Propbot User Manual

Project : Propbot Autonomous Robot

Authors : Team- 50,

Amr Almoallim (AA) - 83386714,

Apoorv Garg (AG) - 39485545,

Cole Shanks (CS) - 54950860,

Johanan Agarwal (JA) - 29188166,

Sajjad Al-Kazzaz (SA) - 23401565

Affiliation : UBC Radio Science Lab

ROS Installation and Workspace Setup

The following goes over how to install ROS and setup the catkin workspaces.

Note: this guide is adapted from https://github.com/hannahvsawiuk/Propbot/wiki/Installation

1) Installing ROS Melodic

Follow the instructions here to install the melodic version of ROS:
http://wiki.ros.org/melodic/Installation/Ubuntu

Ensure that you install the **Desktop-Full Install** version.

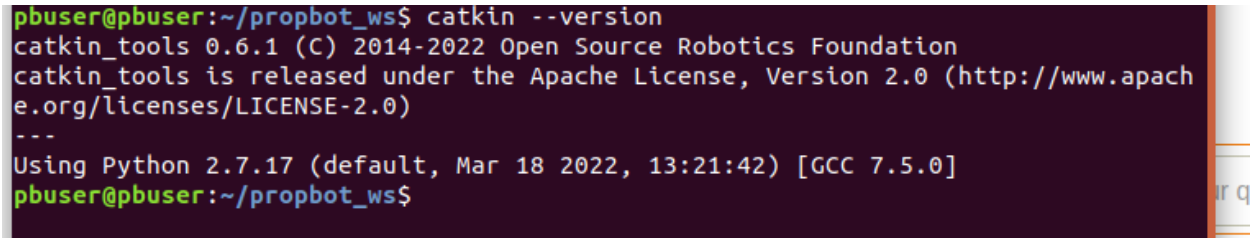
A note on catkin

Documentation: http://wiki.ros.org/catkin

catkin is the build system used for ROS. ROS comes with catkin. After doing the installation of ROS, run the following command:

catkin --version

You should see an output similar to this:



If you do not, there may have been an issue installing catkin. To resolve this, run the following command and try the version check command again:

sudo apt-get install ros-kinetic-catkin python-catkin-tools

2) Create the workspaces

Third Party Packages workspace

In your home directory, make a workspace directory for the third party dependencies needed for Propbot

1. cd ~/
2. mkdir propbot_3pp_ws

Propbot workspace

1. In your home directory, make the propbot workspace directory
2. cd ~/
3. mkdir propbot_ws
4. Then, make the src directory and clone the Propbot repo into the new src directory.
5. cd ~/propbot_ws
6. mkdir src
7. cd src
8. git clone https://github.com/UBC-RSL-PropBot/PropBot

3) Build an third party dependencies for Propbot

Now that ROS has been installed, we need to start by building third party dependencies. The most important being Google Cartographer for our application. Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations.

Cartographer documentation:

- General information: https://google-cartographer.readthedocs.io/en/latest/
- Integration with ROS: https://google-cartographer-ros.readthedocs.io/en/latest/

3a) Run the custom Cartographer setup script

The script to setup the custom version of Cartographer is in vehicle_autonomy/scripts. To run the setup for the propbot_3pp_ws, run the following command:

~/propbot_ws/src/PropBot/scripts/setup_third_party.sh propbot_3pp_ws

3b) Build the 3PP workspace

Run the following to initialize and build the workspace

cd ~/propbot_3pp_ws

catkin build

This will take some time (up to **30 minutes**), so please be patient.

4) Setup the Propbot Workspace

Now that the custom Cartographer version has been built, it will be extended with the Propbot packages.

4a) Make Propbot an extension of third party workspace

To initialize the workspace and indicate that propbot_ws will be an extension of 3pp_ws, run the following commands

cd ~/propbot_ws

catkin init

catkin config --extend ../propbot_3pp_ws/devel

The --extend argument explicitly sets the workspace you want to extend. In this case, we want the packages in propbot to extend the custom cartographer setup.

For more information on catkin config, see here:
https://catkin-tools.readthedocs.io/en/latest/verbs/catkin_config.html

4b) Install dependencies

To ensure that everything runs smoothly, the dependencies specified by the Propbot packages need to be installed. To accomplish this, run the following:

Ubuntu

cd ~/propbot_ws

rosdep install --from-paths src --ignore-src

Alternative Distros If you are not using Ubuntu, the --os argument is needed, e.g. --os=ubuntu:bionic:

cd ~/propbot_ws

rosdep install --os=<distro info> --from-paths src --ignore-src

A note on rosdep

rosdep is a dependency manager for ROS that helps you install external dependencies in an OS-independent manner.

rosdep documentation:

- Wiki: http://wiki.ros.org/rosdep
- Command reference: https://docs.ros.org/independent/api/rosdep/html/commands.html

4c) Build the mapviz plugins

Prior to building the entire propbot_ws, the mapviz plugs need to be built. Run the following

catkin build mapviz_plugins

4d) Build the workspace

Finally, build the entire propbot_ws:

cd ~/propbot_ws

Depending on what computer is running the build, not all packages need to build successfully.

For Autonomy Computer, run "catkin build propbot_autonomy propbot_control propbot_description propbot_drivers propbot_mission propbot_navigation propbot_slam propbot_state_estimation propbot_common_msgs propbot_util"

For Mission Command Center computer, run "catkin build mapviz_plugins propbot_mission_gui propbot_common_msgs propbot_util"

For Simulation Computer, run "catkin build propbot_gazebo propbot_rviz propbot_simulation"

Now your propbot_ws directory should have the following structure

```
├── build/          # Build space
├── devel/          # Devel space
├── logs/           # Logs space
└── src/            # Source space
    └── Propbot/    # Propbot packages
```

For more information on catkin workspace mechanics, see here:
https://catkin-tools.readthedocs.io/en/latest/mechanics.html

Autonomy Setup Procedure

These instructions describe how to set-up and run an autonomous waypoint mission using Propbot. If you are not executing the mission in simulation, skip the instructions for *Setting up the simulation computer*.

Note: this guide is adapted from the one offered by a previous team, see https://github.com/hannahvsawiuk/Propbot/wiki/User_Guide for the original.

Setting up communication between multiple machines

In order to set up the mission command centre to communicate with the vehicle autonomy computer and simulation computer, perform the following steps:

1. Make sure all computers are on the same network. Do a simple ping test to make sure they can communicate with each other.
2. Make sure each hostname can be resolved to the correct ip address. To do this modify the /etc/hosts files of each computer to include the hostnames of the other computers, and the ip addresses of those hosts.
3. On the vehicle autonomy computer, run: export ROS_MASTER_URI=http://vehicle_autonomy_computer_hostname:11311.
4. On the vehicle autonomy computer, run: roscore.
5. On the mission command centre and simulation computer, run: export ROS_MASTER_URI=http://vehicle_autonomy_computer_hostname:11311.

Make sure the export command is either in ~/.bashrc or is run in every terminal used.

Setting up the simulation computer

On the simulation computer, build all of the packages inside the *simulation* directory.

catkin build propbot_gazebo propbot_rviz propbot_simulation

The following commands will launch simulation environment that is modeled off of this area on UBC campus.

If you would like use a static simulation environment launch the following file:

roslaunch propbot_simulation ubc_student_union_sim.launch

If you would like to use a dynamic simulation environment launch this file instead of the one above:

roslaunch propbot_simulation ubc_student_union_sim_people.launch

Setting up the vehicle autonomy computer

Build all of the packages inside the *vehicle_autonomy* and *common* directory:

catkin build propbot_autonomy propbot_control propbot_description propbot_drivers propbot_mission propbot_navigation propbot_slam propbot_state_estimation propbot_common_msgs propbot_util

Launch the autonomy stack:

roslaunch propbot_autonomy autonomy.launch

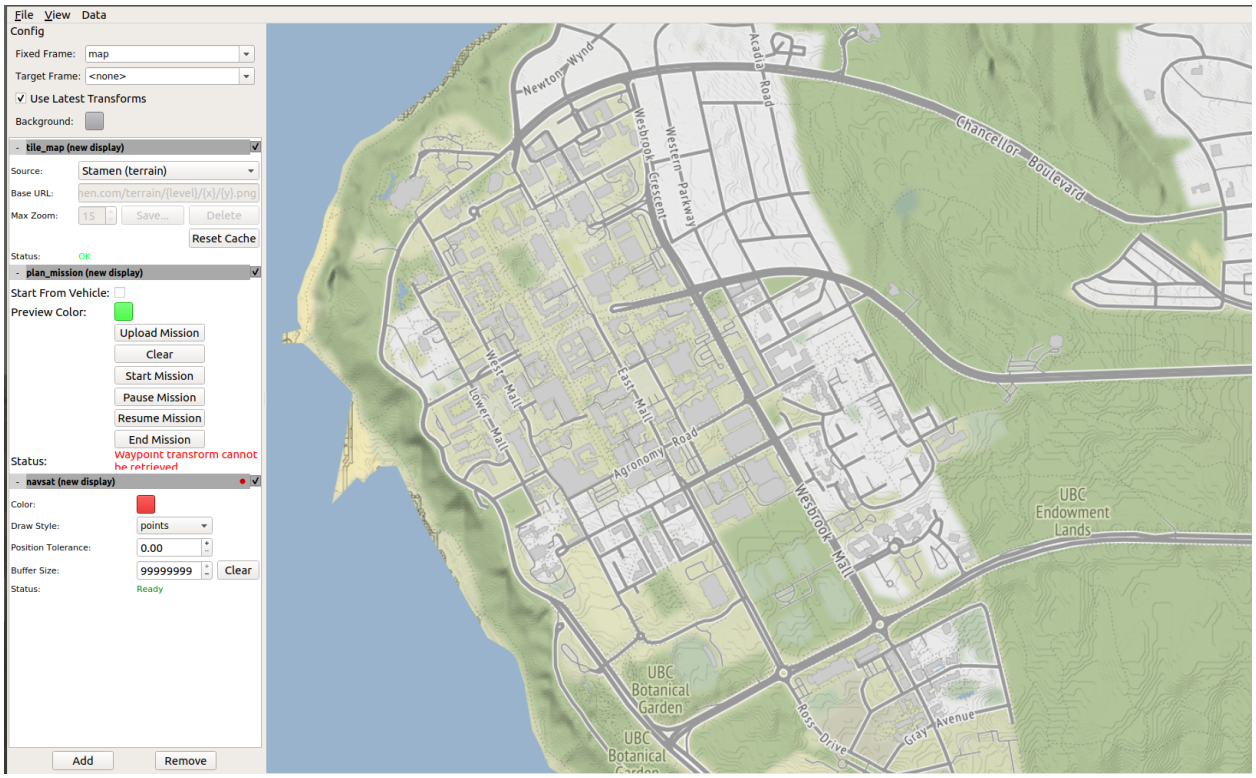
Setting up the mission command centre

Build all of the packages inside the *mission_command_centre* and *common* directory:

catkin build mapviz_plugins propbot_mission_gui propbot_common_msgs propbot_util

Launch the mission command centre GUI:

roslaunch propbot_mission_gui mapviz.launch



The above shows the default GUI configuration. If you would like to display satellite map tiles follow the instructions from this repository

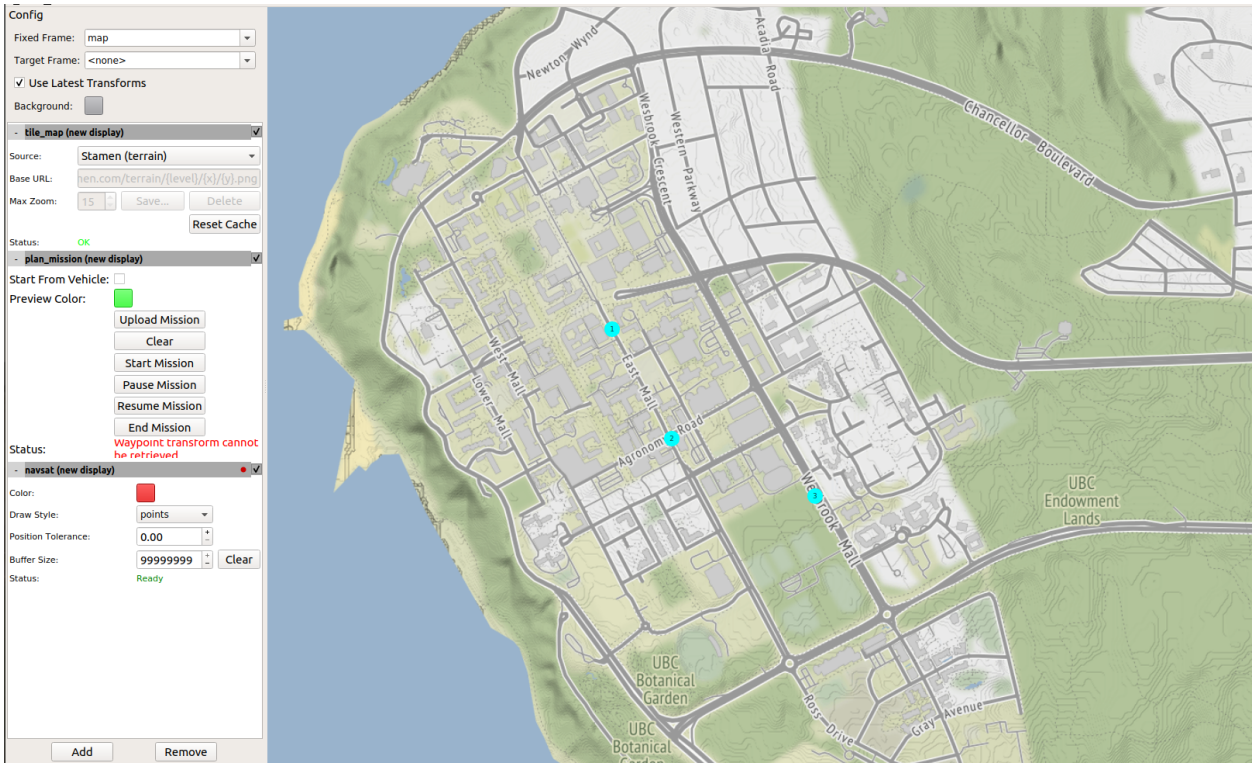
Executing the mission

Configure and command a mission using the *plan_mission* section of the control panel:

1. Select mission waypoints by clicking on the map and clear them as you like by selecting *Clear*
2. Once you are satisfied with your mission configuration, upload the mission to propbot by selecting *Upload Mission*
3. Start the mission by selecting *Start Mission*
4. You may pause, resume, or end the mission at anytime by selecting *Pause Mission*, *Resume Mission*, and *End Mission*

5. Monitor the robots progress on the display

Note: You must wait until a mission is finished or end the mission before uploading a new one



Teleoperation Setup Procedure

Elevating Propbot:

1. Ensure E-Stop is engaged. Ensure that the battery is switched off
2. Place the jack under one end of Propbot and start elevating that side
3. Put a jack stand in place under the elevated side
4. Remove jack and repeat steps 2-3 on the other side of Propbot
5. Engage E-Stop

Flashing Code onto Vehicle Interface board:

1. Download and Install Microsoft Visual studio code from https://code.visualstudio.com/download by following the instructions.
2. Download and Install Microsoft Visual studio code from https://platformio.org/install/ide?install=vscode by following the instructions.
3. Clone Github Repo from https://github.com/UBC-RSL-PropBot/Propbot_Vehicle_Interface_Firmware
4. Navigate to Project_Teleoperation/Teleoperation_Propbot/
5. Open the folder in VS Code and build the project.
6. Flash onto Arduino Mega Board via USB connection
7. Switch on battery
8. Disengage E-Stop
9. Power on Transmitter and ensure successful pairing with receiver
10. Make sure all switches are in the UP position and place the left throttle down to activate the transmitter for teleoperation control
11. Place SWA in the down position (disengaging remote E-Stop)

12. Place the left throttle in the UP position. Verify that all wheels are moving forward
13. IF some wheels are moving backward, check the direction PIN connection to the motor controller
14. Place the left throttle in the DOWN position. Verify that all wheels are moving backward
15. Place the right throttle in the UP position. Verify that Propbot is executing a Right-Hand turn
16. Place the right throttle in the DOWN position. Verify that Propbot is executing a Left-Hand turn
17. Engage wheels of Probot and trigger remote E-Stop by placing SWA in the UP position. Repeat for all motions (ie: Go forward, Go backward, Turn right, Turn left)
18. Engage E-Stop. Power off Transmitter

Lowering Propbot:

1. Ensure that E-Stop is engaged. Ensure that the battery is switched off
2. Place the jack under one end of Propbot remove the jack stand
3. Begin lowering that side
4. Place the jack under the other end of Propbot remove the jack stand
5. Begin lowering that side until Propbot is securely placed with all six wheels firmly on the ground

FLYSKY i6x Transmitter and receiver (Remote Controller)

- Left throttle UP: Move forward
- Left throttle DOWN: Move backwards
- Right throttle UP: Right turn
- Right throttle DOWN: Left turn
- SWA UP: Switch to E-Stop mode
- SWB DOWN: Switch to Autonomy Mode