Robots navigation and mapping using SLAM

CHETTOUR Hamza

Abstract—This paper presents the difficulties of mapping a mobile robot has to overcome. It presents a few techniques to use to solve them. And by mapping two simulated environments asses the performance of the rtab-maping techniques and presents some of its limitations.

Index Terms—Mapping, SLAM, RTAB-Mapping, GraphSLAM, Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

Mobile devices are amongst the most wanted and needed technological products by humanity. For their ability to move accurately and tirelessly and in many cases autonomously has become fundamental to accomplish tasks deemed difficult or even impossible for humans. We can give examples of space discovery where rovers have outrun us to roam first a whole planets, robots used to discover underground caves or even unmanned flying vehicles used to discover and photograph deserts to assess the evolution of the desertification.

Some real challenges and difficulties face every mobile robot. The most intuitive ones are the first to pop in mind when talking about navigation for robotics. When hearing these two keywords, one thinks directly about the capacity of the robot to move and follow a path which is the problem of actuators and motion, or thinks about the capacity of the robot to understand it's surrounding which is the problem of sensing and scene understanding, or the problem of thinking and planing which is the problem of the embedded software in relation with algorithms and decision making. Although all these problems are very challenging and researchers are still finding new techniques to better solve them everyday, they are common to all robots. The problems facing navigation for robotics are very special and are a layer built upon the ones discussed above. There are 3 mandatory abilities a mobile robot has to master: 1- The ability to localize itself in a map provided or constructed. 2-The ability for planning a path in the map. The elementary requirement for this path is to avoid obstacles but the required path to find can be more challenging like finding the shorter path to a desired location or the least dangerous path in an hostile environment. 3 - The ability to construct or at least update a map while navigating using information from sensors and actuators. You can see that these three abilities can't be achieved unless the ability to sense and move and decide are already acquired.The state of the art solutions in these three areas are efficient enough for the majority of applications and are sufficient to build upon to solve the navigation problems and seek the three abilities discussed. This paper focuses on mapping challenge and covers some techniques used to solve it.

2 Background

Any localization or path finding effort is meaningless without a map. In the simpler versions of the problem we can assume that the environment is static and won't change during the whole mission of the robot and thus go with the hypothesis that a map furnished at the start will always match the ground truth map.

But in the general cases, the environment is unknown at first or is dynamic and changes a lot making this assumption invalid. And even if the environment can be considered static the most conservative approach is to construct and update a map nonetheless to enable the robot to adapt to sudden changes. Also, comparing a freshly constructed map to a prior assumed true map will help reduce uncertainties due to measurements and motion errors.

This establishes the first statement : estimating a pose or a path needs a known map.

On the other hand, how can any estimation of a map using inputs from sensors and odometery be possible without linking these informations to a landmark. At the very best the robot can construct a local map of its surrounding but any linkage to a general or global map will need an accurate estimation of the localization where this local map was constructed.

This establishes the second statement : estimating an accurate map needs a known pose.

And here we are facing a famous chicken or egg problem.

Trying to estimate one part until we can assume that it can be considered as known may be possible in some cases but has been proven inefficient regarding both computation power and memory and also estimation error and accuracy. For example one can try to estimate the whole map using a map linked to each particle in the AMCL algorithm and of course filtering the particles if done the right way will keep the most probable map and thus an estimated map. But having a map as an attribute for each particle will consume huge amounts of memory and will need a serious CPU/GPU power to filter all these informations. Also, the particles generated won't be enough to cover a sufficient area of the probability distribution because each map has a great number of attributes.

To simplify, if you are trying to estimate a state of a 6 faced dice using particle filters you'll have to either generate enough particles or if the number of particles is limited go through enough filtering loops. You can't expect an accurate result if you go through only 2 loops and generate 1 or 2 particles while the features are of number of 6 for example. In addition nothing guarantees that the particle with the right estimation will appear. Actually since the number of attributes and features is high the probability of such particle never appearing is high.

The solution that can come handy in such case is a combination of local map estimation that will be used for a global map estimation, and a pose estimation relative to this map. This process is composed of an alternating steps : estimating a map , estimating a pose relative to it, reestimating the map that will be used to estimate the next poses and this goes on. This is called SLAM or (Simultaneous localization and mapping). The simultaneous aspect doesn't come from the chronology of the steps but from the fact that during the whole mission the map and poses are enduring a continuous estimation.

For the step of estimating a pose, It can be seen as estimating a pose assuming a known map because the prior map was given in the step before, and thus we obtain a classical localization problem which is not treated in this paper. On the other hand estimating the map given a prior poses or path can be discussed.

2.1 Occupancy grid mapping

The occupancy grid mapping is a way to discretize the world. The environnement is evenly divided into 2D cells and each cell can be in one these stats : occupied, free, unknown. The algorithm is probabilistic because the stat of each cell must be estimated using the prior informations and the data from sensors. The most used filter to estimate it is the Bayesian filter that will use the prior assumption on the state, the assumed known poses and the measurements to give a state of occupancy.

2.2 GraphSLAM

GraphSlam is a SLAM algorithm that uses graphs to estimates both maps and poses while consuming data from sensors.

A graph is a representation of a system where relation between two components of this system can be described using a link.

In Slam we can give at least two main components : a localization and a and object of the map or a a feature. The link between two components is often called a constrain because it limits the space solution and the state of of the system modeled can't be explained unless this constraint is taken into account.

Intuitively, the link between two poses is a motion and the link between a pose and a feature is a measurement.

These links are flexible to account for the uncertainty of motion and sensor measurements.

Thus, after modeling the whole system an optimization happens afterwards to minimize the uncertainty by trying to find an equilibrium of the tension of these links.

After obtain a graph that contains all the poses and features the robots has seen, the solution that will explain it or in other world the estimated stat of the world has to fit

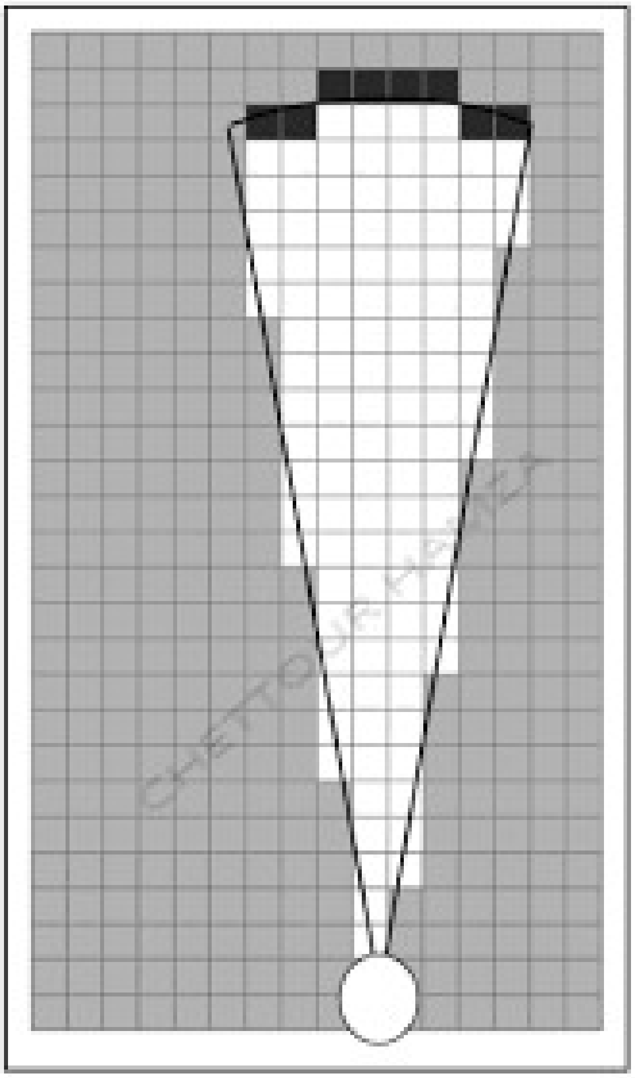


Fig. 1. Occupancy Grid map. White : free, Black: occupied, Grey : unknown

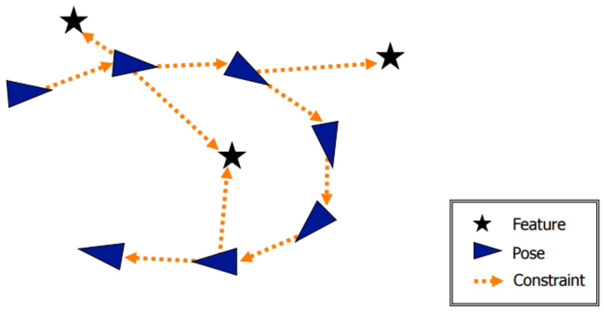


Fig. 2. GraphSlam

this graph. Simply, an estimation of the map has to comply with all the nodes and constraints of the graph. If the graph says there is a feature measured at a certain distance after a certain pose, the map has to comply with it. Seen differently , each node and constraint gives an information and we want the best map that will explain this outcome. This is exactly what the likelihood algorithm does. And this is the last step of the GraphSlam by which using the maximum likelihood estimator, a map is generated that verifies the best the graph constructed by the robot.

2.3 RTAB mapping

RTAB mapping or Real-Time Appearance-Based Mapping is a graph based approach to SLAM using visual data from an RGB-D camera in the estimation.

Rtab-mapping is composed of a front end and a back end. In the front end the graph is constructed and constraints link the nodes.The focus here is on how to use the sensor data to prepare the graph. The graph constructed is different from the other graph slam approaches by the fact that no landmark/feature nodes are drawn and only odometery and loop closure constraints are needed. Loop closure detection refers to verifying whether a location is new or is similar to a previously mapped area. Finding these matches increases the accuracy of the map constructed and also the accuracy of the localization since the robot is able to identify its pose relative to what it sees while understanding that it was there before.

In the back end the graph is optimized and the 2D and 3D map are generated.

3 Scene and robot configuration

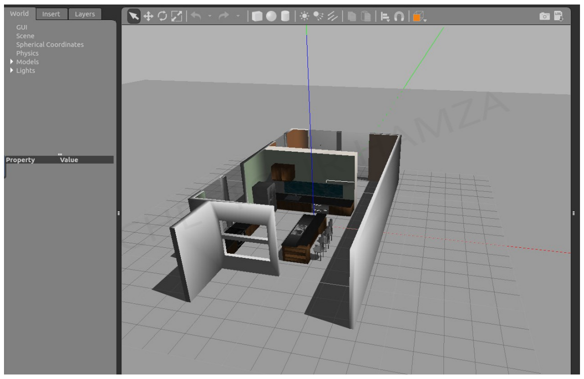
3.1 GAZEBO world

To test Rtab-map 2 environments were used. The first is the pre-built kitchen environment where the robot is asked to map a flat composed of 2 rooms, a kitchen and a living room. This environment is relatively easy to map since the flat is not very big, there isn't many similarly looking locations and the number of features is not very high.

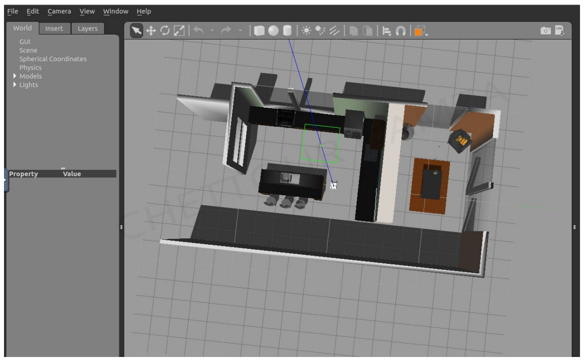
A second world was designed to challenge more Rtab mapping under ROS. This environment was divided to three main areas to test three kinds of difficulties of mapping a new environment. The first area is composed of similarly looking objects and features but irregularly placed. The objects that were chosen are 2 kind of trees. The first one is smaller and cover the middle space of the area and the second is a larger kind with larger roots that bump out of the soil and present a difficult obstacle to detect and thus to map. Fig.5

In the second area we tried to represent a similarly looking objects that are regularly placed in the spaced to increase the confusion. The object chosen are a pillar construction positioned just after the first area. Fig.6

The third area symbolize a noisy environment with a lot of bumps in the floor and irregular shapes of objects and walls. A pre-built destroyed fire-station exists in gazebo and was chosen to represent this environment.Fig.7



(a) Far view



(b) Top view

Fig. 3. Kitchen world

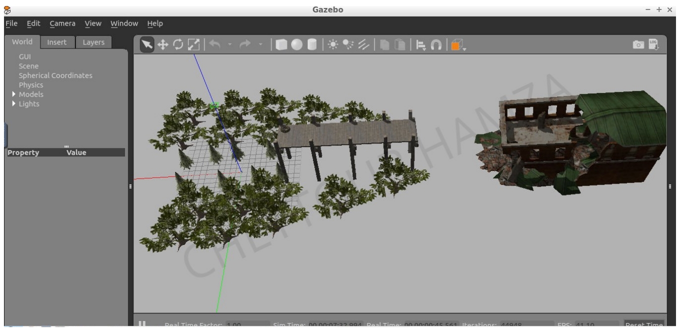


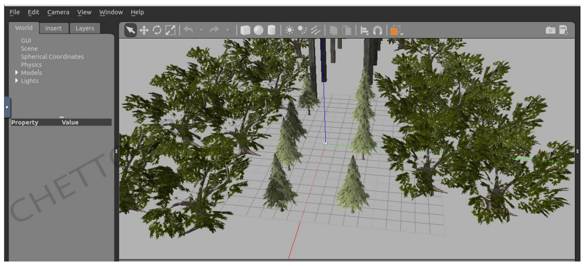
Fig. 4. Second world tow view.

3.2 Robot configuration

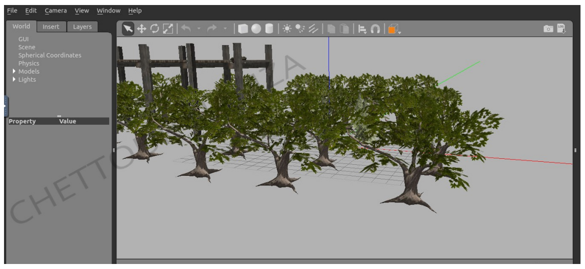
The robot used is the same robot used in a previous paper about the problem of localization. It is build around a 0.3x0.4x0.1 meters platform. The robot is 4 wheeled. Each wheel is a 0.05 length an 0.1 radius cylinder connected to the platform by a continuous joint.

The RGB-D camera is attached to the front of the robot while the laser is placed on a 0.1 length and 0.02 radius cylinder support to ensure that the laser beams are not obstructed by the wheels as shown in the Fig.4. The laser is not used in this simulation and instead the data from depth camera was used to replace the scan informations.

This type of robot is using the skid drive. It means that the locomotion is achieved by controlling the velocity and rotation sense of each side independently. The revolution around the Z axis can be achieved by having the same velocity with a different orientation which causes the wheels

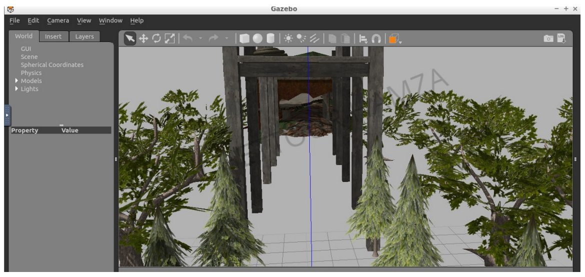


(a)

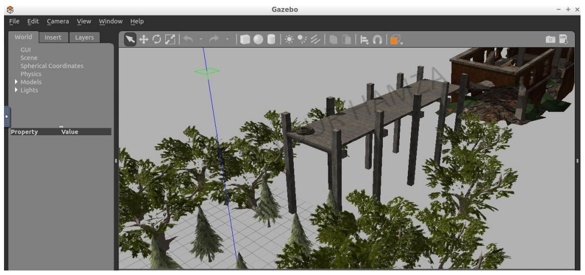


(b)

Fig. 5. First area: irregularly placed similar trees.



(a)



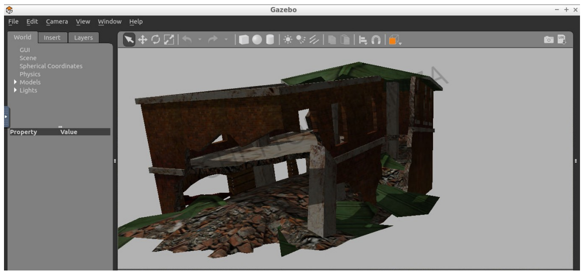
(b)

Fig. 6. Second area: regularly placed similar pillars.

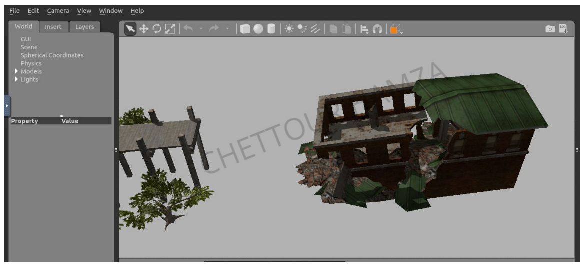
to slip on the ground and thus the robot can turn over without any translation.

By using 4 wheels the robot can achiever higher speeds and perform better curved motions.

One modification is essential to note that solves a problem encountered while mapping and visualizing the point cloud map. The frame orientation of the robot is not the same as RTAB and Rviz. And because of that the space scanning is not read properly by these 2. To correct this a joint and a link were added between the camera and the the base of the robot in order to rotate the joint and account of the error.

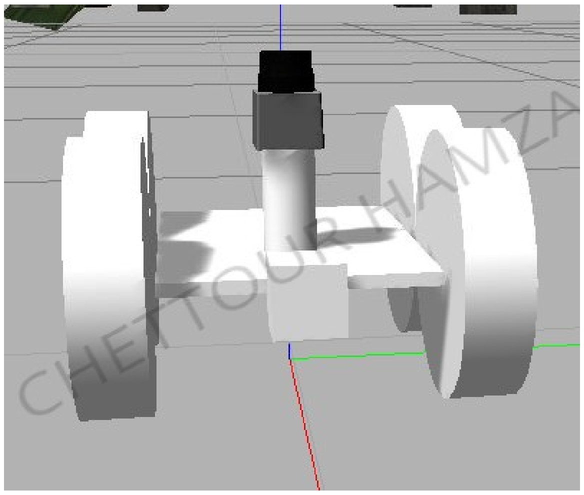


(a)

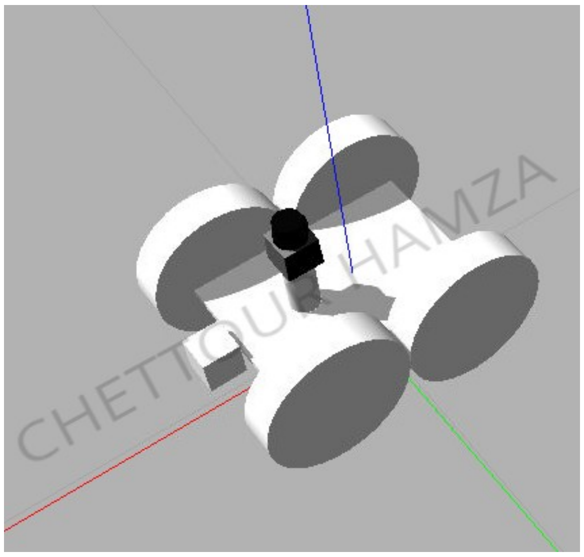


(b)

Fig. 7. Collapsed fire station



(a)



(b)

Fig. 8. Robot model

3.3 ROS packages

In this paragraph we discuss the main mackages used to test RTAB mapping under ROS.

The first one is the rtab-map package. It's the one that will do the hard lifting. As described on the official ROS wiki the package is a ROS wrapper of RTAB-Map (Real-Time Appearance-Based Mapping), a RGB-D SLAM approach based on a global loop closure detector with real-time constraints. This package can be used to generate a 3D point clouds of the environment and/or to create a 2D occupancy grid map for navigation. It contains many nodes and only 2 were used. The node called rtabmap is where the map is incrementally constructed and loop closure detected. The second node is rtabmapviz. It communicates with RTAB-MAP and has the same purpose of rviz with additional mapping visualization options.

The second package is depthimage_to_laserscan that has only one node that takes a depth image and generates a 2D laser scan.

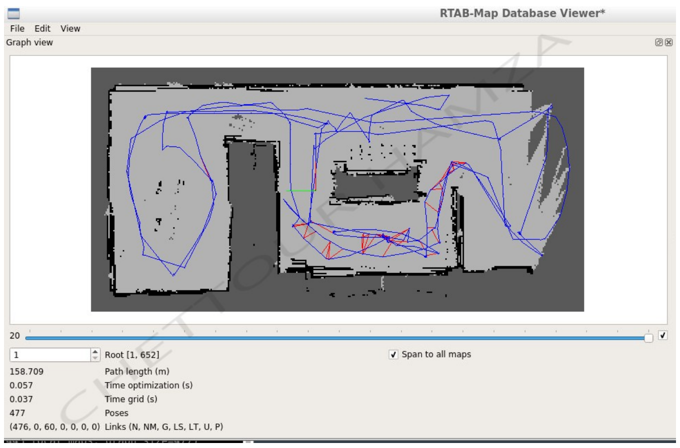
The other essential packages for this project and worth mentioning are the tf package that keeps track of the multiple frames and maitain the relationship between these frames in a tree structure, the joint_state_publisher that allows to publish the state of the joints given an URDF and the robot_state_publisher that takes the joint angles of the robot and published a 3D poses of the robot link to the other packages and this way the state and motion of the robot is tracked.

4 RESULTS

4.1 Kitchen world

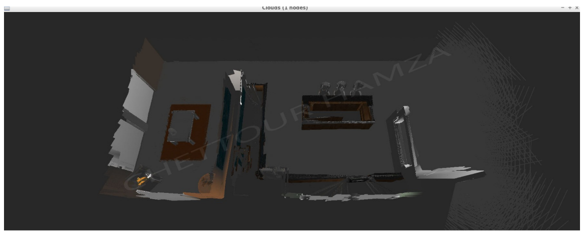
The kitchen world is relatively easy to map and unsurprisingly the robot has done well mapping it. Since the world is not very big we were able to navigate it three times over and this augment the loop closures probability and the details the robot sees. Using rtabmap-databaseViewer we can visualize the results: The figure 9 shows the grid map of the world and also that there was 60 look closure detections.

The figure 10 shows the reconstructed 3D map.

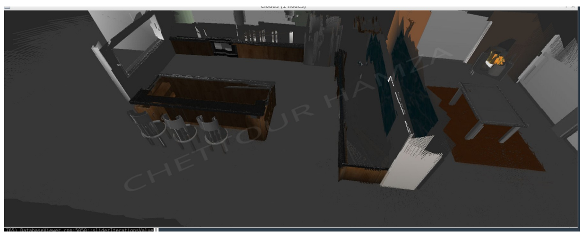


```
1                  Root [1, 652]
158.709            Path length (m)
0.057              Time optimization (s)
0.037              Time grid (s)
477                Poses
(476, 0, 60, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)
```

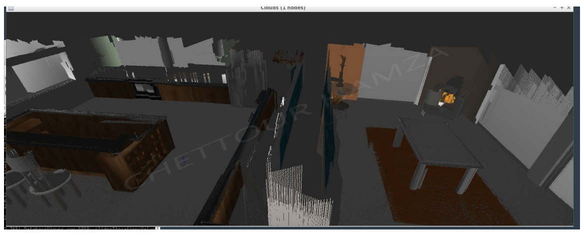
Fig. 9. Gridmap of the kitchen.



(a)



(b)



(c)

Fig. 10. 3D map of kitchen World

4.2 Designed world

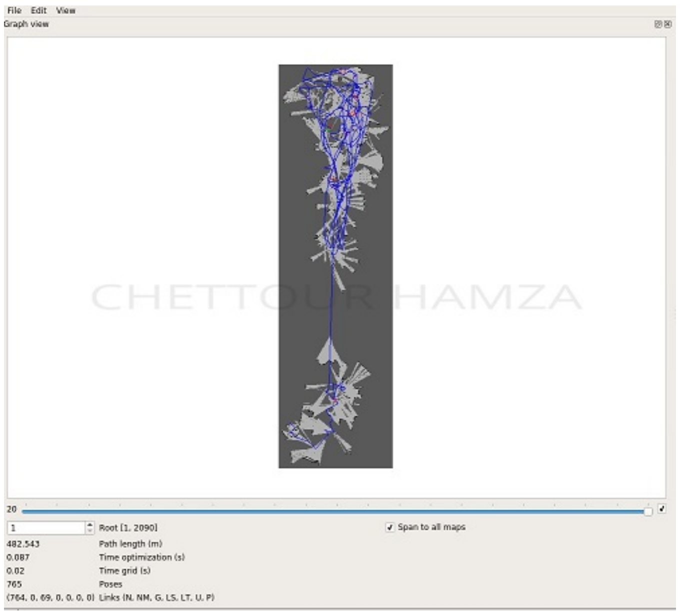
This world has been really challenging to map. The strategies to show the limitations and difficulties have worked very well. Figure 11 shows the grid map of the world. It shows two separate areas and no passage between the two. Also the upper area is very messy while the bottom one seems to make more sense. In the following we'll try to read them an give an interpretation of the results.

The robot seems to do quiet alright at first when mapping. We noticed that when the robot didn't encounter a second similar object the map was alright but as soon as it encounters it the map becomes messy. When using a model from gazebo we guarantee that the objects are identical. When using two trees of the same model they are 100 per cent same looking.

Figure 12 show the confusion of the robot when encountering a second tree. In Fig12.a the robot was at the first tree. When it shifts to the second we can see how the map changes and gets messed up. Fig12.b

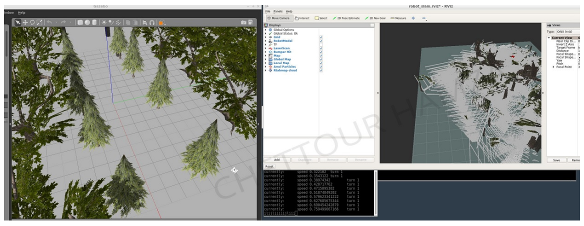
Same observation around pillars. Fig 13 shows that even when the robot advances in the corridor, because of the pillars the robots thinks that it has revisited the same spot. This is why in the grid map the corridor didn't appear. We can see in Fig 14 that no pillars appear when traveling from the area 2 to the area3 (the fire station). They are all thought to be in the same location (entrance of the area 1).

Unlike the area 1 and area 2, area 3 or the collapsed fire station doesn't have redundant objects. It's noisy and has a lot of bumps in the soil, irregular shapes, a small hill, but all these don't seem to affect the mapping like the similarity in

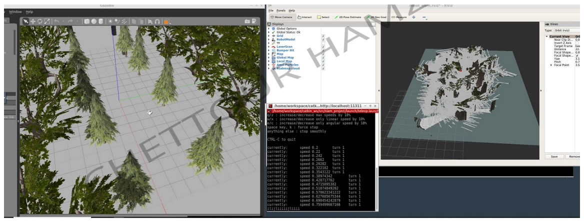


```
1                    Root [1, 2090]
482.543              Path length (m)
0.087                Time optimization (s)
0.02                 Time grid (s)
765                  Poses
(764, 0, 69, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)
```

Fig. 11. Gridmap of the second world.



(a)



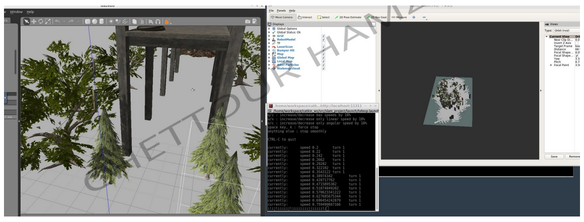
(b)

Fig. 12. Confusion around trees.

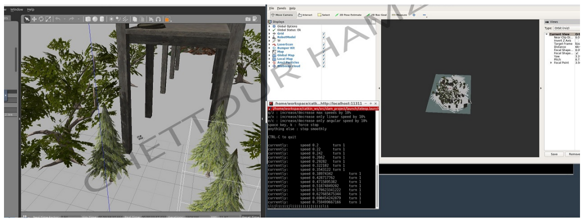
looks does. Fig 15 shows the performance of the mapping. It's quiet alright.

5 Discussions

In the result part we've seen the most influential characteristics that messed the mapping was the redundant objects. The kitchen world didn't have many (only the tree chairs) but they were too close to each other so the robot knew they were different objects since they appear in the same images many times. The second world has many similar trees and pillars and it has to travel some distance to visit them, add to that the similarity and you got the confusing resulted. What we couldn't really asses is if the regularity of the position of the objects played a role in this confusion. The robot was confused in both of the areas. From our intuition we think that the irregularly placed similar objects are less confusing since the irregularity of position is also a data the robot can



(a)



(b)

Fig. 13. Confusion around pillars.

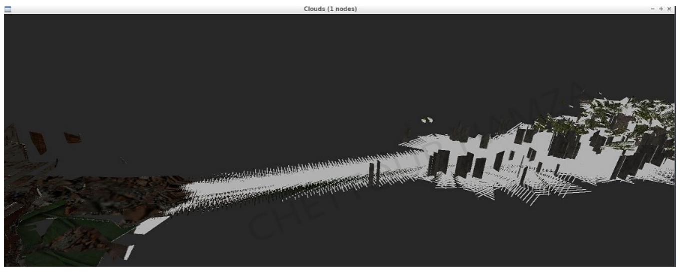


Fig. 14. Road between the area 2 and 3.

use to understand that this is a new item or a previously encountered one.

Due to time constraints only the default rtab-map were used. In a future work what would be interesting is to study how each parameter can be used to solve this challeng. Maybe reducing the probability of having a loop closure can help.

Another idea that can be crazy or maybe useful, it has to be studied in a future work. It's to map with front camera but also a back camera. This way the robot can take into account the scene it faces but also the scene it leaves. But unless it improves a lot the performance adding a camera is not an elegant solution.

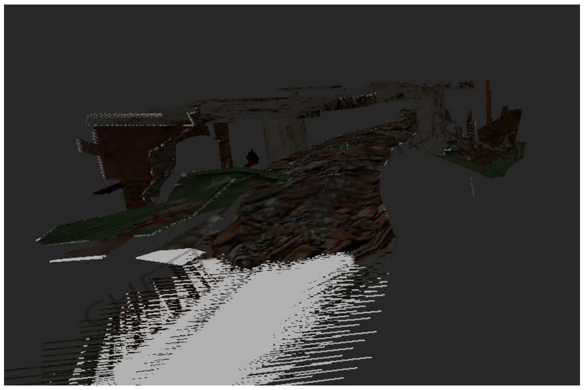
All in All the best we can suggest for a future work is to study and understand better how to tweak the parameters of the method and simulate their effects in each situation.

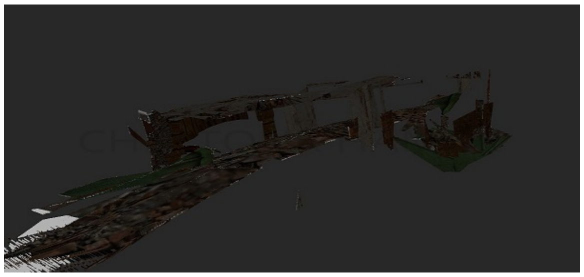
The next work has to include mapping using a real robot in a real environment and asses the real performance under a real environmental factors.

6 Conclusion

This paper discussed the importance of mapping some techniques used for that and more importantly the challenges a robot trying to mapp an environment can face.

Through testing the rtab mapping technique or real-time appearance based mapping in two simulated environments, a comparison showed the limitation and diffculties a future work has to study and overcome.


(a)


(b)

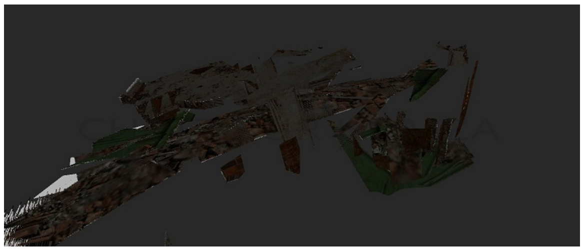

(c)

Fig. 15. Mapping of the collapsed fire station .

Generally speaking, mapping is a very desired ability in modern robots. The only limitation to its applications is the imagination but it can be really useful in mapping newly discovered caves or even forests or to map new reached planets or asteroids, mainly hardly reachable areas and newly discovered places.