Automonomous Exploration for 3D Map Learning

Dominik Joho, Cyrill Stachniss, Patrick Pfaff and Wolfram Burgard

University of Freiburg, Department of Computer Science, Germany
{joho, stachniss, pfaff, burgard}@informatik.uni-freiburg.de

Abstract. Autonomous exploration is a frequently addressed problem in the robotics community. This paper presents an approach to mobile robot exploration that takes into account that the robot acts in the three-dimensional space. Our approach can build compact three-dimensional models autonomously and is able to deal with negative obstacles such as abysms. It applies a decision-theoretic framework which considers the uncertainty in the map to evaluate potential actions. Thereby, it trades off the cost of executing an action with the expected information gain taking into account possible sensor measurements. We present experimental results obtained with a real robot and in simulation.

1 Introduction

Robots that are able to acquire an accurate model of their environment are regarded as fulfilling a major precondition of truly autonomous mobile vehicles. So far, most approaches to mobile robot exploration assume that the robot lives in a plane. They typically focus on generating motion commands that minimize the time needed to cover the whole terrain [1,2]. A frequently used technique is to build an occupancy grid map since it can model unknown locations efficiently. The robot seeks to reduces the number of unobserved cells or the uncertainty in the grid map. In the three-dimensional space, however, such approaches are not directly applicable. The size of occupancy grid maps in 3D, for example, prevents the robot from exploring an environment larger than a few hundred square meters.

Whaite and Ferrie [3] presented an exploration approach in 3D that uses the entropy to measure the uncertainty in the geometric structure of objects that are scanned with a laser range sensor. In contrast to the work described here, they use a fully parametric representation of the objects and the size of the object to model is bounded by the range of the manipulator. Surmann et al. [4] extract horizontal planes from a 3D point cloud and construct a polygon with detected lines (obstacles) and unseen lines (free space connecting detected lines). They sample candidate viewpoints within this polygon and use 2D ray-casting to estimate the expected information gain. In contrast to this, our approach uses an extension of 3D elevation maps and 3D ray-casting to select the next viewpoint. González-Baños and Latombe [5] also build a polygonal map by merging safe regions. Similar to our approach, they sample candidate poses in the visibility range of frontiers to unknown area. But unlike in our approach, they build 2D maps and do not consider the uncertainty reduction in the known parts of the map.

The contribution of this paper is an exploration technique that extents known techniques from 2D into the three-dimensional space. Our approach selects actions that reduce the uncertainty of the robot about the world and constructs a full three-dimensional
model using so-called multi-level surface maps. It reasons about the potential measurements when selecting an action. Our approach is able to deal with negative obstacles like, for example, abysms, which is a problem of robots exploring a three-dimensional world. Experiments carried out in simulation and on a real robot show the effectiveness of our technique.

2 3D Model of the Environment

Our exploration system uses multi-level surface maps (MLS maps) as proposed by Triebel et al. [6]. MLS maps use a two-dimensional grid structure that stores different elevation values. In particular, they store in each cell of a discrete grid the height of the surface in the corresponding area. In contrast to elevation maps, MLS maps allow us to store multiple surfaces in each cell. Each surface is represented by a Gaussian with the mean elevation and its uncertainty $\sigma$. In the remainder of this paper, these surfaces are referred to as patches. This representation enables a mobile robot to model environments with structures like bridges, underpasses, buildings, or mines. They also enable the robot to represent vertical structures by storing a vertical depth value for each patch.

2.1 Traversability Analysis

A grid based 2D traversability analysis usually only takes into account the occupancy probability of a grid cell – implicitly assuming an even environment with only positive obstacles. In the 3D case, especially in outdoor environments, we additionally have to take into account the slope and the roughness of the terrain, as well as negative obstacles such as abysms which are usually ignored in 2D representations.

Each patch $p$ will be assigned a traversability value $\tau(p) \in [0, 1]$. A value of zero corresponds to a non-traversable patch, a value greater zero to a traversable patch, and a value of one to a perfectly traversable patch. In order to determine $\tau(p)$, we fit a plane into its local 8-patch neighborhood by minimizing the $z$-distance of the plane to the elevation values of the neighboring patches. We then compute the slope and the roughness of the local terrain and detect obstacles. The slope is defined as the angle between the fitted plane and a horizontal plane and the roughness is computed as the average squared $z$-distances of the height values of the neighboring patch to the fitted plane. The slope and the roughness are turned into traversability values $\tau_s(p)$ and $\tau_r(p)$ by linear interpolation between zero and a maximum slope and roughness value respectively. In order to detect obstacles we set $\tau_o(p) \in \{0, 1\}$ to zero, if the squared $z$-distance of a neighboring patch exceeds a threshold, thereby accounting for positive and negative obstacles, or if the patch has less than eight neighbors. The latter is important for avoiding abysms in the early stage of an exploration process, as some neighboring patches are below the edge of the abysm and therefore are not visible yet (see Fig. 1 (a)).

The combined traversability value is defined as $\tau(p) = \tau_s(p) \cdot \tau_r(p) \cdot \tau_o(p)$. Next, we iteratively propagate the values by convolving the traversability values of the patch and its eight neighboring patches with a Gaussian kernel. For non-existent neighbors, we assume a value of 0.5. The number of iterations depends on the used cell size and the robot’s size. In order to enforce obstacle growing, we do not perform a convolution