"Promobot" LLC

Building a map using cartographer

Manual

2019

Contents

Preparation

Before mapping you need to make sure:

- That the robot is calibrated;
- The robot has a lidar;
- The robot has IMU installed;
- The floor is suitable for the robot and has no height differences and obstacles:
- Walls surface do not impact the lidar's effectiveness.
-

Mapping

To create a map, proceed as follows:

1. Access the terminal;
2. Check the IMU data

```
rostopic echo /imu
rostopic echo /imu/data
```

The terminal should show real-time IMU data. If the data is not available, check IMU connection and settings.

3. Check the LIDAR data:

```
rostopic echo /scan
```

The terminal should show real-time LIDAR data. If the data is not available, check LIDAR connection and settings.

4. Shut down robot's processes;

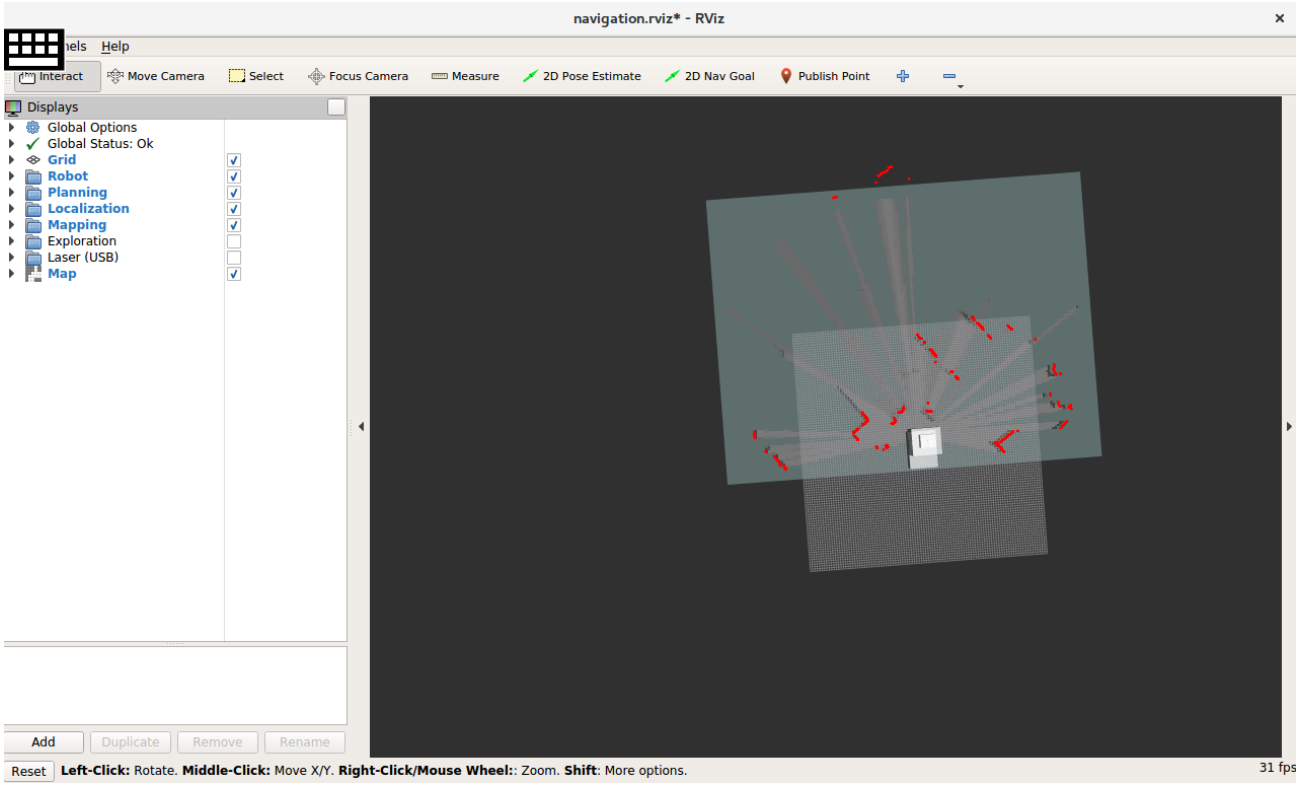
```
killall promobot_gui_op
killall core.sh
killall roscore
```

After completing these commands, the robot GUI will be closed.

5. Run mapping;

```
rosrun promobot_bringup map_create.sh
```

After that you should see the **Rviz** window.



Make sure that obstacles are painted red, robot's icon is shown fully and there are no error messages.

Rviz has built-in configuration that is most suitable for mapping. To use this configuration press **ctrl+o** or **File → Open**, then follow the path:

/opt/promobot/share/promobot_cartographer/configuration_files

and choose *demo_promobot.rviz*

If you can't find the robot's icon in the `Rviz` interface check the `cartographer_ros` folder in /opt/promobot/share

6. To create a map, you should move the robot through the room (use the controller);

   It's better to do it several times. Mapping ending should be in the same place with mapping beginning. **To cancel mapping press ctrl+c** in the terminal.

7. Make sure you save the map.

   Run secondary terminal and save the map using:

   ```
   rosrun promobot_bringup map_save.sh
   ```

   Close the secondary terminal. To finish mapping press **ctrl+c** in the terminal that is used to run Rviz.

   IMPORTANT NOTE. If map is not saved after mapping process, robot won't move neither in auto-mode nor in manual control mode.

8. Set the following parameters in the database web-interface:

Parameter	Value
robot_settings/driving/useMap	true
robot_settings/driving/usePeopleSearch	false
robot_settings/driving/useRadius	0
robot_settings/driving/skipStationScanning	true false (for release 1.9.3 and above)
robot_settings/tf_switcher/default_in_frame	cartographer_frame

Virtual walls, points arrangement and recommendations

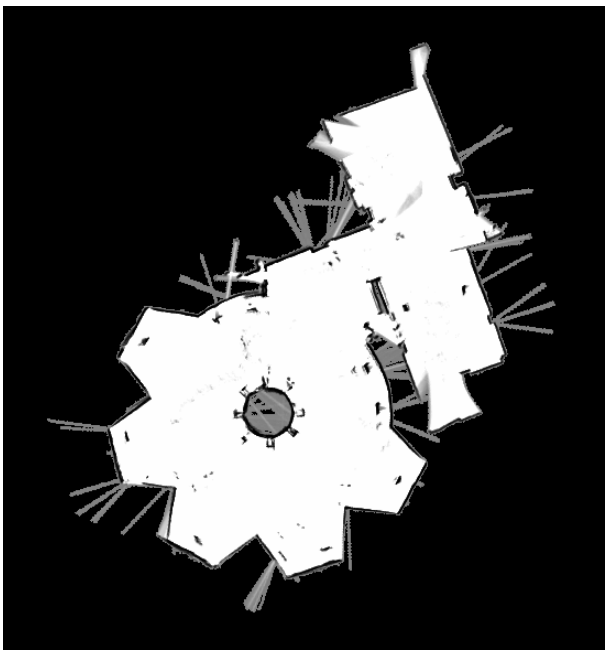
Virtual Walls

Saved map will be located at:

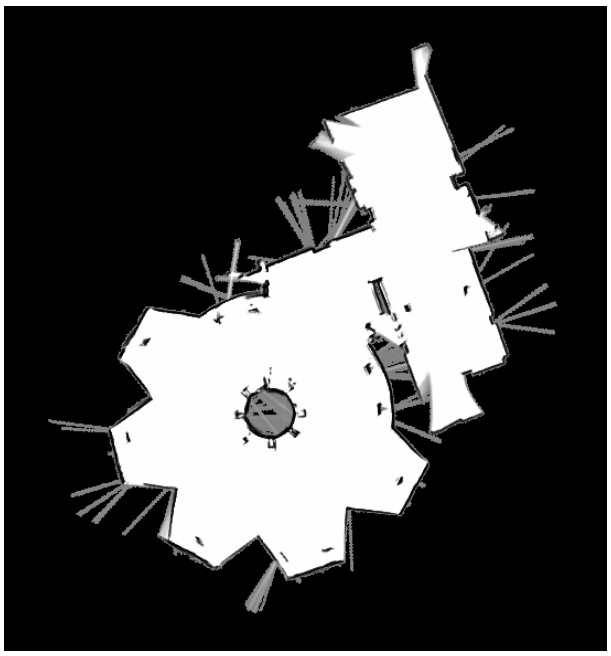
```
/home/promobot/.promobot/resources/maps/map.pgm
```

You can open the map in the graphics editor (GIMP, Inkscape, etc.). Dark color is for walls and obstacles and white is for the space where the robot will be able to move.

The automatically generated map file is likely to be filled with “phantom”, non-existing obstacles that should be cleaned in the graphics editor. The example below shows the initial map and the cleaned map.

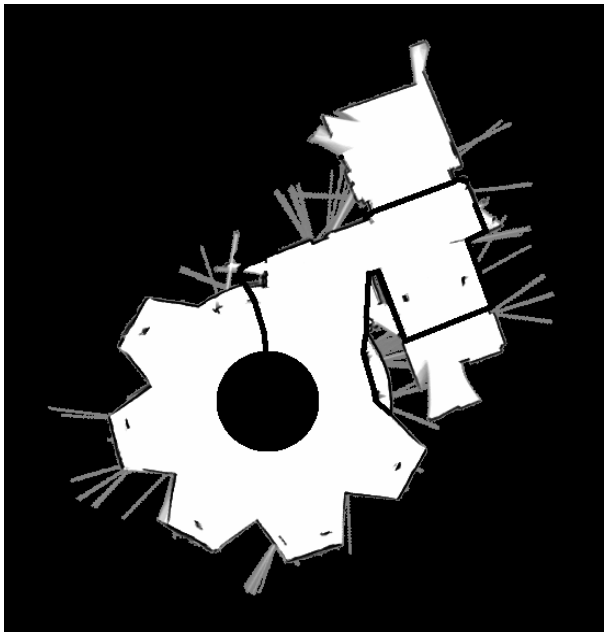


Initial map



Cleaned map

If you want to restrict robot's movement you can add "phantom", virtual walls the same way you delete "phantom" obstacles: by drawing them in the graphics editor. The example below shows the initial map and the map with some virtual walls.



Map with virtual walls

IMPORTANT NOTE. Virtual walls shouldn't have any gaps because they can be treated as a free space. That may cause unpredicted robot's movement and should be avoided. Check the map using Bucket Fill tool.If you want to limit robot's movement you can add "phantom", virtual walls the same way you delete "phantom" obstacles: by drawing them in the graphics editor. The example below shows the initial map and the map with some virtual walls.

Points arrangement

There are 2 ways to position points:

1. Semi-automatic mode;
2. Manual mode.

Advice. Use the 1st method as the main one, and the second method only if you need to shift a particular point.

Semi-automatic mode

Before the point positioning you should open GUI and switch the robot to the Manual mode by pressing **back+start** on the controller.

IMPORTANT NOTE. After the point positioning robot should be in the starting point in the exact position it was in the beginning of the positioning (it's initial turn angle deviation should be $\pm 5^{o}$). Use the charger as a reference point.

To position points in the Semi-automatic mode, proceed as follows:

1. Minimize GUI;
2. Open new terminal and run

   ```
   tail -f ~/.promobot/log/latest/movement.txt
   ```

   Information on the screen will show the points positioning process.

3. Record the current point:
   - Take a controller and press **back+a**;
   - Or type the following in the terminal:

   ```
   rostopic pub /drive/points/save std_msgs/Bool "data: true"
   ```

4. Wait for 5 seconds (an appropriate entry “point N is saved” will be created in movement.txt) and then either:
   - Press **back+a**, if you are using a controller
   - Or type the following in the terminal:

   ```
   rostopic pub /drive/points/save std_msgs/Bool "data: false"
   ```

5. Move the robot to the next point using the controller and repeat steps 2 and 3;
6. After you create all the points, finish the point positioning by returning the robot to the starting point;
7. Make sure there is **point.json** file in **.promobot/resources** directory;
8. To enable automatic movement copy **point.json** and rename it to **map.json**

   You should now have 2 identical files: points.json and map.json

   The first point should be no closer than 2 meters to the charger and the rest – not closer than 1 meter.

Automatic mode

Movement order is set in the **map.json** file.

Map.json requirements:

The file consists of multiple commands and points’ parameters:

- **angle** – w and z coordinates are used for robot’s turning on the point;
- **id** – point number;
- **pos** – x and y coordinates of the point in 2d space
- **text_start** – action set that is used by a robot before it moves to the next point (robot performs actions on “previous” point after it receives command to move to that point). Consists of several mandatory parameters:
  - **action** – action (see action description) that robot will perform whet it will be at the point. You can only set one action of each type (for example, Navigation + Script, but not Script + Script). All actions are performed simultaneously;
  - **anchor** – activates a linguistic base anchor (see anchor description);
  - **animation** – facial expression that robot shows on it’s screen (see facial expressions description);
  - **text** – phrase that the robot will say;
  - **url** – web page that the robot will open and show on it’s screen.

If you are not using a parameter you should set it to *0* (numeric) or *null* (text).

- **text_finish** – action set that is used by a robot when it will arrive to the point. Parameters are the equal to "**text_start**".

File should be valid. You can check it at the third-party services like **https://jsonlint.com/**

IMPORTANT NOTE. File could be valid but have a wrong structure that won't work for the robot's software.

Map.json file should correspond to the following structure:

```
\\ Start of the file, "{}" are used for points set
{
  "points":
    \\ setting a points set with "["
    [
      \\ first points set element, setting it's description with "{"
      {
        "angle":
        \\ first parameter, setting it's description with "{"
        {
          "w": 0.376174405040294,
          "z": 0.92654887458384
        \\ after you fill the parameter's description close it with "}"
        },
        "id": 0,
        "pos":
        \\ next parameter that consists of multiple values, setting it with "{"
        {
          "x": -0.906758110520606,
          "y": -5.17015431614186
        \\ after you fill the parameter's description close it with "}"
        },
        "text_start": null,
        "text_finish":
        \\ this parameter is a data set, so you start it with "["and, because there is only one list of values, it is set in
"{"
          [{
            "action": null,
            "anchor": 0,
            "animation": 0,
            "text": null,
            "url": null
          }]
      \\ after you fill the first point's description close it with "}"
      },
      \\ follow the same steps for the next point
      {
        "angle":
        {
        "w": 0.429732129790555,
        "z": 0.902956420114323
        },
        "id": 1,
        "pos": {
        "x": -2.12233353289366,
        "y": -3.59353399042218
        },
```

```
            "text_start": null,
            "text_finish":
            \\ this set has several list, each of them is in “{}” and they all are divided by “,”
            [
               {
               "action": "navigation:2",
               "anchor": 0,
               "animation": 0,
               "text": null,
               "url": null
               },
               {
               \\ another data set that consists of several actions
               "action": "[lingvoCase:default],[script:get_hand_boy]",
               "anchor": 0,
               "animation": 0,
               "text": null,
               "url": null
               },
               {
               "action": "navigation:6",
               "anchor": 0,
               "animation": 0,
               "text": null,
               "url": null
               }
            \\ after you fill the data set’s description close it with “]”
            ]
         \\ closing the data set
         }
      \\ closing the points set
      ]
\\ closing the file
}
```

Problems that may occur during the automatic mode setup and their solutions:

Problem	**Solution**
Robot is not moving to any point	Check if map.json is valid
Robot is moving to the wrong place after it is sent to a certain point	Check the map.json’s structure
Robot doesn’t perform a script when it arrives to the point	Check the map.json’s structure, try to change action’s order if there are several different actions

Manual mode

Run it while GUI in running.

- Open Rviz. roslaunch promobot_bringup rviz.launch
- Open the topic in the terminal: */drive/destination rostopic echo /drive/destination*
- Go back to Rviz and send robot to the point by the 2D Nav Goal button.

- Write topic data to the file: *~/.promobot/resources/points.json*

Example of topic data **/drive/destination**

```
header:
  seq: 0
  stamp:
    secs: 1523636206
    nsecs:  69873363
  frame_id: "map"
pose:
  position:
    x: -1.20477819443
    y: 0.55451887846
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.998724793097
    w: 0.0504855192539
---
```

Example of *.promobot/resources/points.json* file:

```
{
  "points": [{
        "angle": {
          "w": 0.376174405040294,
          "z": 0.92654887458384
        },
        "id": 0,
        "pos": {
          "x": -0.906758110520606,
          "y": -5.17015431614186
        },
        "text_start": null,
        "text_finish": [{
          "action": null,
          "anchor": 0,
          "animation": 0,
          "text": null,
          "url": null
        }]
      }, {
        "angle": {
          "w": 0.429732129790555,
          "z": 0.902956420114323
        },
        "id": 1,
        "pos": {
          "x": -2.12233353289366,
          "y": -3.59353399042218
        },
        "text_start": null,
        "text_finish": [{
          "action": "navigation:2",
```

```
            "anchor": 0,
            "animation": 0,
            "text": null,
            "url": null
         }, {
            "action": "navigation:4",
            "anchor": 0,
            "animation": 0,
            "text": null,
            "url": null
         }, {
            "action": "navigation:6",
            "anchor": 0,
            "animation": 0,
            "text": null,
            "url": null
         }]
}]}
```


Initial position

The initial position is set by the parameters in the database:

```
/navigation/initial_pose/position/x
/navigation/initial_pose/position/y
/navigation/initial_pose/orientation/w
/navigation/initial_pose/orientation/z
```

To find the coordinates of a point, you can use the first 3 items in the section "**Manual mode**" in the section Point positioning.

Charger should be in the starting point. If you want robot to move to the charger set **/driving/skipStationScanning** to **true**.

Recommendations

While mapping, avoid moving slowly and staying in one spot, it is recommended to move the robot fast along the perimeter of the room, the cartographer will combine all the sub-maps into one.

It is better to start building a map from the zero point (that is, from the place of charging) and finish at the same point.