POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Simultaneous exploration and mapping for fully autonomous vehicles: a mixed graph-mesh approach

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Pietro Tenani

Advisor: Prof. Lorenzo Mario Fagiano

Co-advisors: Danilo Saccani, Michele Bolognini

Academic year: 2020-2021

Abstract

The focus of this thesis is the development of a novel controller for mobile robots used in exploration and mapping tasks, in outdoor and non GPS-denied environment. Specifically, the controller is run on a multicopter equipped with RGB-D camera and localisation sensor.
The proposed controller solves the problem of real-time autonomous exploration and mapping, with real-time obstacles reconstruction. The controller uses a navigation graph for collecting information about the obstacle-free area and computing paths, and a triangular mesh for representing the obstacles and selecting the best target.
The controller is composed by two loosely interacting data flows. The Mapping part merges the images generated by the camera into a global pointcloud, that is converted into a mesh. Frontiers are computed and the best one is selected by weighting the distance and the expected information gain factors. The Exploration part generates a local convex polytope that represents free space, integrates new nodes into the global navigation graph, and plan the path that reaches the node closer to the received target. Additional measures on the safety of the path and edge cases are implemented as well.
The thesis introduces novelties in a variety of domains, namely the study of the use of mesh in exploration and mapping duty, a novel approach for the generation of a convex polytope, a passive obstacle avoidance technique, and a rarely studied frontier definition.
The proposed controller is successfully tested on a simulated environment in Gazebo, and compared with a state of the art exploration and mapping framework.

1. Controller structure

1.1. Background

The proposed controller originated from G-BEAM [1], a Graph Based Exploration and Mapping controller that focus on building a map of environment using a drone that moves on a plane (at fixed height) equipped with a planar lidar. The proposed controller used G-BEAM as a starting point, and the skeleton of the data flow and the subdivision between nodes has been expanded and improved.

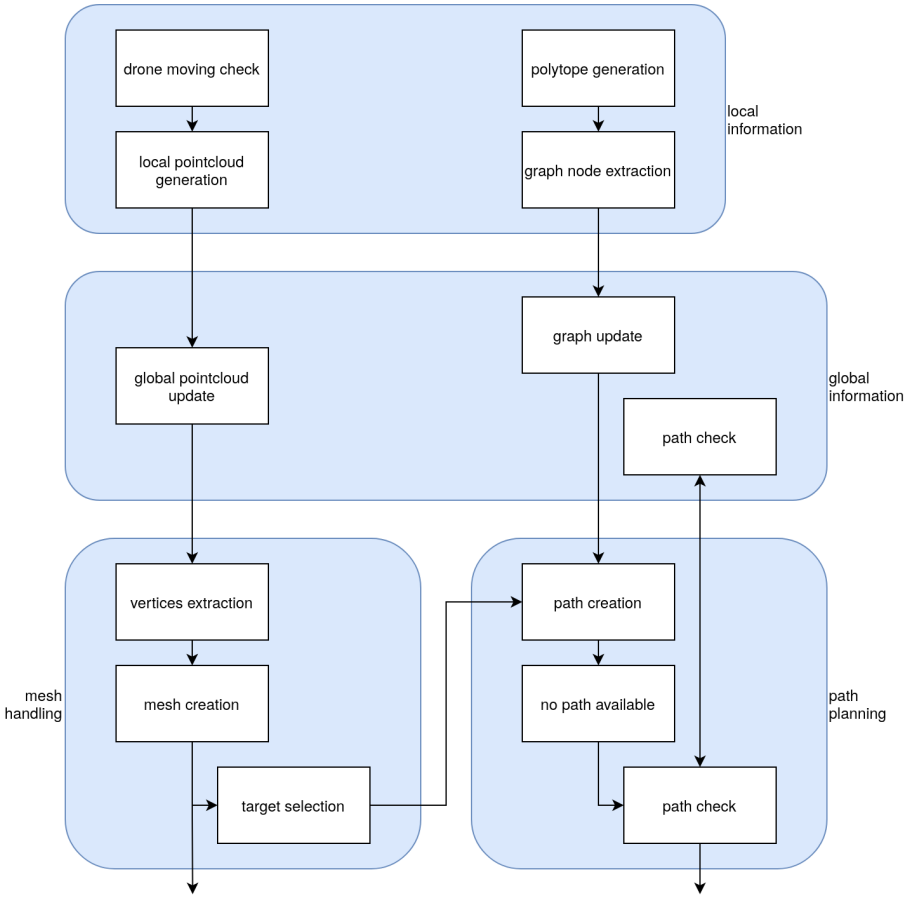


Figure 1: Data flow

1.2. Controller description

The proposed controller is composed by two main workflows: the Mapping part and the Exploration part. The Mapping task is performed by building and maintaining a mesh that represents the obstacles. The mesh is also used for selecting the next target to be filmed. The Exploration task is performed by building and maintaining a navigation graph representing the free space, and performing path computations and security checks. Each workflow is kept as independent as possible from the other, allowing for an higher level of optimization. The data flows and the subdivision between different modules can be seen in Figure 1.

1.3. Innovative contributions in the controller structure

The most important innovative contribution introduced in the structure of the controller is the use of two independent data structures for representing obstacles and free space. The usual approach consists on the adoption of a single data structure (usually an occupancy grid), or a secondary one built from the principal. The proposed approach allows to optimise each step, and use innovative data structures specialised either on Mapping or Exploration

2. Obstacle representation

2.1. Mesh management

The obstacles are stored in a triangular mesh. In order to build the mesh, the pointclouds received from the camera are filtered and merged into a global obstacle pointcloud. From the global pointcloud the set of mesh vertices is extracted. The number of vertices in the mesh (i.e. the density of the vertices) is a tradeoff between the quality of the reconstructed mesh (having more vertices results in a higher precision), the computational time (having more vertices requires more computational resources), and the sensitivity to noise (more vertices results in a more noisy surface if the readings are not errorless). Finally, once the vertices are extracted, the mesh

can be reconstructed. The chosen algorithm for building the mesh is the Greedy Projection Triangulation algorithm [3], an efficient meshing algorithm specialized in fast reconstruction. The computed mesh is provided to the user both as a classic coloured mesh and as a wireframe structure.

2.2. Target selection

The reconstructed mesh is used for computing the next best target that optimise the distance/usefulness tradeoff. In order to find the best target, the frontier set is determined (the frontier is defined as the "border" of the mesh, or more formally as the set of mesh edges that are contained in a single triangle), then the frontiers are grouped by continuity into holes, that are possibly filtered. Finally, for each candidate target (the set of edges that compose the holes) the number of nearby frontier points is counted. The candidate target that maximise a score function (directly proportional to the number of near edges, inversely proportional to the distance of the candidate) is set as next target.

2.3. Pointcloud drift

A problem encountered during the testing of the controller causes the partial pointclouds to be misaligned with the ground truth model, thus making the global pointcloud (and the reconstructed mesh) very noisy and mostly useless. The cause of the problem is not entirely known, and it appears to be present mainly during fast rotations and translations.
The solution adopted consists on checking at every pointcloud integration if the drone is rapidly moving or rotating, and in case the partial pointcloud is discarded. A drawback of this simple solution is the need for the drone to stay still in order to update the obstacles, therefore every few seconds the drone has to stop and the exploration time gets a lot longer.

2.4. Innovative contributions in the Obstacle representation functionality

The use of a mesh in a Exploration and Mapping approach is quite unexplored, as this data structure is usually considered not flexible enough. Therefore the successful use of a mesh, with the aid of a global pointcloud, shows a promising and flourishing research direction.
Moreover, the technique adopted for the computation of the frontiers has very few similar works in literature. The usual approach consists in focusing the exploration on the Free areas that are near Unknown zones, and therefore trying to maximise the explored volume. The proposed approach instead only considers the obstacles, and it tries to minimise the unknown boundary of the reconstructed mesh. This is too an interesting research path, that is especially beneficial when the goal of the exploration is a known obstacle.

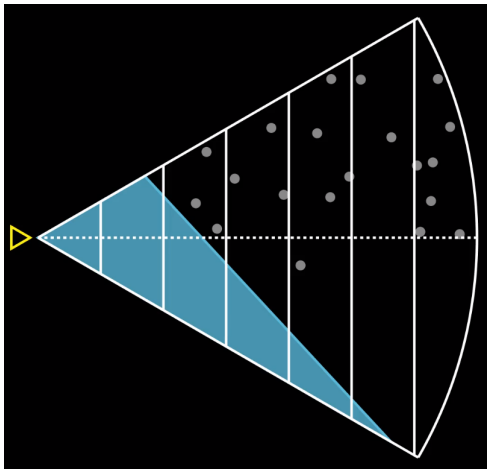


Figure 2: Convex polytope generation

3. Graph management and Navigation

3.1. Convex polytope

In order to build a navigation graph, used for representing the walkable space, it is necessary to extract from each computed frame a volume containing the maximum amount of free space. This is achieved by using a convex polytope. The novel algorithm developed for the thesis starts from the biggest right pyramid, and then iteratively shrinks it in order to exclude all the obstacles from the polytope. In Figure 2 is shown the polytope generated in a two dimensional case. The dots are the obstacles, the white cone is the field of view, and the blue shape is the convex polytope.

3.2. Graph construction and navigation

The graph is enriched of new nodes at each new acquired pointcloud. The nodes are extracted from inside the polytope, and inserted into the graph if and only if they comply the minimum distance between nodes. The vertical minimum distance between nodes is shorter than the planar distance, so that the inherent greater challenge of vertical movements is overcome.
For the navigation in the graph the well known Dijkstra algorithm is adopted, with a simple variation that avoids certain edges and nodes marked as prohibited. The destination node in the graph is chosen as the closer to the target produced by the frontier selection algorithm that is reachable from the starting node.

3.3. No new path edge case

A special edge case is handled with care. This is when the best destination node has been reached, but the mesh target can not be seen. In order to avoid a deadlock, the problem is handled by first completely exploring the current node, by performing a complete rotation (thus ensuring that any information from the current node is taken), and then computing a new path after excluding the current node from the possible destination nodes (thus forcing the exploration and ignoring the current best node, known to be useless). The list of forbidden nodes is deleted at each target change.

3.4. Passive obstacle avoidance

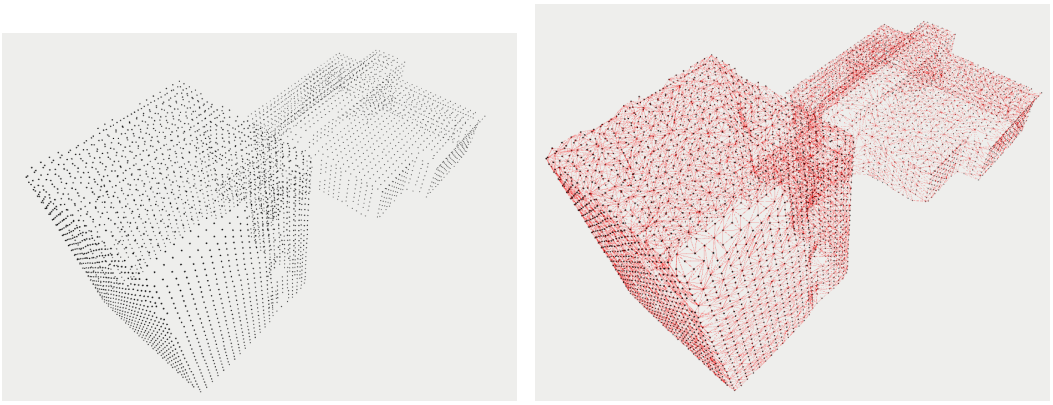
In order to ensure that the paths computed are at a safe distance from the obstacles, a passive obstacle avoidance approach is adopted. Once a path is computed, it is verified that all the nodes and the entirety of all arcs are distant enough from the obstacles. If this is verified path is accepted, otherwise the insecure nodes and edges are marked as non walkable and the path is recomputed.

3.5. Innovative contributions in the Graph management functionality

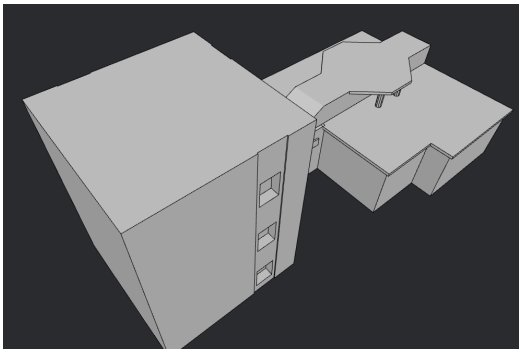
The first innovative contribution is the algorithm for computing the three dimensional convex polytope containing free space. Not much research has been done in this direction, and the proposed approach is well optimised, has a known complexity upper bound (as each obstacle is parsed at most once), and the polytope covers more internal volume than many simpler methods.
The adoption of a passive obstacle avoidance system is quite innovative too, since the most common approach consists on the use of a low-level active obstacle avoidance system that directly interact with the flight controller, thereby making unpredictable the trajectory followed. The proposed approach, instead, ensures that the drone can safely follow the exact computed path, and among other alternatives has much higher performance.

4. Testing results

The proposed controller has been developed in C++ using ROS, and tested in a simulated environment in Gazebo. The goal of the robot is to autonomously map a large building (26 x 13 m, and a maximum height of 11 m) both in absence and presence of sensor noises. The results have been analysed both visually and through some metrics.
In Figure 3 the original model, the reconstructed pointcloud and the mesh wireframe are presented.
The average total flight time is roughly 70/75 minutes. The running time for each module allows for a real time execution, in particular the most demanding task is the meshing computation, that requires half a second in the noiseless case, just above one second in the noisy case. In the first two columns on Table 1 are visible some metrics, in particular the number of data points and the distance values, measured by computing the mean, median and standard deviation of the distances between the mesh and the ground truth. The noiseless case is clearly better, however both the results are very good.
The meshing algorithm does not work perfectly with small and thin objects, as the pointcloud fails in reproducing the complexity of the surface (an example of a shelter with small columns present in the simulated environment can be seen in Figure 4, where the black cloud is the global pointcloud, the yellow squares are the mesh vertices and the red lines are the reconstructed mesh edges). However in most the cases mapping buildings with no small details results



(a) Mesh vertices (b) Mesh wireframe



(c) Ground truth

Figure 3: Mesh reconstruction

in an almost perfect reconstruction.

4.1. RTAB-Map comparison

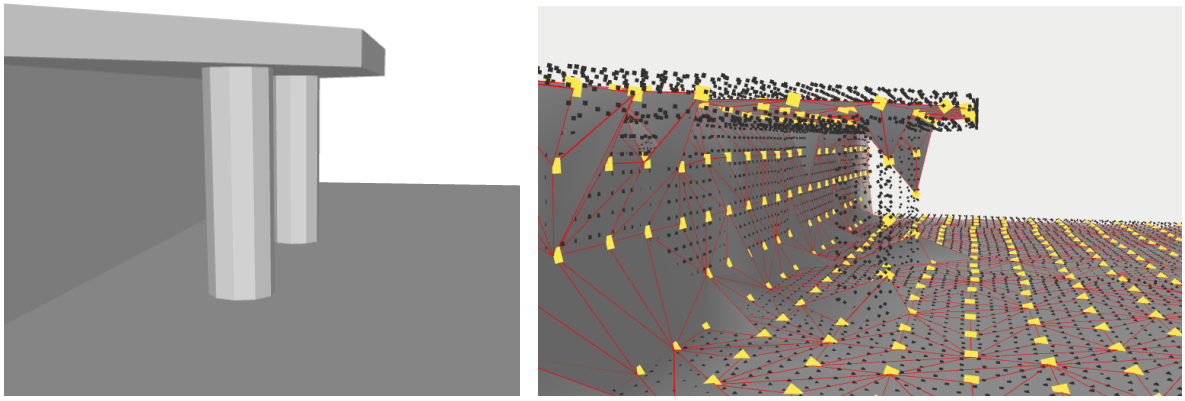
Finally, the proposed controller is compared with a state of the art SLAM approach, RTAB-Map [2]. The two algorithms are tested at the pointcloud level, comparing the global obstacle pointcloud of the proposed approach and the cloud published on the topic `rtabmap/cloud_map` for RTAB-Map.
The RTAB-Map pointcloud's distance metrics with respect to the ground truth are visible on the third column of Table 1 (note that RTAB-Map has been tested without sensors noise, so the values can only be compared with the first column of the table), while the distance metrics between the two reconstructed pointcloud are shown in Table 2.
Although storing one fourth of the points of RTAB-Map, the proposed controller provides very similar results, with the smallest difference between the two clouds. Moreover, the proposed approach does not perform SLAM and therefore is much faster, so in situations that does not require Simultaneous Exploration and Mapping the proposed approach can be preferred over RTAB-Map.

		no noise	noise	RTAB-Map
N_{obs}	[-]	**118273**	**285917**	**447568**
d_{max}	[m]	0.342	0.507	0.548
d_{μ}	[m]	0.048	0.101	0.042
d_{σ}	[m]	0.043	0.062	0.033
d_{med}	[m]	**0.039**	**0.083**	**0.035**

Table 1: distance metrics

5. Conclusions

Simultaneous exploration and mapping for fully autonomous vehicles, adopting a mixed graph-mesh approach that keeps the data flows as independent as possible.
The Mapping task is tackled by building a mesh of the obstacles mapped during the flight and using it for extracting the frontier and the best target to film. The Exploration task is tackled by building a navigation graph that stores informa-



(a) Ground truth (b) Reconstructed model

Figure 4: Detail of the shelter and columns imperfections

d_{min}	[m]	0.0005
d_{max}	[m]	0.438
d_{μ}	[m]	0.042
d_{σ}	[m]	0.020
d_{med}	[m]	**0.041**

Table 2: comparison between RTAB-Map and proposed controller pointclouds

tion about the free navigable space. Some special procedures have been designed for handling special cases: when no new path are available, the pointcloud drift, and the obstacle avoidance. The controller has been tested on a simulated environment and compared with a state of the art algorithm. The results are very promising, the reconstructed mesh is very similar to the ground truth, and the proposed algorithm has similar output and is more efficient than a state of the art algorithm selected for the comparison.
Regarding the future of the research, a real world testing would be the most interesting direction, while further developments could include a collaborative multi drone approach, the possibility to take advantage of prior knowledge at the beginning of the mapping process, and the expansion to a full SLAM algorithm.

References

[1] Leonardo Cecchin, Danilo Saccani, and Lorenzo Fagiano. G-beam: Graph-based exploration and mapping for autonomous vehicles. In *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1011–1016. IEEE, 2021.

[2] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[3] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.