Tailoring 3D Mapping Frameworks for Field Robotics*

Ferreira, Francisco do C., Zaera, Martí, Karfakis, Panagiotis T. and Couceiro, Micael S.

Abstract— **Mapping is an essential part for the adoption of robots in agricultural and forestry environments. Providing the robot with the ability to map its surroundings, facilitates its navigation and is necessary for implementing obstacle avoidance, without human interference. Herein we present the challenges of outdoor environments, present an overview of existing mapping frameworks and then evaluate their suitability for field applications. Two widely used mapping frameworks, OctoMap and RTAB-Map are analyzed within the Robot Operating System (ROS) ecosystem and a parametric study is carried out in order to assess their performance, under both simulated and real-world constraints. Finally this work aims to be utilized as a deployment reference guide for mobile robotic applications in outdoor environments.**

I. INTRODUCTION

In field robotics, systems are developed to replace humans in laborious, repetitive and dangerous tasks, with a goal of higher workflow efficiency and lowering associated costs. Industrial applications rely on structured conditions and have some indicative representation of the environment (e.g, factory floorplan) available, prior to robot deployment. In non-structured environments, this can only be achieved if the mobile robots are endowed with mapping capabilities, be it to find a path from its current location to a given target location, or to properly identify obstacles, people and other relevant surrounding features [22].

Initially, we start by addressing the challenges inherent to mapping in field robotics, defining a set of key performance indicators that need to be taken into account when selecting or developing a mapping framework (Section II). Then we describe and theoretically compare the most widely adopted mapping frameworks compatible with the Robot Operating System (ROS) (Section III).

From this comparison, we select OctoMap and RTAB-Map as the most promising robotic mapping frameworks, capable of tackling the challenges of field applications (Section IV). An experimental comparison of both methods is supported by the parametric study of the methods, in terms of simulation and real-world experiments (Section IV).

This work contributes to the preparation of a properly tailored mapping framework for field robotics. We investigate which is the the most prominent mapping framework and evaluate it on our mobile robotic platforms in a simulated and real world application scenarios, and finally provide our results to serve as guide for future applicability within the robotics domain.

*This work is co-funded by the program Portugal 2020, under both SEMFIRE (ref. CENTRO-01-0247-FEDER-032691) and SAFEFOREST (ref. CENTRO-01-0247-FEDER-045931) R&D projects and the by the European Union's Horizon 2020 research and innovation programme 5GSmartFact (under the Marie Skłodowska-Curie grant agreement ID 956670).

II. OUTDOOR MAPPING

Mapping for robots operating in outdoor environments has been studied for several decades. Despite many years of research, many problems are not robustly solved yet, due to difficulties in dealing with ever changing environmental conditions, the homogeneity of natural landscapes, the diversity of natural obstacles and especially the effect of vibrations or external forces, among other technical challenges [19, 12, 24].

Field robots are required to operate under **irregular terrain** and deal with dynamic obstacles [20]. For that, robots need to be able to generate maps taking into account parameters such as update capability [17], flexibility [1], loop closures [6] and sharing efficiency [11].

As maps are typically used for navigation, localization is a key factor. The robot is equipped with a set of sensors, capable of acquiring relevant information of its surroundings and eventually determining its own location within the map. This process is known as Simultaneous Localization and Mapping (SLAM). Perception limitations exist, such as maximum sensor ranges and potential obstructions from obstacles which are common problems in laser and visual methods.

All sensors have **measurement noise** that can be caused by the robot motion, reflecting surfaces, signal obstruction and interference from other external effects which are difficult to model. This calls upon the need to implement filters to reduce these errors, such as the Kalman Filter [26], Extended Kalman Filter [25], Unscented Kalman Filter [18], Particle Filter [23] and graph-based optimization [5], among others. Measurement noise poses a great challenge to robotic mapping operating in outdoors.

Another challenge within robotic applications, lies in the **high dimensionality** of map generation. The size and detail of the map can have a heavy toll on computational resources, such as memory and processing power and affect overall map generation frequency and quality. The higher the resolution the more time is required to generate it, as the features are more and their processing is intensive.

One of the hardest problems in robotic mapping is the correspondence problem, also known as **loop closure** or data association [2, 6, 9]. This concept consists of the robot being able to recognize a place previously visited via the detection of landmarks [2] or features. These landmarks are set during the exploration of the environment. The robot sets nodes during the exploration and compares a new node with previously detected nodes to detect feature similarities. Finding enough correspondences can eventually lead to a higher mapping accuracy, which reduces the accumulated

errors in the map.

However, some application scenarios may have similar landmarks at different locations that may mislead the robot's perception, i.e. passing through two different lanes in a vineyard. Furthermore, **static landmarks** may also change in outdoor environments, be it due to day-night cycle, where illumination is changing, or the effects of seasons, where flora appearances and soil conditions change. By moving large obstacles to different locations, affects the landmark recognition process as they can alter the scene association process. Other factors include removal or destruction of previously mapped obstacles due to bio-degradation, fires, floods and rock slides.

Dynamic environments can make the mapping process very difficult. The examples given previously are effects that happen over a long period of time. Others changes, however, may happen more quickly, like a gate that was previously open but now closed while revisiting that same location. Even faster changes may occur due to moving entities, like people, animals, vehicles and other robots. This problem aggravates the loop closure problem, making it even harder to solve [20].

Updatability is key to have a stable representation of the environment. This requires establishing a format that is compact and fast to update the map's representation with new information [15]. It also adds validity to the previously recorded representation, without influencing the future received data [17]. Most ways to solve this problem, are by dividing the environment into subsets of homogeneous characteristics based on geometric structures such as, occupancy maps [15, 7], grids [8, 3], graphs [7] and trees [15].

Share-ability of maps is also an important feature in multi-robot systems. The literature has been addressing this under cloud robotics and robotic cluster domains. In field robotics, cloud robotics might not be viable due to the fact that, in remote locations, we may not have access to a network connection, e.g. mining, forestry etc. This is why the robotic cluster concept is often employed in field robot applications. This approach focuses on sharing the maps between robots directly, sharing their processing capabilities within the robot cluster with an overall aim to solve complicated computational problems. This promotes locally distributed processing from individual robots to solve high-dimension of problems [11].

III. MAPPING FRAMEWORKS

A mapping framework is a collection of software packages that allows to run mapping algorithms. In our case we are focusing on ROS-compatible mapping frameworks. ROS has been chosen since it is the most widely employed open-source meta-operating system for robotics, which provides hardware abstraction, low-level device control, message-passing between processes, and package management [13]. ROS encompasses multiple tools and libraries which are constantly updated by the robotics community, with mapping frameworks being abundant. However, whether or not those mapping frameworks are in a deployable state for challenging domains, such as agricultural or forestry applications, is still unclear. Table I compares the most popular mapping frameworks, addressing their readiness for adoption and dealing with the challenges stated in the previous section.

GMapping is considered due to its popularity in indoor applications[14, 16]. Yet, as a 2D SLAM approach, GMapping is unable to cope with the irregular terrain inherent to outdoor environments. The Elevation_Map is a framework that provides a map with 2.5 dimensions, i.e. it is a 2D map with height information [8]. Compared to GMapping, the Elevation_Map can be generated based on 3D LiDAR or a RGB-D camera data and is able to cope with the irregularity of outdoor terrains. Furthermore, Elevation_Map uses Kalman Filters to reduce the measurement errors. Nevertheless, the Elevation_Map cannot work well under dynamic environments, being unable to fully represent certain environments, such as flying or suspended obstacles, such as bridges. The representation provided by the Elevation_Map framework is enriched when combined with the Traversability_Map [3].

The Traversability_Map is an excellent source of information for robot navigation algorithms, since it provides a cost value to each point along the four cardinal directions and their diagonals. With the recursive nature of multi-layer perception, the Traversability_Map's processing power requirements grow exponentially with the dimension of the map. Furthermore, as opposed to other approaches, the Traversability_Map changes according to the robot's current position, making it very difficult, if not impossible, to implement a global map, shareable by multi-robot systems. Therefore, while this framework has been often used as a local map[4], it cannot be used as a global map capable of tackling all the aforementioned challenges.

Cartographer [10] uses local SLAM that builds sub-maps to feed navigation systems and uses global SLAM that runs on the background, being capable of detecting loop closures [9]. It uses Extended Kalman Filter algorithms to address the measurement noise and it is able to perform range odometry using point clouds, while fusing the odometry from multiple sources. Despite widely used, this framework presents a few weaknesses, namely its inability to work well under dynamic environments and very detailed environments and is computationally intensive.

OctoMap [15] is a framework based on octrees and is highly efficient in terms of 3D model storage, being able to record huge environments in compact maps. OctoMap uses a particle filter and a probabilistic approach to cope with the measurement noise and continuously updates the map in a efficient manner. Due to its octree representation, it can be used as a a global mapping method for multi-robot systems, allowing robots to efficiently share maps to each other.

RTAB-Map [7] is a mapping framework that can also internally integrate OctoMap for a more efficient representation of 3D environmental maps. RTAB-Map can use a variety of sensors such as stereo cameras, LiDAR sensors and RGB-D cameras and create pointclouds of the environment. RTAB-Map encompasses a loop closure detector that uses a bag-of-words for feature detection and matching within images and

TABLE I

COMPARISON OF MAPPING FRAMEWORKS

Framework	Irregular Terrain	Noise Filtration	High Dimensionality of the Entity	Loop Closure	Dynamic Elements	Updatability	Muti-Robot System
GMapping		X	X	X	X	X	X
Elevation Map	X	X	X	X		X	
Traversability Map	X	X		X	X	X	
Cartographer	X	X		X		X	X
OctoMap	X	X	X		X	X	X
RTAB-Map	X	X	X	X	X	X	X

laser data. Although it is an optimized mapping approach , RTAB-Map still requires a high processing capability to create maps.

From Table I, it is clear that both OctoMap and RTAB-Map seem to be the most promising mapping frameworks for generating global maps which are essential in mobile robotic applications. Both provide shareable multi-session 3D maps that can be well-suited for outdoor applications. The extent and degree they can tackle the challenges inherent to field applications is addressed in the next section.

IV. PARAMETRIC STUDY: OCTOMAP VS RTAB-MAP

This section describes the OctoMap and RTAB-Map frameworks and establishes a parametric study with the objective of obtaining a global map. The heuristic approach we follow is the generation of a global map without errors, despite the loss of descriptive information about minor obstacles. This study has been supported by numerous experiments, carried out under both simulation and real-world experiments.

In each experiment, we tailor the parameters at hand through trial and error while taking into account their effect on the map creation process, in order to obtain a more accurate and efficient map representation of the environment. It is noteworthy that no memory management has been implemented in RTAB-Map in order to make a fair comparison between the two packages even though this can be detrimental on medium scale maps.

A. OctoMap

It is a probabilistic mapping framework implemented as a library of algorithms in C++ and also as a ROS package [15]. The package called `OctoMap` is used as a core and also depends of `OctoMap_ros` and `OctoMap_msgs` to provide messages, wrappers and conversion methods. The map is built by `OctoMap_server` and can be visualized with the `octovis` tool.

OctoMap subscribes to an incoming 3D point cloud for scan integration and provides:

- The free and occupied space compacted as an OctoMap binary stream;
- The centers of all occupied voxels as a point cloud;
- The projected 2D occupancy grid representations.

The maps can be merged making use of `marble_mapping` for multi-robot applications.

OctoMap can be tuned using 38 parameters to improve the quality of the map, taking into account the robot, the environment and the mission. Through our several experiments, the parameter `sensor_model/max_range`, `pointcloud_min_z` and `pointcloud_max_z` had to be reduced in order to avoid detecting slopes that were not an obstacle, but were considered as such by OCTOMAP. These parameters, by default, are set to the maximum range of the sensor itself.

The `sensor_model/min` and `sensor_model/max` are used to determine the probability of clamping while dynamically building a map. We have increased this threshold slightly to achieve better results.

The `occupancy_min_z` and `occupancy_max_z` are limits of height that OCTOMAP will consider to map and, therefore, they were defined based on the physical properties of the robot. The `filter_ground` parameter allows to configure the ground filter by enabling `ground_filter/distance`, `ground_filter/angle` and `ground_filter/plane_distance` to affect the map representation. The `ground_filter/distance` is the parameter that will set the minimum height of an obstacle. `ground_filter/angle` will set the angle threshold for points (in z direction) to be segmented to the ground plane. The `ground_filter/plane_distance` sets the minimum distance between a plane to the z=0.

By manipulating these three parameters, the robot will consider as ground any plane within this threshold. Therefore, these have been tuned based on the physical traversability of the robot.

B. RTAB-Map

RTAB-Map is an open source library implementing loop closure detection with a memory management approach, that limits the size of the map so that loop closure detections are always processed within a fixed time window, thus satisfying online requirements for long-term and large-scale environment mapping [7].

The ROS package subscribes to RGB-D, Stereo or 3D LiDAR sensor data topics. It is also able to estimate visual robot odometry or use that information as an input, thus being able to provide:

- RTAB-Map's graph, information and latest node data.
- 2D occupancy grid generated with laser scans or image data.
- 3D point cloud map.
- Convenient way to show graph's labels in RVIZ.

RTAB-Map can also provide an octree-based map by making use of the OctoMap library. The package also

implements 13 nodelets integrated to synchronize, compress or decompress images, along with conversions and merging options between the different types of inputs.

Obstacle detection can be also implemented by a nodelet, so that segmenting the obstacles and the ground from a point cloud is possible. All the maps can be saved to a database, allowing multi-session mapping and post optimizations by the standalone application.

One of the advantages of RTAB-Map is the number of configurations that can be used to better suit to the robot, environment and mission, with a total of 464 parameters grouped in 42 types for locatization, mapping, feature matching, filtering and general robot-centric management. The most relevant ones for mapping applications were obtained during the simulations and real-world experiments carried out. We then have been configured and tuned them gradually, leaving the any remaining ones in their default value.

The main parameters have been configured to fit within the robot´s specifications, using point clouds as an input and not publishing any transform. An approximate synchronization is used because we cannot guarantee that the inputs from cameras, LiDARs and odometry will be perfectly synchronized at all times.

The parameter `map_filter_radius` and `map_filter_angle` are the minimum distance and angles between the nodes to be added to the map, respectively. We decrease the number of matches between two different readings with the `Vis/MinInliers` parameter to improve loop closure detection. The `RGBD/LocalRadius`, `RGBD/ProximityMaxPaths`, `RGBD/ProximityPathFilteringRadius` and `RGBD/ProximityAngle` are increased significantly from their default values to make faster loop closures and are related to the robot's scale.

With `RGBD/AngularUpdate`, we have increased the minimum angle for an update of the map to minimize the misreadings while the robot is rotating. We gave more importance to the linear updates with the `RGBD/LinearUpdate` parameter. We want to update the map from the newest nodes to help navigation systems, ignoring the transformation of the map frame to the odometry frame. This can be done by setting `RGBD/OptimizeFromGraphEnd` to true.

The `RGBD/OptimizeMaxError` was set to zero because we do robust optimizations and we do not want to reject loop closures based on the optimization error ratio. The `RGBD/StartAtOrigin` allows to deal with the *kidnaped robot problem*, which is the problem of moving the robot while its not mapping or turned off, from current position and the mapping is restarted.

The `RGBD/CreateOccupancyGrid` parameter was set to true to create local occupancy grid maps. `RGBD/MarkerDetection` has been tuned to generate a better graph optimization. `RGBD/ProximityOdomGuess` was also set to true to help proximity detection. `RGBD/ProximityGlobalScanMap` was set to true to help with loop closures. Going through trial and error, we have found that the best map generated was with kNN-FlannNaive, which we selected by setting `Kp/NNStrategy` to 0.

We have selected the g2o strategy in the `Optimizer/Strategy` parameter since it led to a more optimized graph for our case study. Better results have been observed by setting `Optimizer/VarianceIgnored` to true. `Optimizer/Robust` allows to have a robust optimization to deal with the outdoors conditions.

`Grid/RayTracing` allows to generate the empty space between obstacles detected and the robot. `Grid/DepthDecimation` has been increased to solve an error while optimizing the grid. `Grid/3D` has been set to true to be able to see 3D representations and use the OCTOMAP library internally.

`Grid/FromDepth` has been set to false since point clouds have been used to generate the map. `Grid/CellSize` has been increased to generate the map faster, being able to optimize the map without losing a significant amount of data. `Grid/FlatObstacleDetected` was set to true to help the ground segmentation. We have found that with a `Grid/NoiseFilteringRadius` at 0.35 and a `Grid/NoiseFilteringMinNeighbors` at 5 (default) to reduce the robot rotation noise. Bear in mind, however, that these two parameters are dependent and the relationship between them will influence directly the robot rotation noise. We increased the range `Grid/RangeMax` because we are using point clouds with higher range, thus allowing to map larger environments faster.

`Grid/MaxObstacleHeight` has been increased to make sure that we do not map obstacles that the robot can go under. `Grid/MaxGroundHeight` is maximum height that RTABMAP will consider as ground, i.e. where ground segmentation is applied. `Grid/MaxGroundAngle` maximum angle of the normal to be considered as ground. Because we do not want to use any type of cluster filter, we have set `Grid/MinClusterSize` to 0. `Grid/NormalK` defines the amount of normals that can be considered to apply normal segmentation.

The parameter `GridGlobal/Eroded` was set to true help deleting noise identified as small obstacles. `GridGlobal/MinSize` has been set to 30 to coordinate with our navigation system, this parameter will create at the start of a new map an empty map 30 by 30. `GridGlobal/ProbHit` and `GridGlobal/ProbMiss` have been tuned to increase the probabilities of hit or miss, increasing these parameters by trial and error method lead to a better defined map. With `Reg/Strategy` we use ICP since both camera and LiDAR are used as inputs.

The `Marker/Dictionary` selects dictionaries to choose the settings for the detection of markers. We chose ARUCO dictionaries because it was the one we had installed for other purposes (e.g. autonomous docking). We choose 12 (7x7_50) since it corresponds to the size of marker (7x7) and the number of markers (50) to consider. The less markers we have, the more distance we have between them. This allows to place landmarks in large environments.

`Marker/CornerRefinementMethod` has been set to true to use corner refinement methods considering our environment.

The above list of parameters although exhaustive are essential to achieve better maps using the framework and need to be tuned for the application at hand accordingly.

V. EXPERIMENTAL RESULTS

This section compares the performance of OctoMap and RTAB-Map. The evaluation encompasses a qualitative comparison of the maps produced by both frameworks to infer the level of correspondence between the real world and created maps. Moreover, a quantitative analysis is also carried out, comparing the 2D occupancy map generated by each framework with the ground truth information.

A. Simulations

In the first phase, we have evaluated OctoMap and RTAB-Map by testing both approaches in a realistic simulation environment. The Gazebo Agriculture Simulation has been used for our experiments, which fits our requirements of being an outdoor scenario with irregular ground and several obstacles[1].

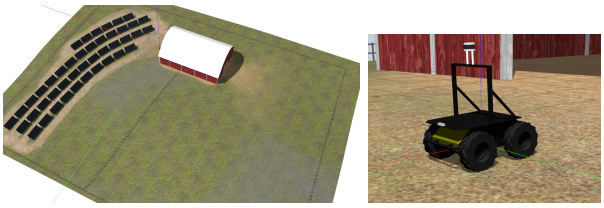

Fig. 1. Agriculture simulated scenario in Gazebo (left) and husky model with 3D LiDAR (right).

The ground truth has been obtained by recording a map using perfect odometry and sensor readings straight from the simulator. Alongside the map, a rosbag containing command velocities provided to the robot to the i.e. `/cmd_vel` topic, has also been recorded. This is particularly useful whenever one wants to reproduce different experiments under the same conditions, ensuring that the robot can repeat the same trajectory for both frameworks.

Clearpath Robotics' Husky is used as the robotic platform in the simulation experiments, being equipped with a 3D LiDAR, a RGB-D camera and an Inertial Measurement Unit (IMU).

Figure 2 depicts the maps obtained using the OctoMap and RTAB-Map frameworks. The average error and the average accuracy when compared to the ground truth map are presented in Table **II**. The error represents the number of cells that do not correspond to the ground truth, i.e. the sum between the number of false positives and false negatives. Put it differently, it is the number of cells that the mapping framework identifies as occupied when they were actually empty, and the other way around. The accuracy is the number of cells that correspond to the ground truth divided by the total number of cells (333099), i.e. the sum between true positive and true negative divided by the total number of cells.

[1]https://clearpathrobotics.com/blog/2020/07/clearpath-robots-get-new-gazebo-simulation-environments/

TABLE II
RESULTS FROM THE SIMULATION EXPERIMENTS.

Framework	Error	Accuracy
OCTOMAP	11848	0.96443
RTABMAP	15264	0.95291

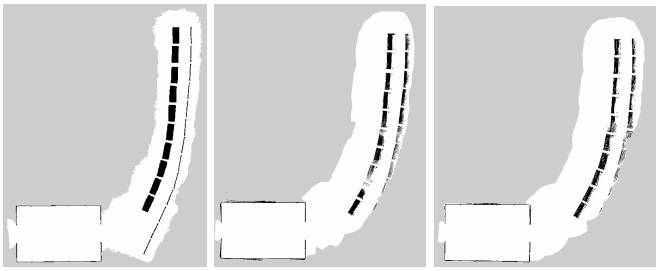

Fig. 2. The map of the ground truth used as the benchmark reference on the left, and the maps obtained by the simulation experiments, OctoMap in the center and RTAB-Map right.

Despite the fact that it is realistic simulated environment with integrated physics of Gazebo, one may still consider it far from ideal since it does not encompass dynamic objects and the terrain is relatively flat in the area of the test. Under these conditions, differences between OctoMap and RTAB-Map are found to be not significant (approximately 1%).

Both frameworks lead to minor mapping issues. Localization errors have been observed during the experiment under certain conditions, though both methods were able to also mitigate this problem without any problems.

B. Real-world

In similar methodology, real-world experiments were carried out at Ingeniarius facilities, in Alfena, Portugal. Since perfect odometry cannot be guaranteed in real-world scenarios, a first version of the map has been acquired and polished afterwards with the support of both Google Earth and in-field measurements.

The approach followed in the previous simulation experiments is to ensure repeatability of results, by feeding the robot with command velocities cannot be adopted here due to the non-linearities of the environment (soil degradation, dynamic obstacles, etc.) and the robot (battery voltage fluctuations, variable response time from low-level controller, etc). Because of these reasons, a new setup to compare both mapping frameworks has been established: for each trial, we roughly followed the same path by teleoperating the robot through a set of landmarks established a priori to the trials, with at a constant defined velocity of 0.2 m/s. The robot would map the surrounding area until the end of the route.

OctoMap and RTAB-Map were integrated in the Ranger robot (forestry compact loader) and in the VineBot (agriculture robot). Both robots were developed by Ingeniarius Ltd

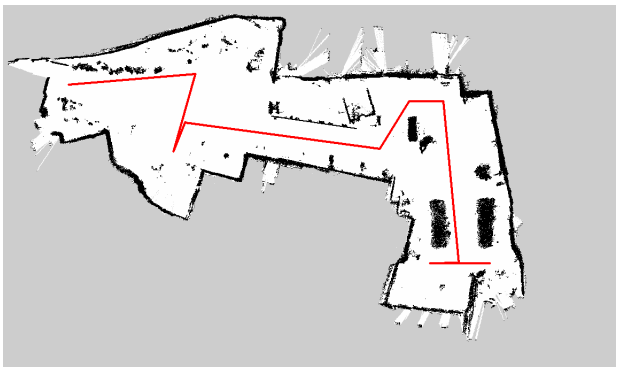

Fig. 3. The route that the robot followed in every test run.

following the same standards and similar hardware resources. Both are all-terrain tracked vehicles equipped with two 3D LiDARs, one RGB-D camera, and an IMU. The internal computer also shares the same hardware, including a Intel® CoreTM i7-8700T Processor (12M Cache, up to 4.00 GHz), a GTX NVIDIA Quadro P2200 5GB, and a SSD of 256GB. Point clouds acquired by the two LiDARs are combined into a single point cloud and fed directly to both OctoMap and RTAB-Map frameworks. As with the simulation experiments, the maps generated by both frameworks were qualitatively and quantitatively compared by considering ground truth information.

While the integration of the frameworks has been made to both robots being able to tune and check their adaptability to different robots with similar hardware, Vinebot has been used to present the results due its greater freedom of movement in the trial area and smaller size. The Ranger robot was used as a testbed for evaluation on to larger platforms and accelerating the experiments.

Fig. 4. VineBot robot used for the real world experiments (left) and Ranger used for optimizing the frameworks (right).

TABLE III

AVERAGE CPU USAGE DATA DURING EXPERIMENTS.

Framework	CPU	Memory
OCTOMAP	60.3%	19.2%
RTABMAP	73.6%	27.5%

CPU usage, real-world footage and maps have been recorded over the course of real world experiment to ensure repeatability and validation of results.

Both qualitative and quantitative results favor RTAB-Map over OctoMap. With a difference of approximately 4% in

TABLE IV

RESULTS FROM THE REAL WORLD EXPERIMENTS.

Framework	Error	Accuracy
OCTOMAP	36834	0.95256
RTABMAP	6141	0.99209

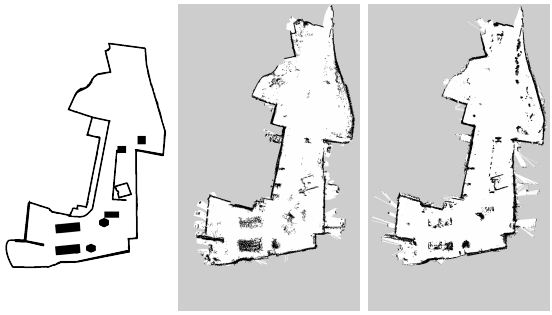

Fig. 5. The map of the ground truth used for the benchmark tool on the left, and the maps obtained by the real world experiments, OctoMap in the center and RTAB-Map right.

the accuracy, RTAB-Map depicts a superior performance in the outdoor scenario which includes dynamic objects and a more irregular ground with slopes above 20°, which were absent during the simulation experiments in section V-A. This difference in accuracy may severely affect the map representation and as a consequence affect the autonomous navigation of the robot. A noteworthy fact is that although the OctoMap error is higher, it generates maps faster than RTAB-Map. However, during rotations, OctoMap tends to accumulate more noise than RTAB-Map, which is only later on corrected once the area is re-mapped.

Moreover during the experiments, we noticed that both OctoMap and RTAB-Map were able to adapt to some sporadic dynamic obstacles. Besides humans, some unforeseen dynamic obstacles were also captured by the Vinebot sensors, in our case cats. OctoMap and RTAB-Map are able to clear these entities from the map after a while. Yet, we have noticed that OctoMap took longer to clean large dynamic objects (e.g. humans), leaving a trail long enough to affect path planners during autonomous navigation.

RTAB-Map also presents some minor drawbacks when facing large dynamic objects. However, these are represented as discontinuous noise instead of a continuous trail and are resolved by the next iterations. Additionally, it is important to highlight that RTAB-Map systematically updates while moving, as opposed to OctoMap. This implies that, whenever dynamic obstacles are tracked when the robot is not moving, these are not even registered in the map.

In both cases, noise filters are based in Bayes theorem [21], but only RTAB-Map's noise filters can be further tuned to reach a better result. Furthermore, the ground segmentation of RTAB-Map is more developed than OctoMap, offering a larger range of parameters that may be tuned, to better suit one's objectives. By combining these tools, it is possible to increase the mapping range, which is a critical feature for OctoMap that often struggles for larger mapping

ranges.

Some large obstacles are also not represented by OctoMap, namely the step shown in Figure 6. This 0.4m step is very important because it is not traversable by the Vinebot. However, if an autonomous navigation system would rely on OctoMap under this particular situation, it is likely that a plan would be established between its current position and the region identified in Figure 6, which would lead to serious damages to the robot. This tends to happen because both OctoMap and RTAB-Map filter plane objects, as ground depending on a series of properties. Nonetheless, RTAB-Map provides a considerably larger range of parameters when compared to OctoMap, which allows for better tuning. For instance, it is possible to see that RTAB-Map partially identifies this step in Figure 6.

Fig. 6. The important obstacle absent from the map generated by each framework. OCTOMAP (Figure 5 center map) it almost erases the line completely, and RTABMAP (Figure 5 right map) shows only the planters.

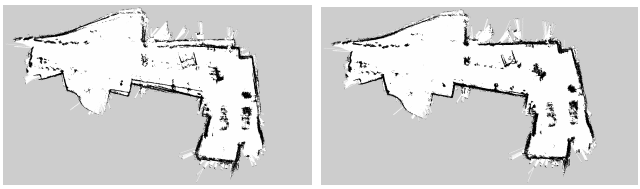

Fig. 7. Loop closure during real world experiments (in this case RTABMAP).

Another point to note is the CPU usage. As expected, OctoMap is lighter than RTAB-Map in terms of computing resources. With the Linux command `glances`, we monitored the CPU usage of both frameworks and reported the result in Table V-B. These data was only collected after optimizing both frameworks. The VineBot CPU was at 60% of its load capacity while mapping with OctoMap, rising to 70% under sporadic situations. The VineBot CPU was almost constantly at 72% of its capacity while mapping with RTAB-Map. During our real world experiments RTAB-Map was consistent, in terms of CPU usage (without spikes). Considering the processor and graphics card of the computer of our robot, this difference of 12% is not significant for the Vinebot. However, for lower-end robots, such as low-cost drones, using RTAB-Map might not be a possibility.

It has been found that OctoMap might not be ideally suited for robotic mapping in large unstructured and dynamic scenarios, it still presents an efficient approach to represent the 3D map using octrees, without draining onboard resources. Due to its efficiency in representing complex 3D maps, RTAB-Map integrates the OctoMap library as an alternative way of 3D map representation. Unfortunately, it does so without being as computationally efficient as OctoMap. The compromise to produce a better map quality with more computational resources, makes RTAB-Map ideally suited for complex outdoor applications in which such resources might be available within the robot.

VI. CONCLUSION

In this article, we sought out the overall best mapping framework for field robotics and performed in depth analysis and comparison on OctoMap and RTAB-Map.

Both frameworks come with parameterizable filters, ground segmentation and can map large areas. The RTAB-Map database requires more memory than OctoMap. They have strategies to solve the correspondence problem, with RTAB-Map featuring strategies and configurations that applying loop closure detection. Both can address and map dynamic obstacles, although OctoMap struggles more to update the map, when large dynamic obstacles become absent, such as humans and animals. Both frameworks have the potential to be used in multi-robot systems, however RTAB-Map seems to call upon a more centralized solution using a common database. On the other hand, OctoMap's generated octrees are lightweight and can be easily shared between robots, which could lead to decentralized solutions more reliably than RTAB-Map.

In conclusion, RTAB-Map is a SLAM approach that shows better performance than OctoMap. The difference in the amount of parameters between these frameworks gives RTAB-Map an advantage, providing a better parametrization of the map. It allows several odometry types and multiple strategies for detecting features and graph optimization grant RTAB-Map tremendous flexibility when compared to OctoMap. Despite the minor differences between RTAB-Map and OctoMap in simulation (≈1%), greater discrepancies can be observed in the real world experiments wherein a more irregular and dynamic environment were put to the test. RTAB-Map is generally better suited to field applications and validated from our findings.

REFERENCES

[1] Nils Funk et al. "Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3553–3560.

[2] Michal Tölgyessy et al. "Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2". In: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: 10.3390/s21020413. URL: https://www.mdpi.com/1424-8220/21/2/413.

[3] Paolo Arena et al. "Learning traversability map of different robotic platforms for unstructured terrains path planning". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207423.

[4] Ahmad Kamal Nasir, André G Araújo, and Micael S Couceiro. "Localization and navigation assessment of a heavy-duty field robot". In: *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020)*. 2020, pp. 25–29.

[5] Baichuan Huang, Jun Zhao, and Jingbin Liu. "A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks". In: (Aug. 2019).

[6] Xingliang Ji et al. "LLOAM: LiDAR Odometry and Mapping with Loop-closure Detection Based Correction". In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2019, pp. 2475–2480. DOI: 10.1109/ICMA.2019.8816388.

[7] 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". In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446. DOI: https://doi.org/10.1002/rob.21831. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21831. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831.

[8] Tae Nam, Jae Shim, and Young Cho. "A 2.5D Map-Based Mobile Robot Localization via Cooperation of Aerial and Ground Robots". In: *Sensors* 17.12 (2017), p. 2730. DOI: 10.3390/s17122730.

[9] Wolfgang Hess et al. "Real-Time Loop Closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[10] Wolfgang Hess et al. "Real-Time Loop Closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[11] Bruno Duarte Gouveia et al. "Computation Sharing in Distributed Robotic Systems: A Case Study on SLAM". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2015), pp. 410–422. DOI: 10.1109/TASE.2014.2357216.

[12] Stephanie Lowry et al. "Visual place recognition: A survey". In: *IEEE Transactions on Robotics* 32.1 (2015), pp. 1–19.

[13] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[14] Joao Machado Santos et al. "A sensor fusion layer to cope with reduced visibility in SLAM". In: *Journal of Intelligent & Robotic Systems* 80.3 (2015), pp. 401–422.

[15] Armin Hornung et al. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[16] João Machado Santos, David Portugal, and Rui P. Rocha. "An evaluation of 2D SLAM techniques available in Robot Operating System". In: *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2013, pp. 1–6. DOI: 10.1109/SSRR.2013.6719348.

[17] Ahmad A. Masoud. "Motion Planning with Gamma-Harmonic Potential Fields". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.4 (2012), pp. 2786–2801. DOI: 10.1109/TAES.2012.6324661.

[18] Shurong Li and Pengfei Ni. "Square-root unscented Kalman filter based simultaneous localization and mapping". In: *The 2010 IEEE International Conference on Information and Automation*. 2010, pp. 2384–2388. DOI: 10.1109/ICINFA.2010.5512187.

[19] Alonzo Kelly et al. "Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments". In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 449–483. DOI: 10.1177/0278364906065543. eprint: https://doi.org/10.1177/0278364906065543. URL: https://doi.org/10.1177/0278364906065543.

[20] Denis F Wolf and Gaurav S Sukhatme. "Mobile robot simultaneous localization and mapping in dynamic environments". In: *Autonomous Robots* 19.1 (2005), pp. 53–65.

[21] Richard Swinburne. "Bayes' Theorem". In: *Revue Philosophique de la France Et de l* 194.2 (2004).

[22] Sebastian Thrun et al. "Robotic mapping: A survey". In: (2002).

[23] D.J. Salmond and H. Birch. "A particle filter for track-before-detect". In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 5. 2001, 3755–3760 vol.5. DOI: 10.1109/ACC.2001.946220.

[24] Chuck Thorpe and Hugh F Durrant-Whyte. "Field Robots." In: *ISRR*. 2001, pp. 329–340.

[25] K. Reif et al. "Stochastic stability of the discrete-time extended Kalman filter". In: *IEEE Transactions on Automatic Control* 44.4 (1999), pp. 714–728. DOI: 10.1109/9.754809.

[26] Greg Welch, Gary Bishop, et al. "An introduction to the Kalman filter". In: (1995).