure 5**.

Performance criterion	Measurement
Number of obstacles in the environment	2
MNG	8
UNG	3
ANG	7
OGRR	62.5%
AGRR	12.5%

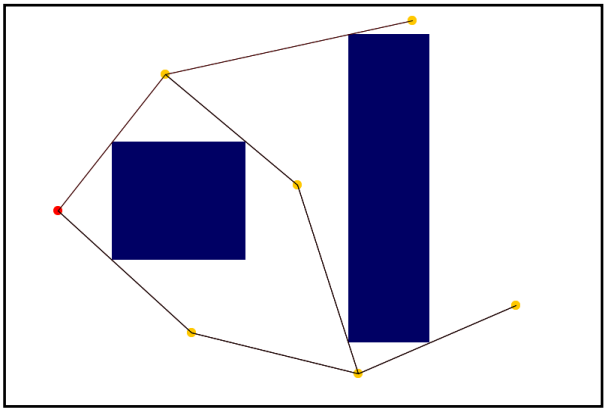

Figure 5. Sparse environment with four obstacles, complete GNT.

The polygonal-shape obstacle covers most of the middle area of the environment. The GNT structure grew surrounding this obstacle in order to cover the whole environment. The gap reduction rate of 55.5% is an encouraging result (**Table 4**).

The proposed algorithm can handle the popular narrow passages problem as well. It is a challenging environment that has been frequently reported in the literature. **Figure 7** shows an example of such environment with 5 obstacles that cover most of the workspace. There are two narrow passes in this environment.

The resulting GNT completely covers the free space. It is noticeable that the larger the obstacles get, the less gaps they share with other obstacles which leads to a less redundancy rate (**Table 5**).

Table 4. Observations on **Figure 6**.

Performance criterion	Measurement
Number of obstacles in the environment	5
MNG	36
UNG	11
ANG	16
OGRR	69.4%
AGRR	55.5%

Table 5. Observations on **Figure 7**.

Performance criterion	Measurement
Number of obstacles in the environment	5
MNG	20
UNG	13
ANG	14
OGRR	35%
AGRR	30%

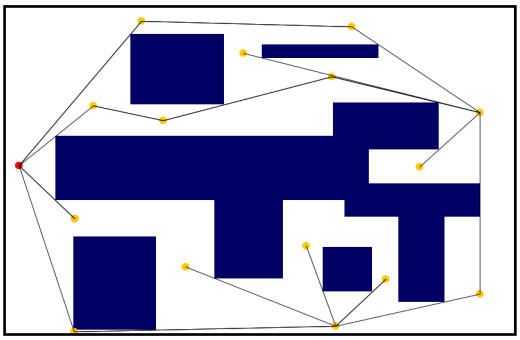

Figure 6. Polygonal-shape obstacle in a cluttered environment.

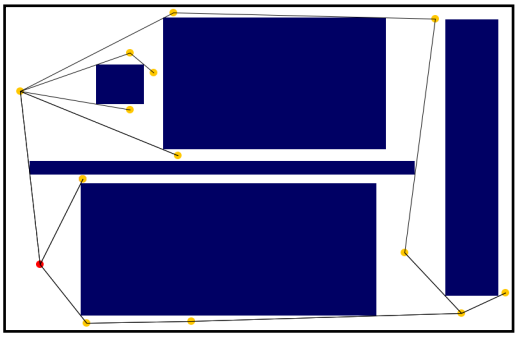

Figure 7. The narrow-passage problem.

Figure 8 combines a narrow-passage problem and a trap-type obstacle. The resulting GNT covers the whole environment's free-space eliminating the effect of such troublesome obstacles. There are eight obstacles. The robot successfully explored the environment and built the GNT. The performance of the algorithm is satisfactory as summarized in **Table 6**.

In summary, the vast set of simulations shed light on the performance of the proposed algorithm to construct a roadmap to cover the free-space. The algorithm is complete as it handles all types of environments including the challenging ones such as the narrow-passage or indoor environments. The resulting GNT is a function of Rs, obstacle size and placement. Although, the algorithm reduces redundant gaps, it does not eliminate all redundancy. This is very much attributed to the limited sensory data we use.

The following bar chart (**Figure 9**) describes the performance of the proposed algorithm with respect to gap redundancy reduction rate. The x-axis represents 13 simulations where **Figures 2-7** are represented as examples 2, 3, 6, 8, 9 and 11 respectively while the y-axis demonstrates the redundancy rate. This chart is a comparison between OGRR and AGRR.

Table 6. Observations on **Figure 8**.

Performance criterion	Measurement
Number of obstacles in the environment	8
MNG	40
UNG	16
ANG	26
OGRR	60%
AGRR	35%

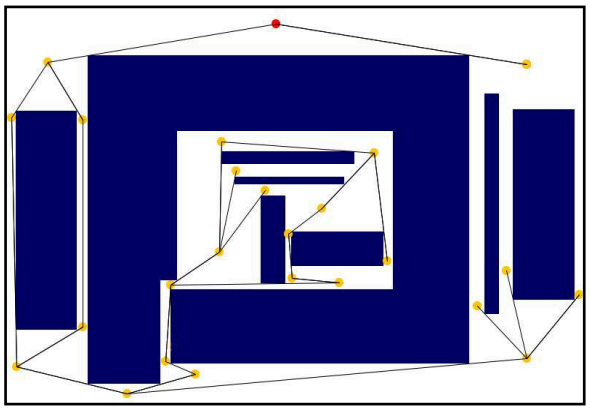

Figure 8. Trap-shape obstacles in an "indoor environment".

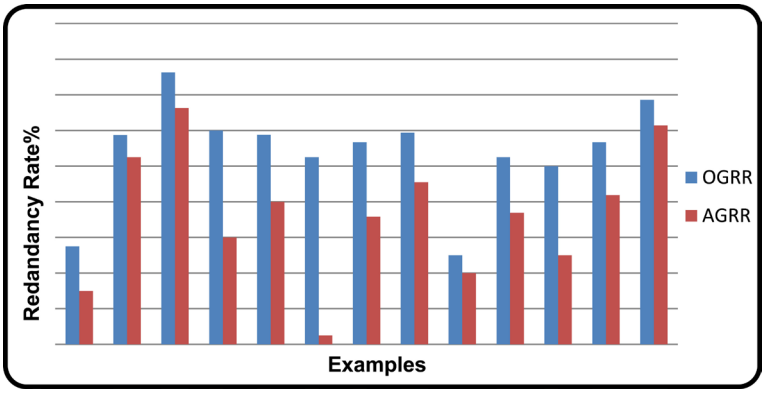

Figure 9. Bar-chart of the gap redundancy reduction rate.

Finding the minimum (unique) number of gaps is considered NP-hard problem. However, the performance of the algorithm (which uses local sensory data) is very much satisfactory when compared to the optimal one (NP-hard and based on global data). It should be noted that Example 6 (**Figure 5**) is a worst-case scenario where we had two obstacles; a scarce case.

5. Conclusions

In this paper, we have proposed using GNT while enabling the robot to count and record the steps it takes (as building its own grid) in order to enable GNT to work in multiply, connected environments. This solution would require using more RAM storage. However, the GNT is a versatile data structure that can serve a variety of applications.

The proposed solution is a cost-effective one which makes it practical to produce it in bigger quantities for specific application that require multiple robots such as search and rescue. A typical major assumption made to solve the robot motion planning using GNT is that obstacles are uniquely identified. Our solution does not require such assumption. Now, the robot is able to explore unknown multiply connected environments on its own with one very affordable sensor. Our solution eliminates the need for a landmark based sensor such as camera and it provides a cost effective prototype of a robot.

References

[1] Canny, J. and Reif, J. (1987) New Lower Bound Techniques for Robot Motion Planning Problems. Proceedings *IEEE Symposium on Foundations of Computer Science*, Los Angeles, 12-14 October 1987, 49-60. http://dx.doi.org/10.1109/sfcs.1987.42

[2] Murphy, L. and Newman, P (2008) Using Incomplete Online Metric Maps for Topological Exploration with the Gap Navigation Tree. *IEEE International Conference on Robotics and Automation*, Pasadena, 19-23 May 2008, 2792-2797. http://dx.doi.org/10.1109/robot.2008.4543633

[3] Craig, J. (2004) Introduction to Robotics: Mechanics and Control. 3rd Edition, Prentice Hall, Upper Saddle River.

[4] LaValle, S.M. and Hinrichsen, J. (2001) Visibility-Based Pursuit-Evasion: The Case of Curved Environments. *IEEE Transactions on Robotics and Automation*, **17**, 196-201. http://dx.doi.org/10.1109/70.928565

[5] Thrun, S., Burgard, W. and Fox, D. (2005) Probabilistic Robotics. MIT Press, Cambridge.

[6] Thrun, S., Burgard, W. and Fox, D. (1998) Probabilistic Mapping of an Environment by a Mobile Robot. *IEEE International Conference on Robotics and Automation*, Levene, 16-20 May 1998, 1546-1551.

[7] Elnagar, A. and Lulu, L. (2005) An Art Gallery-Based Approach to Autonomous Robot Motion Planning in Global Environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2-6 August 2005, 2079-2084. http://dx.doi.org/10.1109/iros.2005.1545170

[8] Tovar, B., Murrieta-Cid, R. and LaValle, S.M. (2007) Distance-Optimal Navigation in an Unknown Environment without Sensing Distances. *IEEE Transactions on Robotics*, **23**, 506-518. http://dx.doi.org/10.1109/TRO.2007.898962

[9] Nasir, R. and Elnagar, A. (2014) Exploration of Unknown Multiply Connected Environments Using Minimal Sensory Data. In: Zhang, X., *et al*., Eds., *ICIRA* 2014, Part I, LNAI 8917, 479-490. http://dx.doi.org/10.1007/978-3-319-13966-1_47

[10] Erdmann, M. (1995) Understanding Action and Sensing by Designing Action-Based Sensors. *The International Journal of Robotics Research*, **14**, 483-509. http://dx.doi.org/10.1177/027836499501400506

[11] Tovar, B., LaValle, S.M. and Murrieta, R. (2003) Optimal Navigation and Object Finding without Geometric Maps or Localization. *IEEE International Conference on Robotics and Automation*, **1**, 464-470.

[12] Tovar, B., LaValle, S.M. and Murrieta, R. (2003) Locally-Optimal Navigation in Multiply-Connected Environments without Geometric Maps. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **4**, 3491-3497. http://dx.doi.org/10.1109/iros.2003.1249696

[13] Tovar, B., Guilamo, L. and LaValle, S.M. (2005) Gap Navigation Trees: Minimal Representation for Visibility-Based Tasks. *Algorithmic Foundations of Robotics VI*, Springer Berlin Heidelberg, 425-440.

[14] Yoon, K-J. and Kweon, I. (2002) Landmark Design and Real-Time Landmark Tracking for Mobile Robot Localization. *Electrical Engineering*, **4573**, 219-226.

[15] Bais, A. and Sablatnig, R. (2006) Landmark Based Global Self-Localization of Mobile Soccer Robots. *Computer Vision-ACCV*, **3852**, 842-851. http://dx.doi.org/10.1007/11612704_84

[16] Se, S., Lowe, D. and Little, J. (2002) Mobile Robot Localization and Mapping with Uncertainty Using Scale-Invariant

Visual Landmarks. *The International Journal of Robotics Research*, **21**, 735-758. http://dx.doi.org/10.1177/027836402761412467

[17] Zhao, L., Li, R., Zang, T., Sun, L. and Fan, X. (2008) A Method of Landmark Visual Tracking for Mobile Robot. *Lecture Notes in Computer Science*, **5314**, 901-910. http://dx.doi.org/10.1007/978-3-540-88513-9_97