Ground Robot Navigation strategies in Known and Unknown Environments

Khokhar, Arsh
khokharm@myumanitoba.ca

29th July 2021

Abstract

Robot navigation is one of the most important problems researched in mobile robotics. For any mobile robot, path planning and navigation is essential component for it to accomplish its task successfully. The problem is studied under many settings, with a combination of things like static or dynamic environments, known or unknown environments, and robots varying in degrees of freedom. In this survey paper, we consider a setting for a ground robot with three degrees of freedom that is placed in a known or unknown static environment with a goal and obstacles. We discuss two different approaches for two-dimensional navigation and path planning for ground robots, and analyze their respective performance, accuracy, and completeness. In the end, we also consider extensiblity for dynamic environments of the discussed approaches.

1 Introduction

Robot Navigation is one of the most important problems researched in mobile robotics. Navigation is an important task for any mobile robot, since being at the right place at the right time is a precursor to performing almost any useful task [1]. Navigation techniques vary highly based on specific environment settings and robot specifications. For example, a competent navigation technique for known environments can fail to provide a solution in a timely manner in unknown environment, or even fail to provide any solution altogether. While environments that are known to the robot beforehand and contain static obstacles are relatively easier to analyze, it is highly important to study the problem under dynamic environment and moving obstacles for realizing more realistic and fail-proof approaches.

Navigation is a highly complex task for the mobile robots to execute. At a high level, navigation can be understood as answering three crucial questions for any mobile robot: Where am I? Where do I have to go? How do I get there? [2] While humans are able to answer these questions in their daily lives without much effort, mobile robots need highly sophisticated techniques in order to find answers to these questions. Formally, to answer these questions, robot navigation can be broken down into the three components as follows:

Self-localization: In order to navigate successfully, a mobile robot has to determine its position in the environment, either in a global context or relative to its goals, also known as self-localization. Self-localization may involve different levels of precision, from exact coordinates to abstract details based on the technique used [2]. A hybrid method that can increase the localization precision as the robot gets closer to its goal can also be used to achieve both accuracy and performance.

Mapping: Mobile robots also need to map their movements with their physical environments. Keeping track of the paths and movements that the robot has taken until a certain point of time can help the robot in self-localization. However, mapping is especially useful in unknown or partially known environments, where the robot can increase its knowledge-base about the environment by keeping track of the obstacles and other key features of the environment [2].

Motion Planning and Path Planning: Motion Planning is the problem of finding a robot motion from a start state to a goal state that avoids obstacles in the environment and satisfies other constraints, such as joint limits or torque limits. [3]. **Path Planning** is a subset of general motion planning that focuses on finding an obstacle-free path from the robot to its goal in the environment. Planning answers the question: What is the best way to reach the goal? [2].

Ground robots operating in two dimensions have 3 degrees of freedom: 2 degrees on the ground plane and an angle. Under these conditions, the navigation problem can be formalized as shown in figure 1. The complexity of the navigation problem is highly dependent on the type of the environment. The environment can either be known to the robot beforehand, or it can be semi-known or completely unknown. Likewise, the environment can either be dynamically changing with the obstacles moving, or it can be static where the obstacles are fixed.

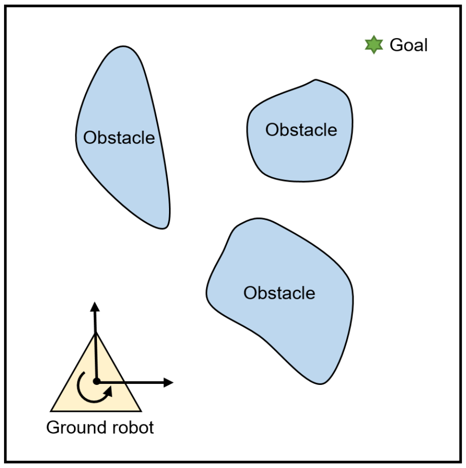

Figure 1: Navigation problem for a robot with 3 degrees of freedom

2 Key Definitions

Configuration Space: In order to take an account of robots with different geometries and mechanical constraints, path planning is solved in a new space called the configuration space, in which the robot is presented in a point form along with the physical environment points mapped to robot's all individual configurations with sizes of the obstacles normalized with respect to the robot. It is commonly known as the C-space in path planning problems [2].

Forbidden or Obstacle Space: Forbidden or Obstacle space is the space occupied by obstacles in the environment. While navigating towards a goal, a mobile robot must avoid going into the forbidden space in order to avoid collision with the obstacles [2].

Free Space: Free space is the subset of the configuration space that is not occupied by any obstacles. It is commonly known as C-free in the path planning problems. The robot has to travel through the free space in the environment in order to reach from one point to another [2].

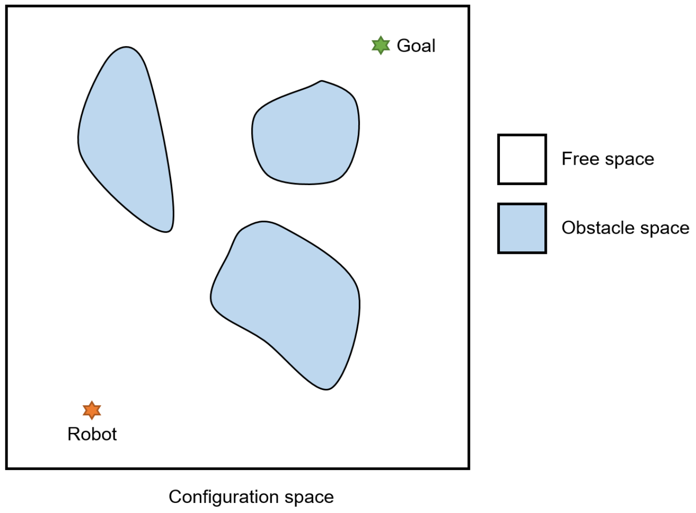

Figure 2: Space decomposition of a navigation environment

3 Path Planning Approaches

3.1 Cell decomposition Approach

Cell decomposition methods are some of the classical methods used in robot navigation. In these methods, the configuration space of the robot is recursively subdivided into multiple smaller regions or cells until a set of regions is found that is free and can be taken by the robot to navigate towards its goal. Combining these cells, a connectivity graph is generated, where each node in the connectivity graph represents a free cell. Based on the nature of the algorithm used for the subdivision, cell decomposition can either be approximate or exact. In an exact cell decomposition algorithm, the union of all final free cells found is equal to C-free, i.e., no free space remains unrecognized in the end. However, in an approximate cell decomposition algorithm, typically, a cell is classified as one of the three types: free (cells which have no obstacles), blocked (cells which are completely filled with obstacles) or mixed (cells which have some part of them occupied by an obstacle). Because of the mixed nature of certain cells in the division, the union of all cells that are classified as free in the end is not necessary equal to C- free [4]. Based on the granularity level of the subdivision technique been used, some of the free space might remain undetected in the end. Figures 3 and 4 demonstrate the exact and the approximate cell decomposition results.

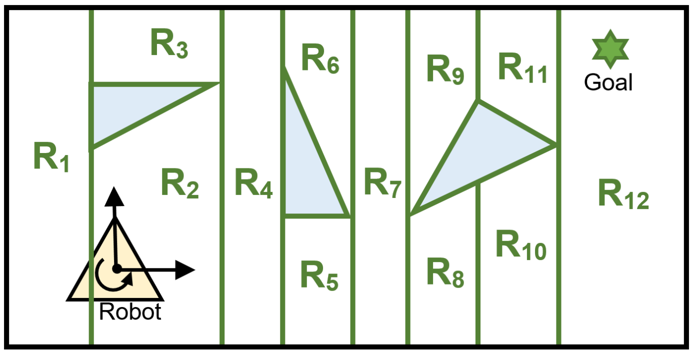

Figure 3: Exact cell decomposition

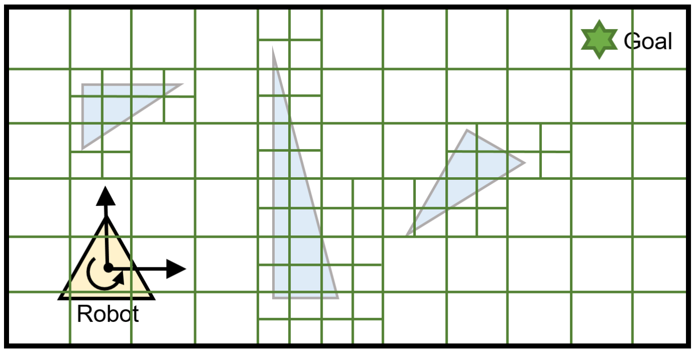

Figure 4: Approximate cell decomposition

In the approximate cell decomposition methods, at each level, any cell that is mixed (i.e. not classified as blocked or free) is further broken down recursively, which results in a tree structure. The leaf nodes of this tree are the cells that are either fully occupied by an obstacle or free. Also, while constructing the tree, the adjacency links between the cells are also maintained, in order to find a path that connects a series of free cells from the robot towards the goal [1]. After constructing the decomposition tree, its traversal is necessary in order to plan the path towards the goal.

3.2 Flexible Binary Space Partitioning

One of the major issues with the cell decomposition methods is that they usually focus more on optimizing and making the decomposition methods more efficient, while for-going the overhead of making more intelligent decisions when making the partitions. Some cell decomposition methods like Quadtree or Octree decomposition subdivide the cells into four or eight smaller cells that are of uniform size (i.e. have the width and height ratio same as the physical space). In many cases, this uniform size constraint leads to a non-ideal subdivisions, because of no attempt in combining the free or blocked spaces into fewer cells together. The uniform subdivision constraint sometimes leads to more mixed cells, which in turn requires further subdivisions. As discussed earlier, a tree structure needs to be constructed in order to plan a path towards the goal. Because of the more subdivisions and more number of cells, the decomposition tree consists of more cells in the end.

In addition, the cells that are connected by the same obstacle can be in different deep branches. As such, two leaf nodes for a free space that are connected, will require inefficient and deep branch traversals in the planning step. In contrast, when this uniform size constraint is relaxed and cells are allowed to have different ratios of widths and heights by intelligently subdividing, it can be seen that the resulting tree can be much smaller, providing a much more efficient path planning towards the goal.

Flexible Binary Space Partitioning addresses this issue by associating an additional heuristic in the subdivision step that allows to identify the connecting open spaces quickly and allowing them to be grouped together in the tree rather than appearing as isolated regions. It is based on the binary-space partitioning technique, where at each level, a mixed cell is further divided into two cells.

Figures 6 and 5 shows the recursive steps taken by the Quadtree approach vs. Flexible BSP. At each level, notice that the Flexible BSP marks each of the transition points from blocked spaces to free spaces as potential places where the splitting could take place.

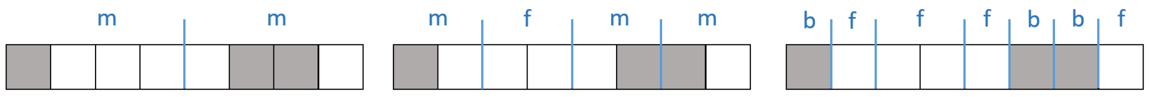

Figure 5: Quadtree decomposition

After identifying these potential splitting points, the algorithm intelligently chooses one of these potential points for the split based on the entropy – the percentage of the mixture in the resulting partitions. The entropy of a spatial area A is defined as

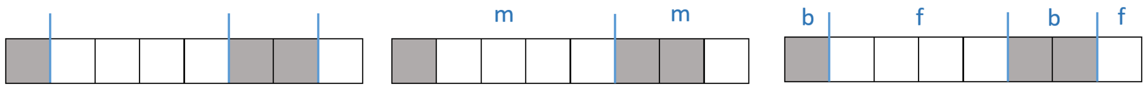

Figure 6: Flexible Binary Space Partitioning decomposition

follows:

$$Entropy(A) = -p_f \log_2 p_f - p_b \log_2 p_b$$

where $p_f = \frac{|f|}{|A|}$ is the percentage of free area and $p_b = \frac{|b|}{|A|}$ is the percentage of blocked area. The heuristic to identify the best point for splitting amongst all the possible split points is the *gain in entropy*, defined as:

$$gain(p_i, A) = E(A) - \Sigma \frac{|P|}{|A|} E(|P|)$$

Using the above equations, Flexible BSP intelligently chooses the split point that gives the highest gain in the entropy. Extending the strategy in a similar way in two-dimensions, the Flexible BSP approach can give much better decomposition results as compared to the classical cell decomposition approaches such as Quadtree decomposition.

3.3 Roadmap Approach

In the roadmap methods of path planning, the free space available to the robot is mapped to a one-dimensional network (e.g. lines, trees, and graphs) where the position of the robot and the position of the goal are two vertices on the network. Each of the path on this network that leads the robot to its goal is known as a road, and hence the collection of these roads is called a roadmap. However, unlike just graph traversal, the roadmap approach requires the robot to follow a unique trajectory while moving from one vertex to another and navigating through the space. Algorithms that fall under the roadmap type of approach attempt for the robot to be able to navigate through as much free space as possible while staying on the network and also be able to reach its goal [5].

3.3.1 Generalized Vornoi diagrams

A Vornoi diagram for a set of sites in a plane is a collection of regions that divide up the plane. Each region corresponds to one of the sites and all the points in one region are closer to the site that induced the region than to any other region [6]. Examples of some of the potential algorithms to generate the Vornoi diagrams are plane and sweep, incremental and divide and conquer [7]. In the given setting, the obstacles correspond to the sites in the Vornoi diagram, while the points on the diagram represent the potential paths that the robot can take in order to reach its goal. But, realistically, the obstacles are not in a point-form. So, each point on the outer edge needs to be considered as an obstacle, which gives many infeasible paths. To omit taking the infeasible path, one of the possible solutions is to use sensors like laser rangefinder or ultrasonic, for localization and detecting the nearest Vornoi edges that are not covered by a physical obstacle in the environment [5]. Moving on the edge that is the farthest from all the visible obstacles, except the first edge and the last edge of the path is also an option [7].

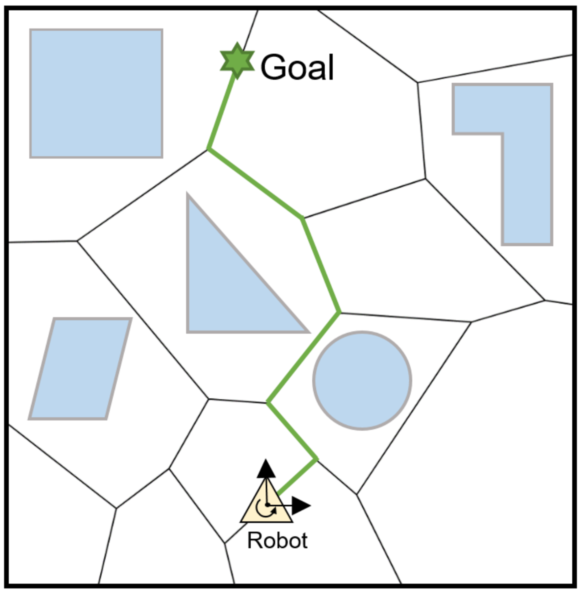

Figure 7: Generalized Vornoi diagram for robot navigation

However, a better approach is to make use of the Generalized Vornoi diagrams. In this technique, instead of considering each point on the boundary as a separate obstacle, the boundaries for any object are approximated as lines, and the bisectors can be computed as lines and parabolas. Therefore, the Vornoi roadmap generated using the Generalized Vornoi diagrams consists of either lines or parabola, which gives much smoother trajectories [7]. In addition, Generalized Vornoi diagram omits a high number of infeasible paths by generating the roadmap that the robot would otherwise have to dynamically avoid using its local sensors while navigating towards the goal.

3.4 Comparision

3.4.1 Computational complexity and Completeness

Generalized Vornoi diagrams Flexible Binary Space Partitioning differ vastly in their nature, and both have their advantages and drawbacks in certain situations. Cell approximation techniques, because of their recursive subdivision nature can construct extremely deep trees in complex environments, leading to a combinatorial explosion. While Flexible BSP addresses this issue by making the subdivision trees smaller, it is still prone to this issue because of its inherent nature. One of the possible safeguards towards this could be using a technique such as iterative deepening. However, using the safeguards would add an additional computational overhead, and increase the possibility of not finding a solution. Due to this, Flexible BSP is not a complete approach, i.e., does not guarantee to find a solution in some particular settings. In contrary, roadmap techniques like the Vornoi diagram are guaranteed to find a solution if there exists one, so they are complete [5]. But, since the Generalized Vornoi Diagram method approximates the obstacle outlines as lines, for complex shaped obstacles, it might not provide an accurate path either.

3.4.2 Extensibility for dynamic environments

Both the Flexible BSP and Generalized Vornoi Diagram methods can be used in dynamic environments. The key advantage that Flexible BSP has over the other methods is the smaller subdivision tree and grouping of the likewise space together. This facilitates rapid regeneration and path calculation in a dynamic environment. Likewise, due to the use of sensors and continuous localization, Generalized Vornoi Diagram can choose and regenerate its navigation strategy dynamically. In addition, since Generalized Vornoi roadmap is designed to keep the automatically farther from the obstacles, moving obstacles can still be avoided without much variance from the original path because of their distance from the robot.

References

[1] J. Baltes and J. Anderson. Flexible binary space partitioning for robotic rescue. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 4, pages 3144–3149 vol.3, 2003.

[2] Anis Koubaa, Hachemi Bennaceur, Imen Chaari, Sahar Trigui, Adel Ammar, Mohamed-Foued Sriti, Maram Alajlan, Omar Cheikhrouhou, and Yasir Javed. *Robot Path Planning and Cooperation*. 01 2018.

[3] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017.

[4] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. 2015.

[5] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2nd edition, 2011.

[6] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[7] Seda Milos and Václav Pich. Robot motion planning using generalised voronoi diagrams. pages 215–220, 08 2008.