Université Paul Sabatier

Master Intelligence Artificielle et
Reconnaissance des Formes
Master Robotique : Décision et Commande

User Manual - Navigation Between Markers

Mobile Robot Navigation

Authors:
Thibaut Aghnatios
Marine Bouchet
Bruno Dato
Tristan Klempka
Thibault Lagoute

Tutors:
Frédéric Lerasle
Michaël Lauer
Michel Taix

30 March 2017

Document tracking

Name	Major Version	Minor Version	Creation Date	Last version
User Manual - Navigation Between Markers	A	5	30/03/2017	6/04/2017

Document authors

Redaction	Integration	Review	Validation
Bruno Dato Thibaut Aghnatios Marine Bouchet	Bruno Dato Marine Bouchet	Bruno Dato	??

Document validation

Validation	Name	Date	Visa

Broadcast list

User Manual - Navigation Between Markers is distributed to all clients and external stakeholders.

Review history

Version	Additions or modifications	Author	Date
A.0	Document creation	Bruno Dato	30/01/2017
A.1	Sections 1 and 2	Bruno Dato	1/04/2017
A.2	Section 1.4.5	Thibaut Aghnatios	02/04/2017
A.3	Section 1.4.3	Thibaut Aghnatios	04/04/2017
A.4	Sections 1.4 and 2.3	Marine Bouchet	04/04/2017
A.5	Review	Bruno Dato	06/04/2017

Contents

1 Prerequisites **3**
1.1 Equipment . . . 3
1.2 Software . . . 3
1.3 Workspace . . . 3
1.3.1 Build workspace . . . 3
1.3.2 Download package . . . 4
1.3.3 Build executables . . . 4
1.4 Map and markers configuration . . . 4
1.4.1 Environment map . . . 4
1.4.2 Markers disposition . . . 5
1.4.3 Graph of the markers . . . 5
1.4.4 Markers static transforms publisher . . . 6
1.4.5 Visibility map . . . 6

2 Navigation Between Markers **7**
2.1 On the TurtleBot PC . . . 7
2.1.1 Basic features . . . 7
2.1.2 Navigation . . . 7
2.1.3 Supervisor . . . 7
2.2 From a remote PC . . . 7
2.2.1 Basic features . . . 8
2.2.2 Navigation . . . 8
2.2.3 Supervisor . . . 8
2.3 Behaviour of the navigation . . . 8

1 Prerequisites

1.1 Equipment

- TurtleBot 2
- AR markers

1.2 Software

To be able to use any TurtleBot 2 with all the basic features, you need to complete the following tutorials :

- Turtlebot Installation

  `http://wiki.ros.org/turtlebot/Tutorials/indigo/Turtlebot%20Installation`

- PC Installation

  `http://wiki.ros.org/turtlebot/Tutorials/indigo/PC%20Installation`

- Network Configuration

  `http://wiki.ros.org/turtlebot/Tutorials/indigo/Network%20Configuration`

You also need the following software and additional package :

- GIT [Installation]

  `https://git-scm.com/download/linux`

- Package *ar_track_alvar*

```
> sudo apt-get install ros-indigo-ar-track-alvar
```

1.3 Workspace

1.3.1 Build workspace

You need a ROS workspace (catkin workspace) to build our project before executing it. If you are running the navigation between markers on the TurtleBot PC you have to create a workspace on the TurtleBot PC. In the case your are running it on a remote PC, you have to create the workspace on this PC.

Place you where you want to build the workspace and execute the following commands :

```
> mkdir -p /catkin_ws/src
> cd /catkin_ws/src
> catkin_init_workspace
> cd ..
> catkin_make
```

Then, in $\sim$ /*.bashrc*, add the following lines (it is normal if some of them are already there) :

```
#Initialisation Turtlebot kinect
```

```
export TURTLEBOT_3D_SENSOR=kinect

#ROS Version

source /opt/ros/indigo/setup.bash

source <WORKSPACE_PATH>/catkin_ws/devel/setup.bash

#Select corresponding TurtleBot on your network

export ROS_MASTER_URI=http://<IP_OF_TURTLEBOT>:11311
```

1.3.2 Download package

Now, you need to clone the package containing the source code. Place you in your workspace (catkin_ws), and execute the following commands :

```
> cd src

> git clone https://github.com/Projet-Navigation-UPS/TurtleBot-pkgs
```

1.3.3 Build executables

Now that you have downloaded the source code, you just need to compile to build the executable files. Put you in your workspace (catkin_ws) and run the command :

```
> catkin_make
```

Several red lines must appear in the compilation description, it means that the executables we need have been created. If you encounter error at the first compilation, try it a second time. If errors persist, delete the workspace and start over from section 1.3.

1.4 Map and markers configuration

1.4.1 Environment map

You need yo create the map of the environment within you navigate if it is not already available in the folder **/catkin_ws/src/TurtleBot-pkgs/turtlebot_proj_nav/map**. To create the map, we use the **turtlebot_navigation** package which provides a SLAM mode :

- SLAM tutorial link

  ```
  http://wiki.ros.org/turtlebot_navigation/Tutorials/indigo/Build%20a%20map%20

  with%20SLAM
  ```

We recommend to put the robot in a parallel orientation with a wall to facilitate the future definitions of markers orientations. After turning on the TurtleBot and its laptop, execute in different terminals the following commands the TurtleBot laptop :

```
> roslaunch turtlebot_bringup minimal.launch

> roslaunch turtlebot_navigation gmapping_demo.launch
```

Then, on a remote computer where the *.bashrc* is configured, execute the visualization of the SLAM :

```
> roslaunch turtlebot_rviz_launchers view_navigation.launch
```

To make the robot move thanks to your keyboard and explore the environment, execute :

```
> roslaunch turtlebot_teleop keyboard_teleop.launch --screen
```

You can then control the robot with the keyboard and explore you environement. Once the map is satisfying for the future navigation, on another terminal you have to save it :

```
> rosrun map_server map_saver -f <WORKSPACE_PATH>/catkin_ws/src/TurtleBot-pk
gs/turtlebot_proj_nav/map/my_map
```

1.4.2 Markers disposition

Within our project we have used 16×16 markers placed 3 m away minimum from each other. We put their centers 31cm above the ground so the kinect-marker is as parallel as possible to the ground.

1.4.3 Graph of the markers

To choose towards which marker (AR marker) to move, we use a graph representing all the markers defined in an XML format file, *graph.xml* :

```
<WORKSPACE_PATH>/catkin_ws/src/TurtleBot-pkgs/turtlebot_proj_nav/rsc/graph.xml
```

Each node of the graph has different properties :

- *Id*: number corresponding to the AR marker;
- *Label*: name of the node, each node corresponds to an AR marker;
- *PositionX*: coordinate along the x-axis of the known environment map;
- *PositionY*: coordinate along the y-axis of the known environment map;
- *Orientation*: angle between the normal of the AR marker and the x-axis of the map (between 0 and 2π).

The *Id* allow a marker to be located in the graph. The coordinates $(PositionX, PositionY)$ of the markers make it possible to know the positions in the map. *Orientation* allows to command the robot to move in front and at a certain distance from the marker to avoid walls.

For each marker that you will use in your navigation, you need to define a node with its properties and all its links with the others. The costs for the links have to be integers, you can use the distances between the marker in cm for example.

For each marker, between the tags $< Nodes >$ and $< /Nodes >$, add a line of this type :

```
<Node Id=" " Label=" "  PositionX=" " PositionY=" " Orientation=" "/>
```

For each link, between the tags $< Links >$ and $< /Links >$, add a line of this type :

```
<Link Source=" " Target=" " Cost=" "  />
```

You can use the visualizer RVIZ to get the positions and orientations of the markers in the map by displaying geometry poses in the map. A geometry pose will give you a position (x, y, z) and an orientation in quaternions (x, y, z, w) that will have to convert to obtain the orientation you need.

1.4.4 Markers static transforms publisher

In this section, you will need to modify *navigation.launch* :

```
<PATH>/catkin_ws/src/TurtleBot-pkgs/turtlebot_proj_nav/launch/navigation.launch
```

AR markers transforms need to be published. These transforms represent where the markers are in our scene in the system. Use one $static\_transform\_publisher$ for each marker. In the $args$ field put the position (x,y,z), the orientation (qx,qy,qz,qw), the transform parent (always $/map$) and the name of the marker transform ($/marker\_X$) of the marker. Thus, for each marker, you have to add a line of this type to *navigation.launch* :

```
<node pkg="tf" type="static_transform_publisher" name="marker_tf_publisher_<ID>"

args="x y z qx qy qz qw map marker_1 100" />
```

Be careful, for each marker, the related mark must have its z-axis on the normal of the marker and its x-axis pointing upwards.

More info can be found at :

- TF documentation

  ```
  http://wiki.ros.org/tf\#static\_transform\_publisher
  ```

1.4.5 Visibility map

To generate the visibility map, it is necessary to previously create a map of the environment like we did.

For a given map size and for the configurations performed correctly in *visib_init.cpp*, make sure that *graph.xml* is up date with all the markers you are using and then to generate the new visibility map, run *visib_pgmwriter_node* :

```
> rosrun turtlebot_proj_nav visib_pgmwriter_node.cpp
```

For a different map scale, see the developer manual.

2 Navigation Between Markers

First, turn on the TurtleBot (there is a switch button on the side of the robot base). Then, turn on the TurtleBot PC. We will now launch all the ROS nodes that we need to run our application.

2.1 On the TurtleBot PC

2.1.1 Basic features

If you are using the TurtleBot PC, open two terminals and chronologically execute the following commands to activate the minimal features and the vision features, one on each terminal :

```
> roslaunch turtlebot_bringup minimal.launch
```

```
> roslaunch turtlebot_bringup 3dsensor.launch
```

2.1.2 Navigation

To launch all the navigation nodes, execute the following command :

```
> roslaunch turtlebot_proj_nav navigation.launch
```

You can now see on the visualizer RVIZ the map, the TurtleBot and all the markers.

2.1.3 Supervisor

To run the supervisor of the navigation, execute the following command :

```
> rosrun turtlebot_proj_nav hightLevelCommand_node X_GOAL Y_GOAL XY_THRESHOLD
 XY_PRECISION DISTANCE_TO_MARKERS
```

with the following parameter :

- **X_GOAL** et **Y_GOAL** : coordinates (x, y) of the final goal on the map we want the robot to reach ;
- **XY_THRESHOLD** : threshold from which the markers are not used for navigation ;
- **XY_PRECISION** : precision constraint required for the final sound signal to be played or not ;
- **DISTANCE_TO_MARKERS** : The distance to the markers that the robot must respect when moving from marker to marker.

The supervisor will then display on the terminal the action running and the corresponding state of the finite state machine who is in charge of supervising the navigation.

2.2 From a remote PC

You can run the application from a remote PC which means that it will be still executed on the TurtleBot PC but the displays will be on the remote PC.

2.2.1 Basic features

To execute the navigation on the TurtleBot PC from a remote PC, fist you have to ssh to the TurtleBot PC to launch the minimal and vision features. Open a first terminal and write the following commands :

```
> ssh turtlebot@<TURTLEBOTP_IP>
> roslaunch turtlebot_bringup minimal.launch
```

Then, in a second terminal :

```
> ssh turtlebot@<TURTLEBOTP_IP>
> roslaunch turtlebot_bringup 3dsensor.launch
```

2.2.2 Navigation

To run the navigation nodes from a remote PC execute the following commands :

```
> ssh -X turtlebot@<TURTLEBOTP_IP>
> roslaunch turtlebot_bringup navigation.launch
```

You can now see on the visualizer RVIZ the map, the TurtleBot and all the markers. You can also run the navigation on the remote PC by not doing the ssh but you can encounter network problems.

2.2.3 Supervisor

To run the supervisor from a remote PC, execute the following command :

```
> ssh -X turtlebot@<TURTLEBOTP_IP>
> rosrun turtlebot_proj_nav hightLevelCommand_node X_GOAL Y_GOAL XY_THRESHOLD
 XY_PRECISION DISTANCE_TO_MARKERS
```

The supervisor will then display on the terminal the action running and the corresponding state of the finite state machine who is in charge of supervising the navigation.

You can also run the supervisor on the remote PC if you don't ssh on TurtleBot PC.

2.3 Behaviour of the navigation

The navigation takes place between an initial position, the real position of the robot, or in the visibility zone of a marker, and a final position:

Scenario	A	B
Start	Absolute initial position	Unknown initial position but in a visibility zone of a marker
Arrival	Final Position	Final Position

These scenarii are (B bis an extension of A):

B If no input position
    The robot performs a marker search in its visual field
    As long as no marker in view
        The robot turns on itself
        It performs another marker search in its visual field

A As long as the goal is further than the next marker in the goal direction
    The robot moves to the most visibility zone of the marker
    ... close in the direction of the goal
    The robot performs a marker search in its visual field
    As long as there is no marker in view
        The robot turns on itself
        It performs another marker search in its visual field
    The robot moves to the goal

Since the knowledge on the visibility of a marker is not yet operational, it will be considered that the robot is always in a visibility zone. The robot as to be placed in a visibility area at each start of navigation, taking care to give a correct initial position.

Thus, depending on the parameters chosen launching the supervisor, the robot will :

1 Move directly to the final goal X_GOAL and Y_GOAL if the distance between the current position and the final goal is less than $XY\_THRESHOLD$, a special signal sound is played is the distance between the final goal and the current final position is less than $XY\_PRECISION$.

2 If not, while it doesn't find a marker, it will search for one until it detects one (sound signal). Then depending on which marker allows the robot to get closer to the final goal, it will move in front of this marker at a distance defined by $DISTANCE\_TO\_MARKERS$. This behavior will be repeated until the closest marker to the final goal is detected. When it is, the behavior 1 occurs.