Report: Map My World Robot

Apr 21, 2018

Abstract

In this project we are exploring the mapping capabilities of `rtabmap` using the previously build URDF robot model with newly equipped depth camera and necessary configurations. We are building the map of the provided environment `kitchen_dining.world` and explore loop closure on the resulting map. Then we are modeling our own environment in Gazebo and building the map for it with our `rtabmap` map configured robot.

Introduction

In previous project `Where Am I` we've solved the problem of determining the robot location on a given map by using the odometry and sensors data. But what if robot doesn't have an available map and our it needs to operate in an unknown environment? Thats the process of *mapping* come into play.

Mapping is a process of building the map of an environment using robots sensors. In this project we are using Real-Time Appearance Based Mapping (RTAB-Map) algorithm that is using RGB-D Graph-Based SLAM approach that using appearance based loop closure detector for correcting accumulated errors due to the errors in sensor/odometry measurements.

RTAB-Map algorithm is very fast due to the special memory management approach that is using the limited number of locations used for loop closure detection and graph optimization so it works in real-time even for large-scale environment.

Background

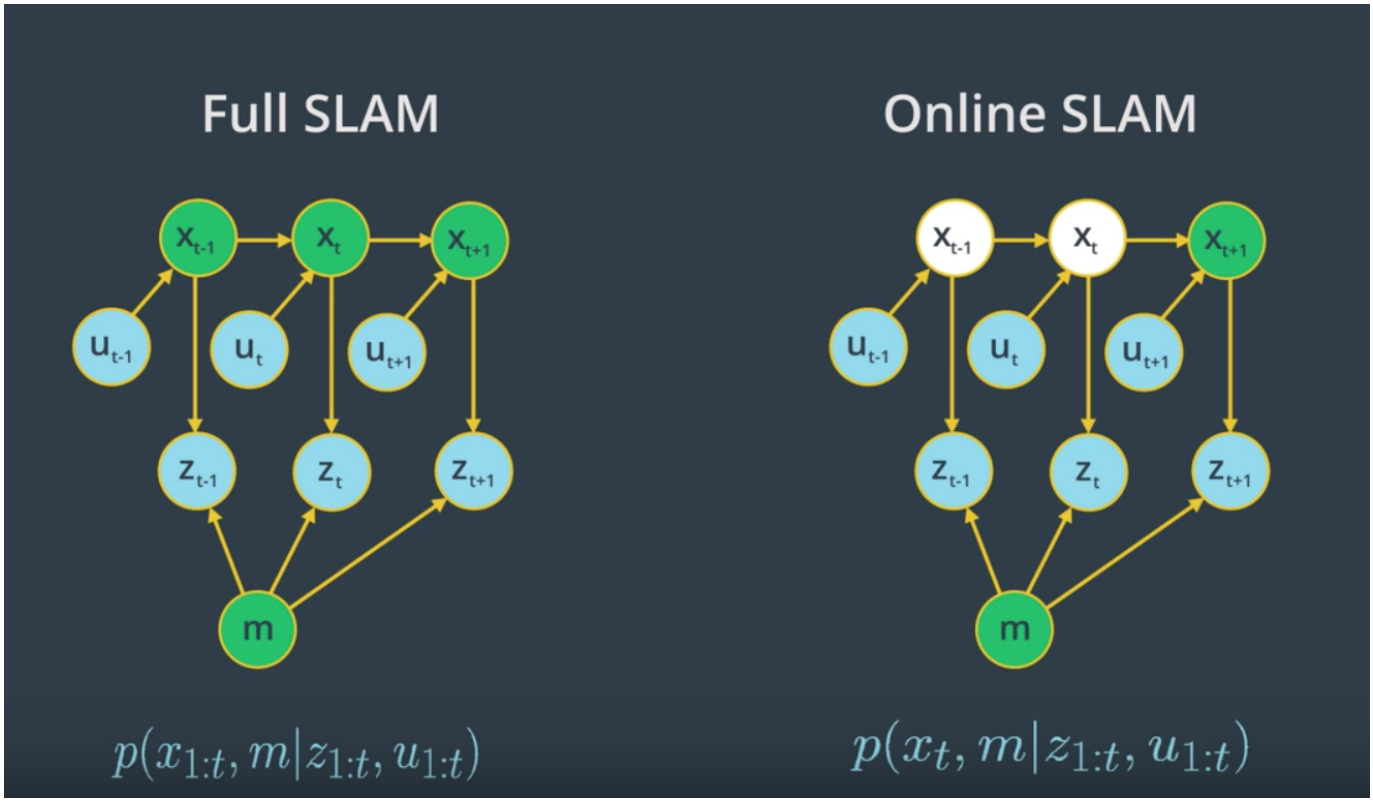
The mapping is more challenging for dynamic vs static environments or continuous vs discrete data. Than we should accommodate for noise in sensors and measurements, detect cycles (previously visited location) and huge hypothesis state that should be calculated on a limited computational resources of a mobile robot and we had very complex problem at hand.

Occupancy Grid Mapping

Occupancy grid mapping algorithm is building the posterior over maps given the known poses and measurements. It is relies on inverse measurement models which reason from the measurements about environment that cause such measurements. This algorithm works mostly for 2D environments because 3D maps quickly becomes very big and hard to compute.

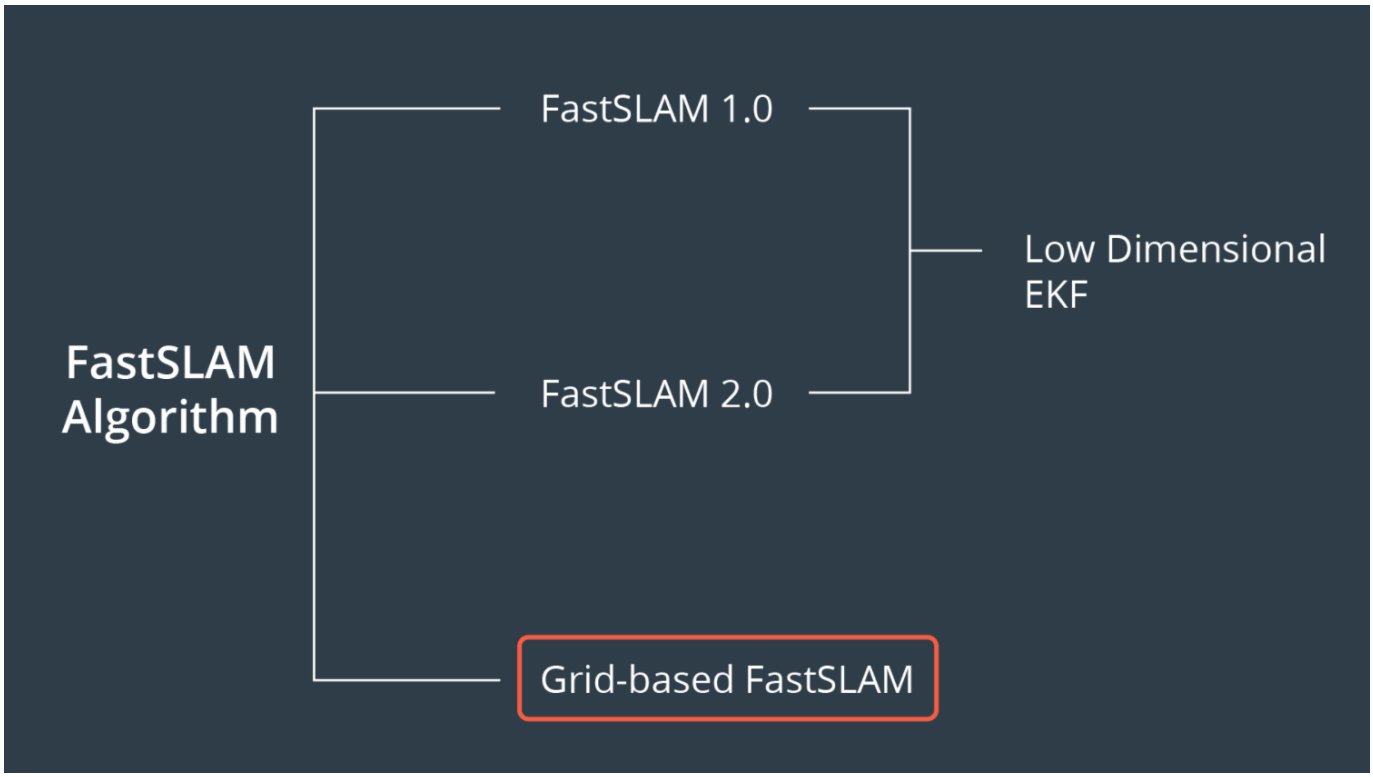
SLAM

Simultaneous localization and mapping (SLAM) problem is building map and localizing robot given the controls and measurements only. It's mapping with unknown poses. There Online SLAM and Full SLAM algorithms, first is determining only current location of the robot at time `t` given measurements and controls and second is finding the whole robot trajectory from `T=0` to `T=t`.

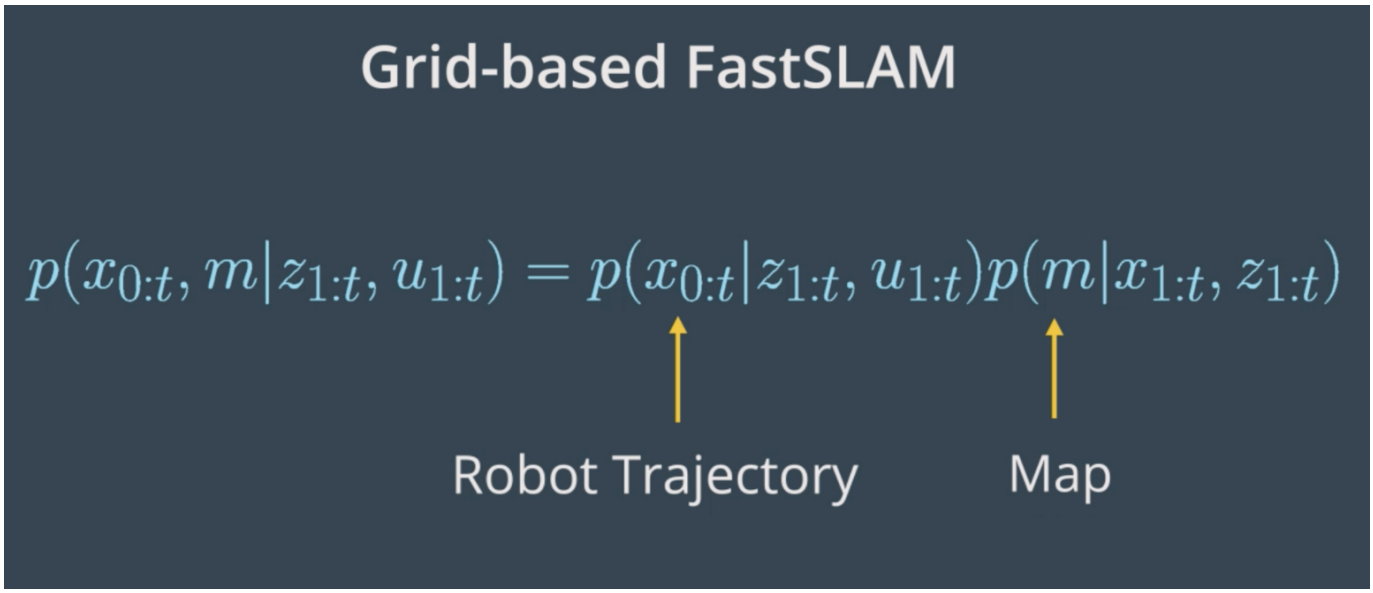


FastSLAM

One of the popular SLAM algorithm is a FastSLAM, that decomposes the SLAM problem into a localization problem and landmark estimation problem that are conditioned on robot pose estimate. FastSLAM uses a modified particles filter for estimating a posterior over robot paths.



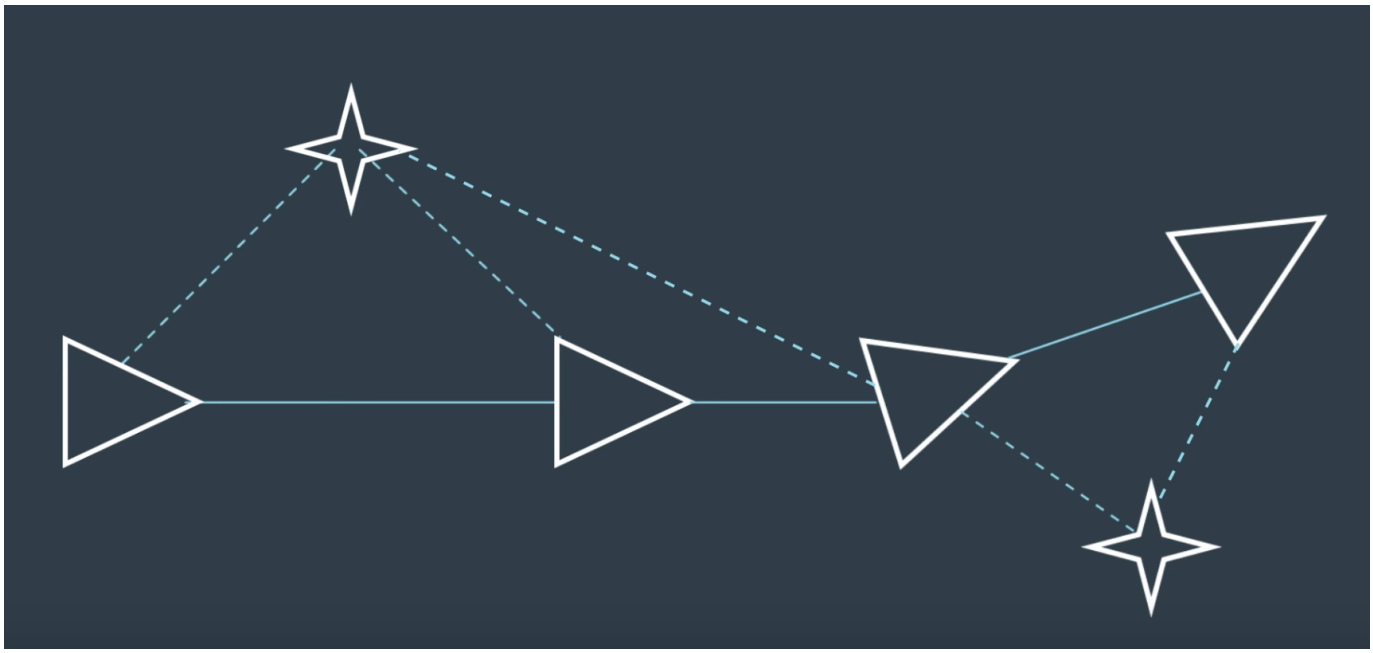
By representing maps as a grid we can get away from the landmarks based approach and estimate a map for any arbitrary environment.



It's a combination of robot pose estimation using particle filter and occupancy based mapping algorithm with known poses.

Graph-SLAM

Graph-based SLAM algorithms calculating graph of poses, features, measurements and motion constraints that is determining the resulting location of robots and features by satisfying all given constraints.



Graph optimization is based on maximum likelihood estimation (MLE) approach. MLE attempts to find the graph parameters that maximizes the likelihood function. For Gaussian distribution model in order to maximize the likelihood we need to minimize its exp() component.

$$J_{GraphSLAM} = \sum_t \frac{(x_t - (x_{t-1} + u_t))^2}{\sigma^2} + \sum_t \frac{(z_t - (x_t + m_t))^2}{\sigma^2}$$

Numerical approach for graph optimization often use gradient-descent algorithm.

Adding Depth Camera to Robot Model

For RTAB-Mapping algorithm we need to provide a depth camera capabilities for our robot, thus we have equipped it with OpenNI Kinnect Gazebo Plugin `libgazebo_ros_openni_kinect.so`.

In order to get the working `laser/scan` topic for the `kitchen_dining` environment we've also added the second `laser1/scan` by converting depth raw image into laser scan topic with the `depthimage_to_laserscan` plugin.

From `si_bot/launch/robot_description.launch`:

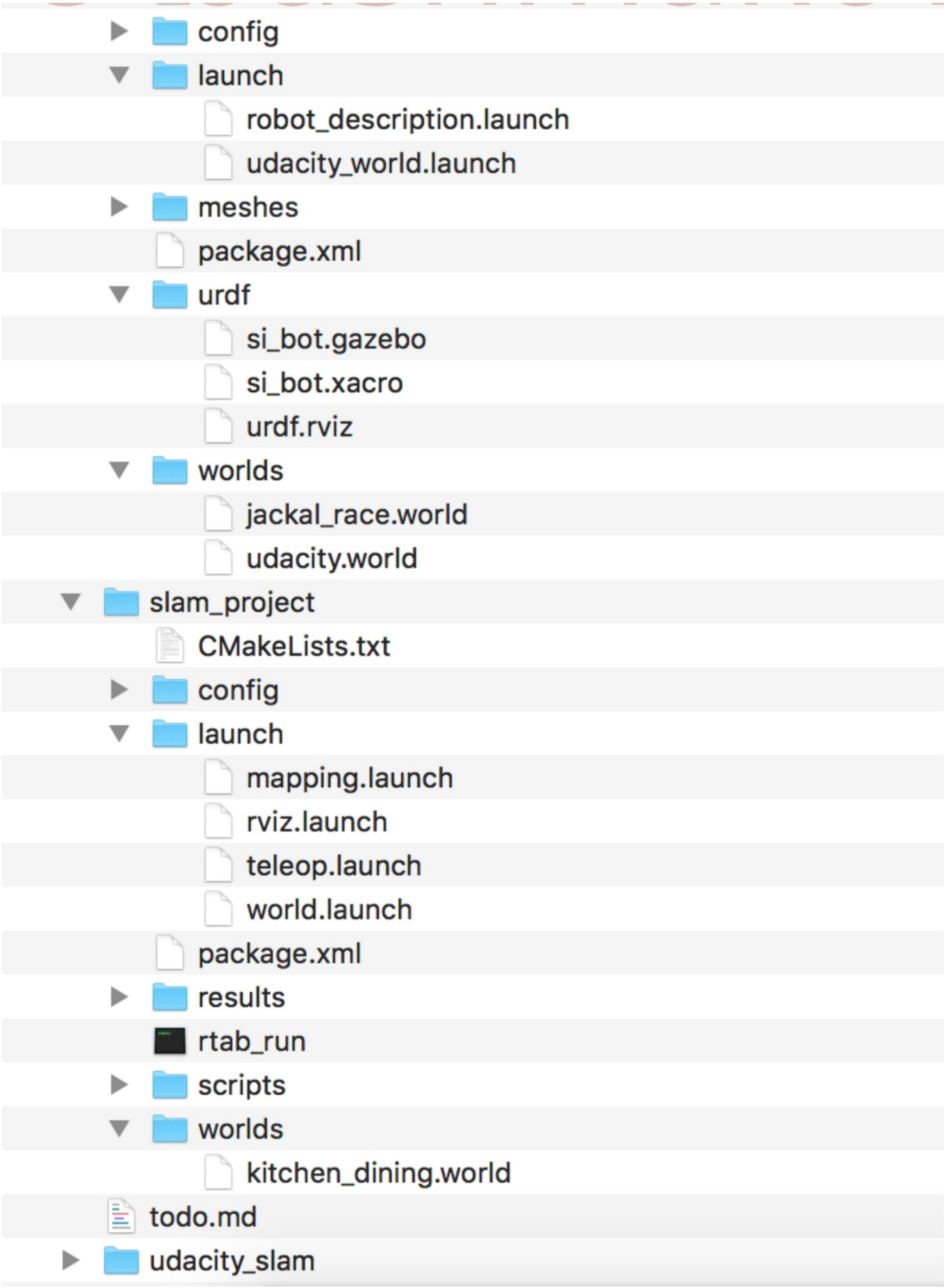
```
<node name="depthimage_to_laserscan" pkg="depthimage_to_laserscan" type="d
        output="screen">
      <param name="range_min" type="double" value="0.10" />
      <param name="range_max" type="double" value="10.0" />
      <param name="output_frame_id" type="string" value="hokuyo" />
```

```
    <!-- Input -->
    <remap from="image" to="/udacity_bot/camera1/depth/image_raw" />

    <!-- Output -->
    <remap from="scan" to="/udacity_bot/laser1/scan" />

</node>
```

The resulting TF frames for the robot are (source `udacity_slam/slam_projects/results/frames.pdf`):

Project folder structure

Project structure for this project was split into two packages `si_bot` with robot description and `slam_project` with everything else. Also top-level `udacity_slam` metapackage used for convenience.


config
launch
robot_description.launch
udacity_world.launch
meshes
package.xml
urdf
si_bot.gazebo
si_bot.xacro
urdf.rviz
worlds
jackal_race.world
udacity.world
slam_project
CMakeLists.txt
config
launch
mapping.launch
rviz.launch
teleop.launch
world.launch
package.xml
results
rtab_run
scripts
worlds
kitchen_dining.world
todo.md
udacity_slam

How to run `my_lab` world

In order to run the mapping on custom `my_lab` world make the following.

1) Change `CATKIN_WS_PATH` in `rtab_run` shell script to point to your Catkin workspace.

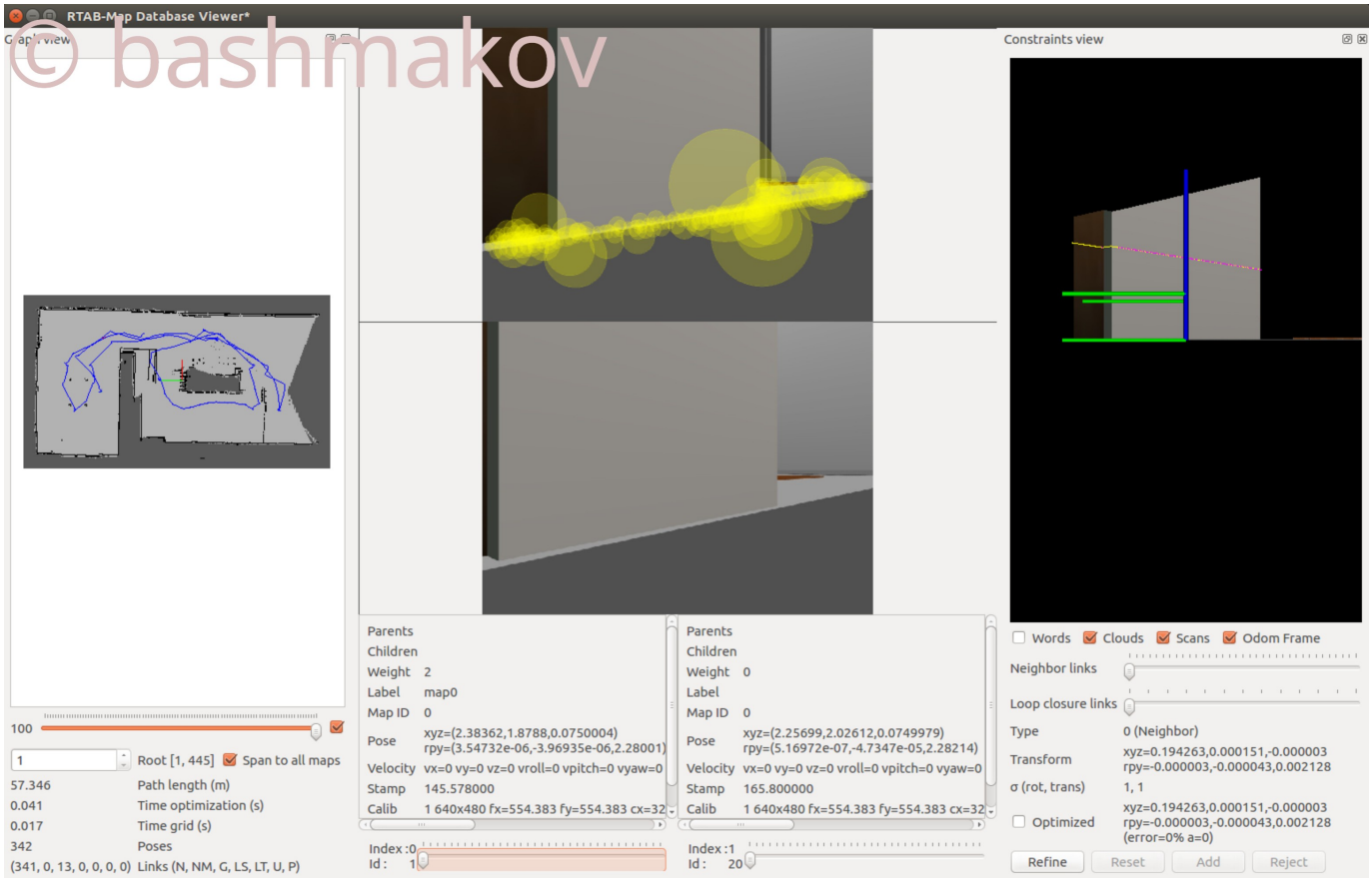
2) Enter “l” letter to trigger the loading of the `my_lab` environment when you’ll see the prompt:

```
Enter target world destination or "d" for default or "l" for my_lab_distin
```

Results

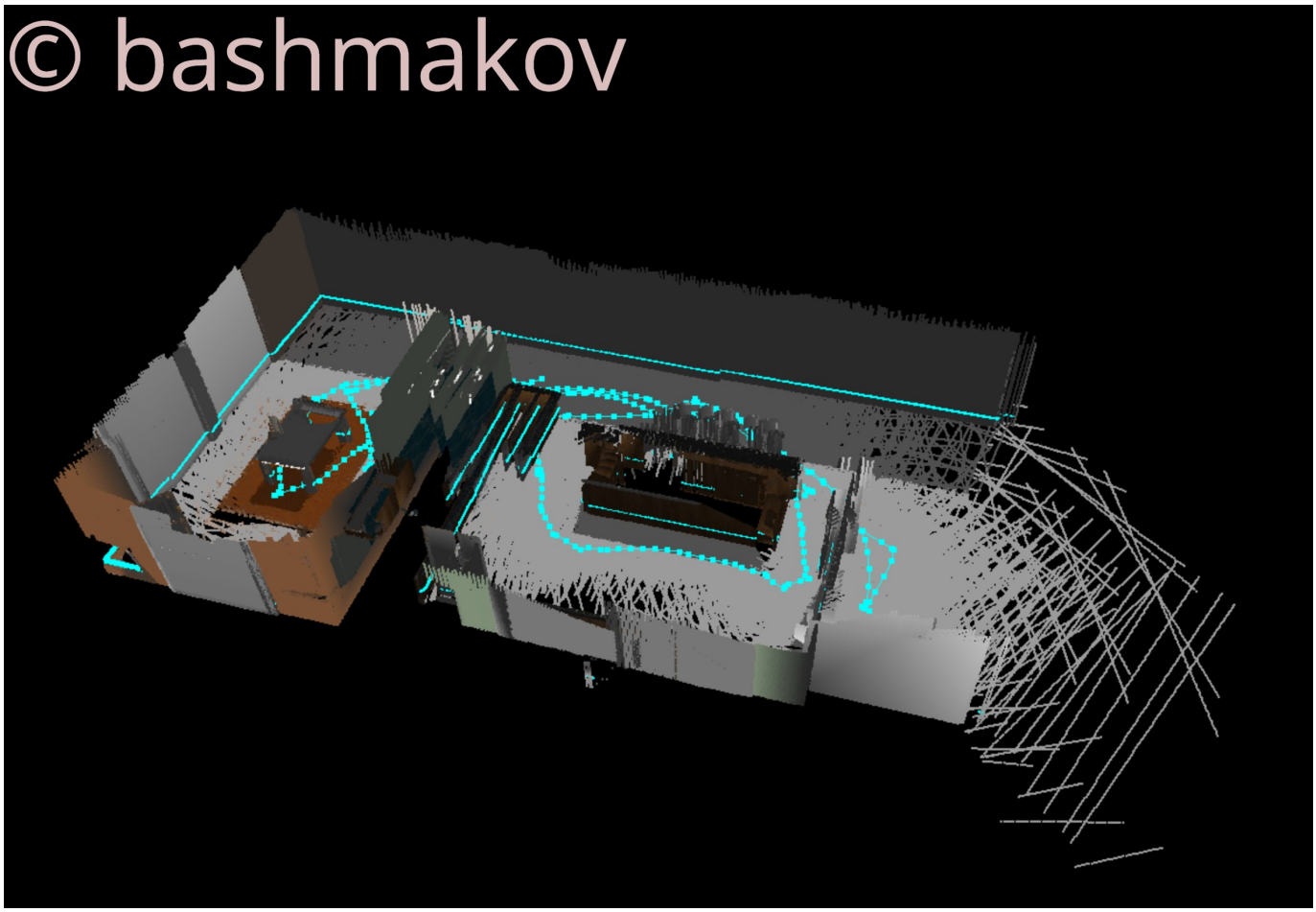
Kitchen Dining

The resulting mapping for provided `kitchen_dining.world` environment is:



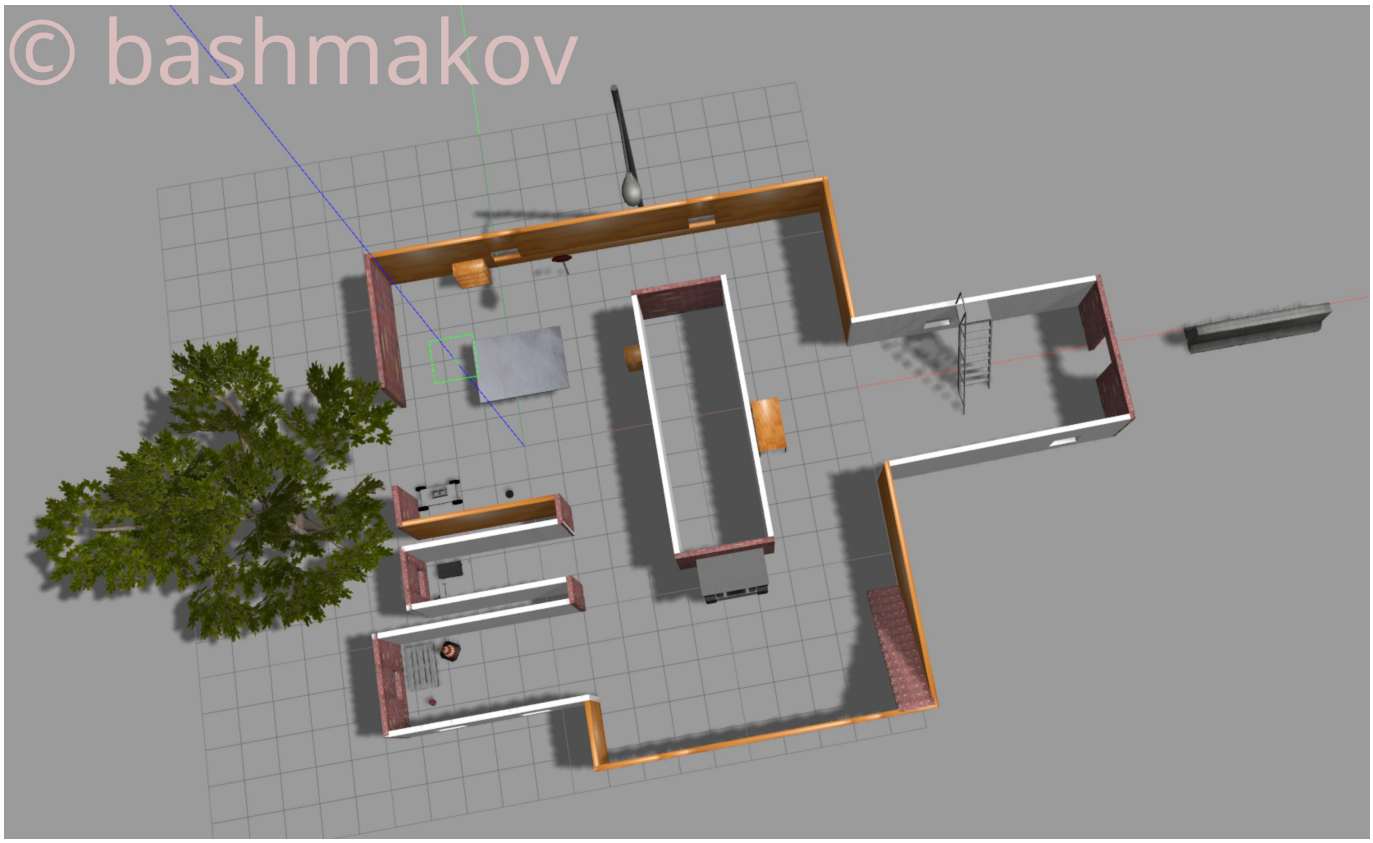
there was 13 global loop closure events during the mapping session.

The same kitchen dining map in 3D representation:



My Lab (new environment)

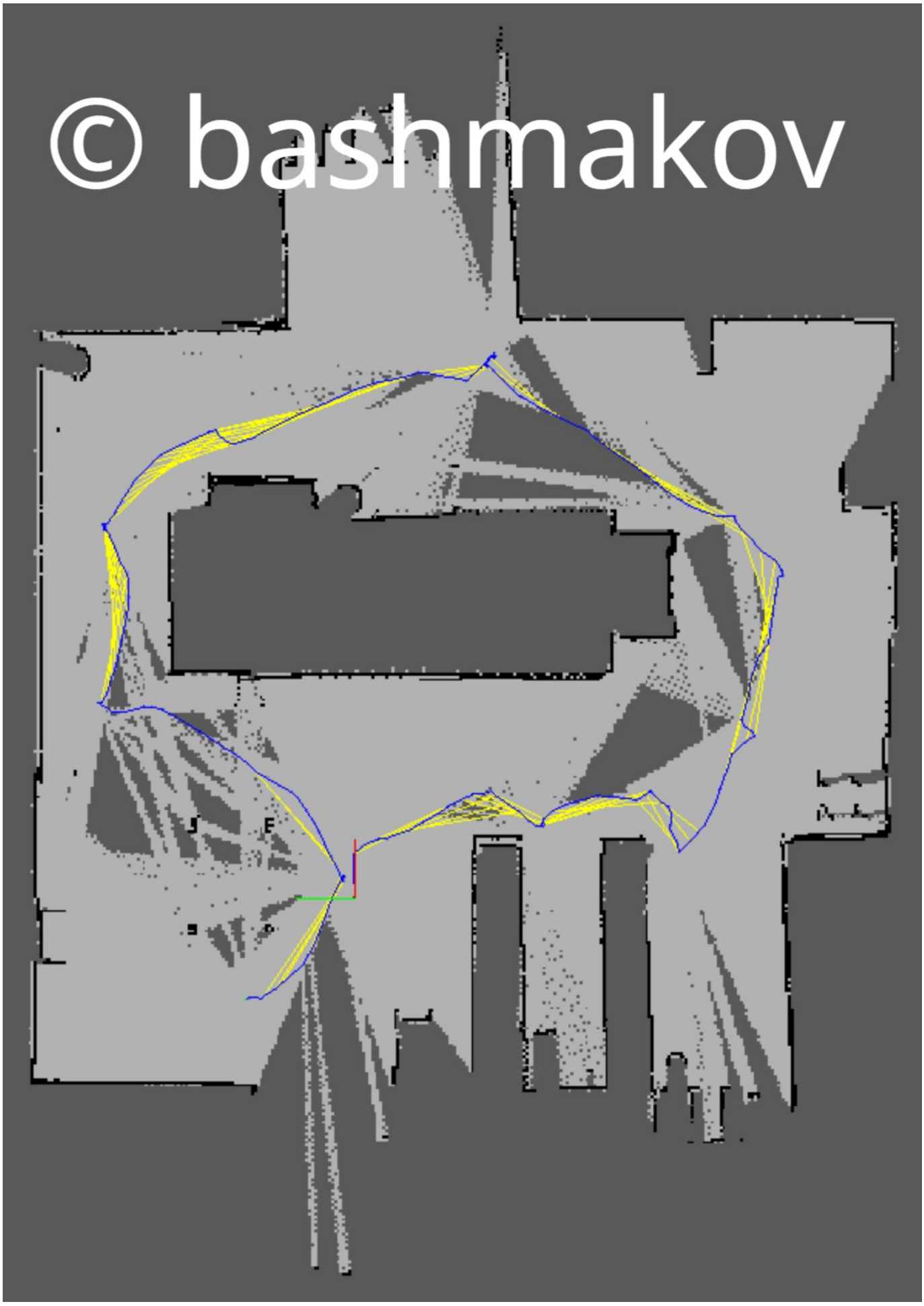
The `my_lab` environment was designed as a second candidate for the mapping algorithm tests.



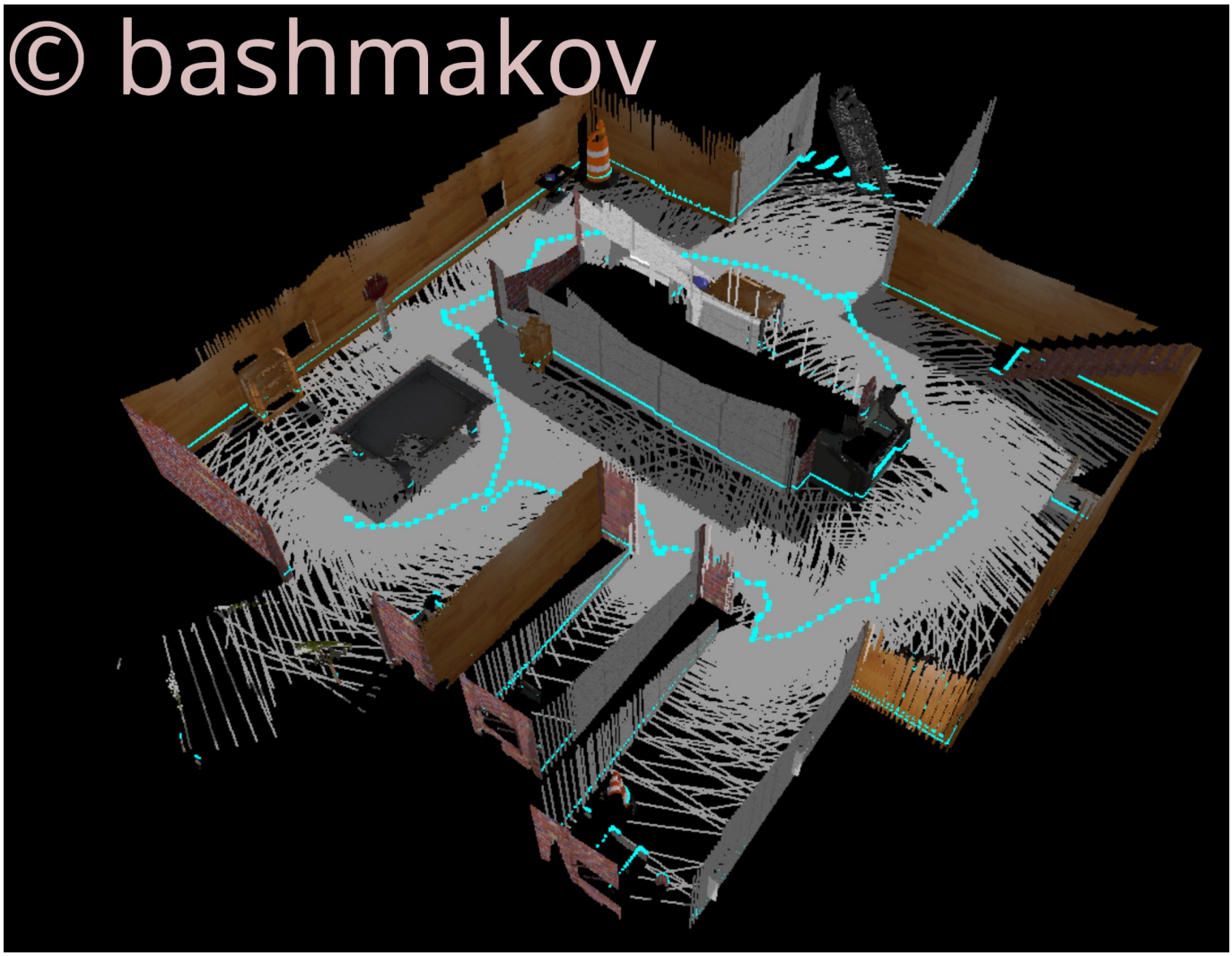
(My Lab version 1 - without some interior features that were added later)

And the map of the environment build with RTAB-Map:


© bashmakov


© bashmakov

And the 3D map from Rtabmap viewer:



Discussion

RTAB-Map is a fast algorithm that worked really well even with `rtabmapviz` launched during the mapping process.

In this project there were two confusing things:

1. Rotation of the depth camera point cloud that was fixed by introducing new invisible link to shift the rotation along the forward looking depth image.

2. Laser scan was not returning any points for `kitchen_dining` environment but worked for other environments. (looks like there wasn't properly defined colliders). This issue was fixed by converting depth image into `laser1/scan` with the help of `depthimage_to_laserscan` ros package.

Another striking particularities of loop closure is when the environment and feature detector algorithm leads to the errornous loop closures the whole maps become completely unusable.

Here is an example of the `my_lab` world with couple of mistaken loop closures (due to environment symmetry caused by geometry and repetitive tile pattern on the walls):

Couple of examples with mistakenly approved loop closures:


© bashmakov

But when loop closures detected correct matches, all obtained maps represent the provided environments in a great detail and can be used with map server for localization algorithms, like AMCL, and save compute power on extensive SLAM computation.

Future Work

Future work can include the real robot operation with RTAB-Map running on Jetson TX2 with Intel RealSense 435 sensor and 4-wheel car base.

This technique can be also used for implementing mapping of the environment with the iPad and Occipital Structure Sensor or with Microsoft HoloLens using access of the depth image sensor data that was introduced in the latest firmware updates in March 2018.

Capsules Bot

Pavlo Bashmakov

bexcite

bashmakov

Exploring autonomous systems, robotics and mapping world around us.