Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Efficient grid-based spatial representations for robot navigation in dynamic environments

Boris Lau [*], Christoph Sprunk, Wolfram Burgard

Autonomous and Intelligent Systems, University of Freiburg, D-79110 Freiburg, Germany

ARTICLE INFO

Article history:
Available online 30 August 2012

Keywords:
Incremental algorithms
Voronoi diagrams
Distance maps
Configuration space
Collision checking
Robot navigation

ABSTRACT

In robotics, grid maps are often used for solving tasks like collision checking, path planning, and localization. Many approaches to these problems use Euclidean distance maps (DMs), generalized Voronoi diagrams (GVDs), or configuration space (c-space) maps. A key challenge for their application in dynamic environments is the efficient update after potential changes due to moving obstacles or when mapping a previously unknown area. To this end, this paper presents novel algorithms that perform incremental updates that only visit cells affected by changes. Furthermore, we propose incremental update algorithms for DMs and GVDs in the configuration space of non-circular robots. These approaches can be used to implement highly efficient collision checking and holonomic path planning for these platforms. Our c-space representations benefit from parallelization on multi-core CPUs and can also be integrated with other state-of-the-art path planners such as rapidly-exploring random trees.

In various experiments using real-world data we show that our update strategies for DMs and GVDs require substantially less cell visits and computation time compared to previous approaches. Furthermore, we demonstrate that our GVD algorithm deals better with non-convex structures, such as indoor areas. All our algorithms consider actual Euclidean distances rather than grid steps and are easy to implement. An open source implementation is available online.

1. Introduction

Many approaches in robot navigation rely on occupancy grid maps to encode the obstacles of the area surrounding a robot. These maps can be learned from sensor data, they are well suited to solve problems like path planning, collision avoidance, or localization, and they can easily be updated to reflect changes in the environment.

In the past, several grid-based spatial representations have been developed that can be derived from grid maps, e.g., distance maps, Voronoi diagrams, and configuration space maps. These representations are important building blocks for many different robotic applications, since they can be used to speed-up algorithms that solve the aforementioned problems. This paper proposes incremental update algorithms to facilitate the online use of these representations in dynamic environments. We also apply our methods to update distance maps and Voronoi diagrams in the configuration space of non-circular robots, e.g., to speed-up path planning or collision avoidance for these types of platforms. This paper extends our previous work on these topics [2,3] and includes additional experiments for 3D distance maps and incremental updates of distance maps and Voronoi diagrams in the context of simultaneous localization and mapping (SLAM).

The generalized Voronoi diagram (GVD) is defined as the set of points in free space to which the two closest obstacles have the same distance [4]. It is a discrete form of the Voronoi graph, which has been widely used in various fields [5]. In the context of robotics, Voronoi graphs are a popular cell decomposition method for solving navigation tasks. Their application as roadmaps is an appealing technique for path planning, since they are "sparse" in the sense that different paths on the graph correspond to topologically different routes with respect to obstacles. This significantly reduces the search problem and can be used for example to generate the n-best paths for offering route alternatives to a user [6]. Also, moving along the edges of a Voronoi graph ensures the greatest possible clearance when passing between obstacles. When Voronoi graphs are discretized and stored as a map, they can lose their sparseness property due to erroneous interconnections between neighboring Voronoi lines. Our method to compute GVDs overcomes this problem with additional conditions that ensure the sparseness of the generated GVDs.

The cells in a distance map (DM) encode the distance to the closest cell that is occupied according to the corresponding occupancy map. Since a cell lookup only requires constant time, DMs provide efficient means for collision checks, to compute traversal costs for path planning, and for robot localization with likelihood fields [7].

[*] Corresponding author. Tel.: +49 761 203 8012.
E-mail address: lau@informatik.uni-freiburg.de (B. Lau).

0921-8890/$ – see front matter © 2013 Published by Elsevier B.V.
doi:10.1016/j.robot.2012.08.010

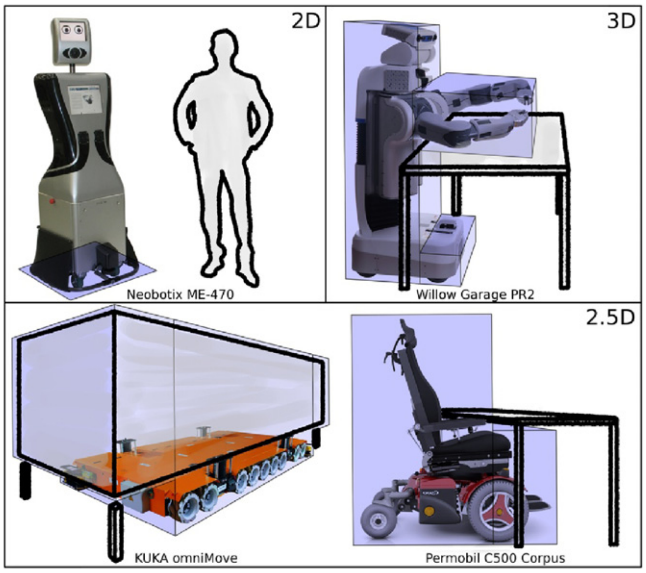

Fig. 1. For some applications, representing obstacles and robots by their 2D footprints can be sufficient (top-left). For overhanging parts of robots, their load, or obstacles, 2.5D representations are needed (bottom), whereas interaction tasks can also require actual 3D obstacle and robot models (top-right). Robot shape approximations as used in our experiments are depicted in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Since the computation of this transform is carried out without considering the shape of the robot, direct application of plain DMs is restricted to circular approximations of the robot's footprint.

For non-circular robots in passages narrower than their circumcircle, however, circularity is too crude an assumption, and collisions have to be checked for in the three-dimensional configuration space (c-space) of robot poses. Also, even for robots moving on a plane as considered in this paper, 3D obstacles and collisions can be important: applications such as robotic transporters, wheelchairs, or mobile manipulators can require the robot to partially move underneath or above obstacles as shown in Fig. 1. In these cases, collision checks easily become a dominant part of the computational effort in path planning. However, by convolving a map with the discretized shape of the robot, one can precompute a collision map that marks all colliding poses. With such a map, a collision check requires just a single lookup, even for 3D obstacle representations.

In changing environments, precomputed GVDs, DMs, and c-space maps have to be updated regularly to always reflect the current state of the corresponding occupancy map. These changes can be caused by moving people or vehicles, newly explored areas during mapping, or when correcting a map after closing a loop in SLAM.

In this paper, we present efficient methods to compute and update these representations. Since our algorithms perform all updates in an incremental way, i.e., recomputing only parts affected by changes, they can be applied online even with large maps or with more than two dimensions. In comparison to previous approaches, our methods require less computational effort, are easy to implement, and work in both indoor and outdoor environments.

Additionally, we combine DMs and GVDs with c-space collision maps, and propose distance transformed c-space maps and c-space Voronoi diagrams. These can be used for efficient collision checking and path planning of non-circular robots. With our algorithms described in this paper, these representations can be updated in an incremental way as well.

After discussing related work in Section 2, we describe the brushfire algorithm in Section 3. It can be used to compute static DMs, and it is an important foundation for our dynamic DM and GVD algorithms proposed in Sections 4 and 5. Section 6 describes our dynamic c-space collision maps, followed by the c-space DM and c-space GVD in Section 7. Our experiments are presented in Section 8 before we conclude our paper in Section 9.

2. Related work

In the past, many different approaches have been proposed to compute DMs, GVDs, and c-space collision maps. With the goal of applying them online in dynamic environments, a lot of effort has been spent on developing more efficient algorithms. However, unlike ours, most of these approaches do not exploit the potential of incremental updates. The remainder of this section presents related work for the different spatial representations and discusses the contribution of our methods.

2.1. Distance maps

Many different approaches have been proposed to compute static two-dimensional DMs, e.g., linear image traversal [8], dimensional decomposition [9], and distance propagation with the brushfire algorithm [10]. We review the brushfire algorithm in Section 3. For a comparative review of other approaches please refer to the survey by Fabbri et al. [9].

Whenever a cell in a grid map is newly occupied or vacated, the corresponding DM has to be updated to reflect this change. A trivial method is to recompute distances for patches within $\hat{d}$ around all changed cells, where $\hat{d}$ is an upper bound on the minimum obstacle distance in the environment. However, this method usually updates substantially more cells than necessary, e.g., if $\hat{d}$ is high due to large open spaces or if the changed cells cover a wide area. Furthermore, efficiently determining the minimal update area is not trivial, if the changes affect the occupancy of several cells.

Kalra et al. proposed a dynamic brushfire algorithm that incrementally updates DMs and GVDs by propagating wavefronts starting at newly occupied or vacated cells [11]. While their method is based on the incremental path planning algorithm D^* by Stentz [12], the algorithm proposed here is directly derived from the brushfire algorithm and requires substantially less computational time for the same task due to a considerably reduced number of cell visits.

The wavefronts of Kalra et al. accumulate 8-connected grid steps to approximate obstacle distances [11]. This overestimates the true Euclidean distances by up to 8.0% [13], which for a robot implies either a collision risk or overly conservative movements. Scherer et al. adopted and corrected Kalra's algorithm for their DM update method [14]. They propagate obstacle locations rather than grid step counts to determine Euclidean distances, which reduces the absolute overestimation error below an upper bound of 0.09 pixel units [13]. According to Cuisenaire and Macq, the shortest distance at which this propagation error can occur is 13 pixels [15], which yields a maximum relative error of 0.69%. Additionally, by propagating obstacle references, our representations can provide the location of the closest obstacle rather than just the distance to it, which can be appealing for collision avoidance methods. In a recent publication, Scherer et al. build on our original method for DM updates and combine it with their approach to map scrolling [16].

This paper extends our DMs presented in [2] to 3D by adding the possibility to limit the propagated distances to maintain online feasibility in large open spaces and outdoors as proposed by Scherer et al. [14].

Additionally, we describe how to further reduce the number of visited neighbor cells, which increases the efficiency for 3D DMs, and we present additional experiments.