Robot Programming Report

Giovanbattista Gravina, Luca Nunziante

March 2022

Project Overview

The project consists in a normal based matcher integrated in the ROS environment. The aim is to:

- Extract the normals from a 2D Laser Scan
- Create a 4D kd tree and use it to run icp with normals
- Output the transformation computed in the tf topic
- (Additional) Build a map of the robot's sorroundings

Project Structure

Hereafter, the nodes that compose our system are presented.

Stage ros

It is the mobile robot simulator provided by ros at the basis of this project. Among other things, it publishes the result of a laser scan on the topic `/base_scan`.

Cloud broadcaster

It reads the laser scan from the above mentioned topic and generates a cloud of 2D points on another one called `/cloud` attaching in the header the timestamp of the scan and the id of their origin frame - `base_laser_link`.

Normal extractor

Firstly, it subscribes to the `/cloud` topic and transforms the cloud of points from the `base_laser_link` frame to the `odom` frame. Then, using a kd tree, it extracts the normals.
Finally, it creates a topic called `/icp_topic` whose message structure is custom made and is as follows:

```
a timestamp
a first array of 4D vectors
a second array of 4D vectors
```

The 2 arrays contain the last 2 clouds of points - together with their normals - received from the cloud broadcaster, and the timestamp is the one of the most recent set received.
The ICP algorithm will run on these sets of points.

Icp runner

It subscribes to `/icp_topic` and uses the aforementioned arrays to run icp, obtaining the relative transformation. Subsequently, it extracts from the isometry matrix the translation vector and the quaternion in order to publish the transformation on the `/tf` topic with the received timestamp.
At last, it publishes on another topic `/icp_data` a `PointCloud` message containing the set of fixed points involved in the icp process.
It is worth noticing that the timestamp inserted in the header part of this message is related to the most recent cloud received - the one of moving points - so as to coincide with the `/tf` timestamp.

Map builder

This node is subscribed to `/icp_data` topic from which it receives a cloud of points. First and foremost, it looks if a new relative transformation is available on the `/tf` topic. If that is not the case, the node does nothing. However, since the icp runner publishes on the `/icp_data` topic only after having published on `/tf`, this should not happen.
Furthermore, the node stores an isometry matrix containing the whole history of the published transformations and premultiplies the received points by it. These points are written in a file, and they will make up the map.
In the end, it updates the history with the newly available transformation.

Compile procedure

Clone the repository on github with the following instruction:

```
git clone https://github.com/gianni0907/rp_project.git <directory>
```

where `<directory>` is a local folder that will be created and in which the files will be put.
Go in the `<directory>/src/` folder and initialize the ros workspace with the following command:

```
catkin_init_workspace
```

A `CMakeList.txt` file will be created in the current folder.
Go back to `<directory>` and execute the build command to compile:

```
catkin build -j
```

This will take approximately 20 seconds.
In the top level directory the `build` and `devel` folders will be created, and to make the system recognize our package we need to issue the following command to source the workspace:

```
source devel/setup.bash
```

Run procedure

The basic command to run a ros node is

```
rosrun <package_name> <node_name>
```

However, in the git repository there is a launch directory to ease the run procedure since 5 nodes need to be run.
In particular, in the `<directory>/src/nicp_package/launch` folder there are 2 launch files:

- `stage.launch` that launches the stage ros simulator
- `start.launch` that runs the 4 nodes that make up this project

These files can be used with this command

```
roslaunch nicp_package <launch_file_name>
```

Testing

To test that the icp process is working, one can issue a velocity command to the robot in the simulator and check that the map is built correctly. It is possible to give a velocity command by publishing a message on the `/cmd_vel` topic on a shell, e.g.:

```
rostopic pub /cmd_vel geometry_msgs/Twist "linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.3" -r 10
```

The `map_builder` node saves a file `map.txt` in the home directory, and it is possible to visualize the map with gnuplot by accessing the gnuplot shell and issuing a command like the following one:

```
plot "map.txt" u 1:2 w p pt 7 ps 0.1
```

It is also possible to examine the node communication infrastructure with the `rqt_graph` command.