Robotics and Autonomous Systems 57 (2009) 123–128

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Incremental reconstruction of generalized Voronoi diagrams on grids

Nidhi Kalra [a,*], Dave Ferguson [b], Anthony Stentz [c]

[a] RAND Corporation, Pittsburgh, USA
[b] Intel Research, Pittsburgh, USA
[c] The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

ARTICLE INFO

Article history:
Received 1 May 2006
Received in revised form
27 September 2006
Accepted 31 January 2007
Available online 3 August 2007

Keywords:
Voronoi diagrams
Incremental algorithms
Robot navigation

ABSTRACT

We present an incremental algorithm for constructing and reconstructing Generalized Voronoi Diagrams (GVDs) on grids. Our algorithm, Dynamic Brushfire, uses techniques from the path planning community to efficiently update GVDs when the underlying environment changes or when new information concerning the environment is received. Dynamic Brushfire is an order of magnitude more efficient than current approaches. In this paper we present the algorithm, compare it to current approaches on several experimental domains involving both simulated and real data, and demonstrate its usefulness for multirobot path planning.

1. Introduction

Efficient path planning is a fundamental requirement for autonomous mobile robots. From a single robot navigating in a partially-known environment to a team of robots coordinating their movements to achieve a common goal, autonomous systems must generate effective trajectories quickly. The efficiency of path planning algorithms can be greatly affected by the type of representation used to encode the environment. Common representations include uniform and non-uniform grids, probabilistic roadmaps, generalized Voronoi diagrams (GVDs), and exact cell decompositions. GVDs in particular are very useful for extracting environment topologies. A GVD is a roadmap that provides all possible path homotopies in an environment containing obstacle regions. The GVD also provides maximum clearance from these regions. Such a representation has practical applications to many robotic domains such as multirobot planning, stealthy navigation, surveillance, and area coverage. Fig. 1 presents the GVD of an outdoor environment.

GVDs can be used as complete representations of their environments [13,3] and to augment other representations such as probabilistic roadmaps [6,5]. Given a GVD, planning from a start position to a goal position consists of three steps. First, plan from the start to its closest point on the GVD (the access point). Second, plan along the GVD until the point closest to the goal is reached (the departure point). Then, plan from the departure point to the goal. Since the GVD is a graph, any graph search algorithm can be used to plan between the access and departure points, often at a fraction of the computational expense required to plan over the complete environment.

In many domains, robots must navigate in environments for which prior information is unavailable, incomplete, or erroneous. To harness the benefits of GVDs for planning in these environments, a robot must update its map when it receives sensor information during execution, reconstruct the GVD, and generate a new plan. GVD reconstruction and replanning must occur frequently because new information is received almost continuously from sensors. However, because new information is usually localized around the robot, much of the previous GVD remains correct and only portions require repair. Unfortunately, as we discuss in Section 2, the existing algorithms for constructing GVDs have no local reconstruction mechanism and cannot take advantage of this feature; instead, they discard the existing GVD and create a new one from scratch. This frequent full reconstruction is both computationally expensive and unnecessary.

In this paper, we present the Dynamic Brushfire algorithm for incremental reconstruction of GVDs on grids. We first discuss related techniques for GVD construction. Next, we present our algorithm both intuitively and through pseudocode. We then compare this algorithm to existing algorithms on several common robot navigation scenarios on both real and simulated data. We also demonstrate the usefulness of GVDs and our algorithm in particular for coordinated multirobot path planning.

* Corresponding address: RAND Corporation, 4570 Fifth Avenue, Suite 600, Pittsburgh, PA 15213, USA. Tel.: +1 412 683 2300x4637; fax: +1 412 683 2800.
E-mail address: nkalra@rand.org (N. Kalra).

0921-8890/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2007.01.009

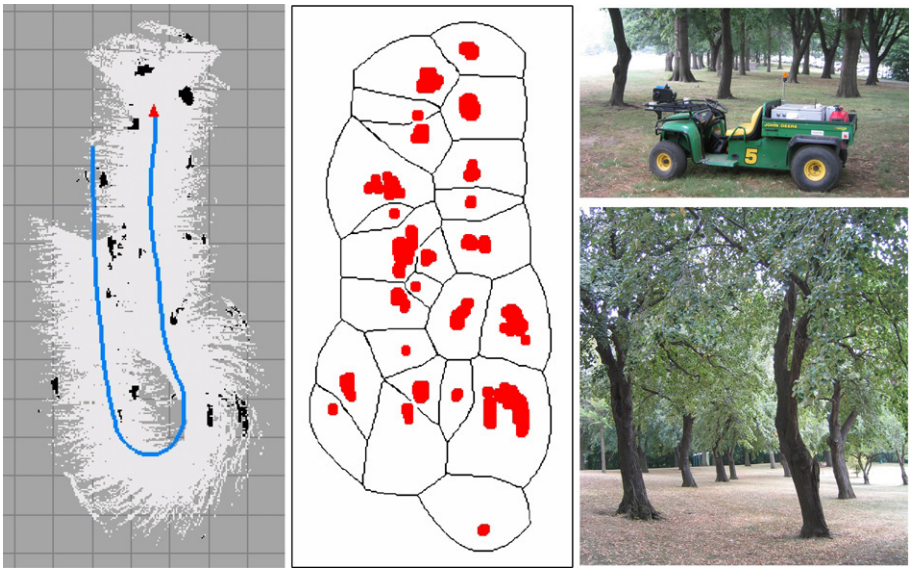

Fig. 1. An outdoor environment traversed by one of our John Deere e-Gator robotic platforms. *(Left)* The map generated by the robot (white is traversable area, black is untraversable, gray is unknown, and the grid has 5 m spacing). *(Center)* The GVD constructed from this map by Dynamic Brushfire with C-space obstacle expansion. *(Right)* The e-Gator and the environment in which these maps were generated.

2. Related work

The Voronoi region of an obstacle O is the set of points whose distance to O is less than or equal to their distance to every other obstacle in the environment. The GVD of an environment is the intersection of two or more Voronoi regions. Each of these intersection points is equidistant from its two (or more) closest obstacles. Several methods exist for computing the GVD. First, the GVD can be constructed in continuous space. For example, Nageswara et al. [11] represent the obstacles as simple polygons and geometrically construct the GVD, and Choset et al. [4] simulate an agent "tracing out" the GVD as it moves through the environment. Second, the GVD can be constructed in discrete space, *e.g.* on grids. In this paper we focus on GVDs constructed on grids because of the prevalence of grid-based environment representations in mobile robot navigation. Here, the Voronoi regions and the GVD are computed over the finite set of grid cells.

Researchers have used graphics hardware to generate grid-based GVDs very quickly [7]; however, this is often infeasible for mobile robot platforms with limited on-board hardware. Alternatively, fast hardware-independent algorithms exist for constructing GVDs on low-dimensional grids [1,2,12]. These algorithms require as input a binary grid and a mapping from each occupied cell in the grid to the obstacle to which it belongs (the latter information helps determine the boundaries between different Voronoi regions). In general, these algorithms scan the grid and compute for each cell its closest obstacle and its distance to that obstacle; those cells that are equidistant from two or more obstacles are included in the GVD. These algorithms run in linear time $O(mn)$ where m and n are dimensions of the grid. Thus, the computation required is a factor of the resolution of the representation rather than the complexity of the environment.

The first of these is the well-known *Brushfire* algorithm [1]. Brushfire is analogous to Dijkstra's algorithm for path planning, in that it processes cells in an *OPEN* priority queue, where decreasing priority maps to increasing distance from an obstacle. Initially, each obstacle cell in the environment is placed on the queue with a priority of 0 (the cell's distance to the nearest obstacle). Then, until the queue is empty, the highest-priority cell is removed, its neighboring cell's distances are computed, and any cell c whose distance dist_c has been lowered is updated to be on the queue with priority dist_c. The distance dist_c of each cell is approximated from the distances of its neighbors:

$$\text{dist}_c = \min_{a \in Adj(c)} [\text{distance}(c, a) + \text{dist}_a], \tag{1}$$

where $\text{Adj}(c)$ is the set of cells adjacent to c (usually 4- or 8- connected) and $\text{distance}(c, a)$ is the distance between c and a (usually Manhattan or Euclidean distance). Brushfire only makes one pass through the grid but has the added expense of keeping a priority queue which usually requires $O(\log(x))$ time for operations, where x is the number of elements in the queue.

The quasi-Euclidean distance transform algorithm developed by Rosenfeld et al. [12] makes two sequential passes through the grid, from top to bottom and then bottom to top. For each cell encountered, it simply performs an 8-connected search of neighboring cells to determine the cells' distance and nearest obstacle using Eq. (1). The Euclidean distance transform algorithm by Breu et al. [2] constructs the GVD by determining which Voronoi regions intersect each row in the grid. For each cell in a particular row, it computes the exact distance to the obstacle that is closest to it.

Although these algorithms are fast, they provide no mechanism for incremental reconstruction. We compare their performances with Dynamic Brushfire on several example robotics scenarios in Section 4.

3. The dynamic Brushfire algorithm

Just as Brushfire is analogous to Dijkstra's algorithm for planning, Dynamic Brushfire is analogous to the unfocused D^* family of efficient replanning algorithms [14,9]. When new information is received concerning the environment (*e.g.* from a robot's sensors), these algorithms only propagate the new information to portions of the map that could be affected. Thus, they avoid unnecessarily reprocessing the entire state space. In the grid-based GVD context, new information consists of obstacles being asynchronously added and removed from the environment, in whole or in part. When an obstacle cell is removed, the distances increase for exactly those cells that were closer to it than to any other obstacle cell. When an obstacle cell is added, the distances decrease for exactly those cells that are closer to it now than to any other obstacle cell. Dynamic Brushfire is efficient because it determines and limits processing to only cells within these two sets.