Project Notes - 1

Joseph Grant

September 22, 2017

1 ROS

Robot Operating System (ROS) is a set of frameworks for robot software developement. This set of frameworks offers a libraries for C++, Python, and Lisp und a BSD license, and many more libraries exist under open source libraries. ROS is structured as a graph where each node in the graph can do some kind of processing which might be running a state machine, path-planning, or controls for acuators. Each ROS node is its own process where one master node is responsible for connecting them all and facilitating communication. Normal nodes communicate via topics. A topic is a unidirectional bus which defines a specific format to the data sent between nodes called a message. One or more nodes can be a publisher to a topic where each message will be sent to all nodes which subscribe to that topic. The master node facilitates this communication by allowing the subscribing nodes to negotiate a connection to publishing nodes through it. The style of communication between publisher and subscriber nodes is not fixed, but is most commonly TCP/IP.

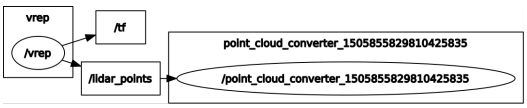

Figure 1: V-REP as a Node

Above is an example graph for ROS. In the graph VREP is a node wish is publishing the topics tf and lidar_points. Another node, the point_cloud_converter, subscribes to the lidar_points topic.

2 V-REP

Virtual Robot Experimentation Platform (V-REP) is a platform which allows for robot simulation and control in a 3-D environment. V-REP offers an API and IDE for developement of LUA scripts in customising simulations while also supporting C/C++ plugins. There is also a remote API for V-REP to communicate with other programs. In this project V-REP will be used in conjunction with ROS. To use V-REP with ROS the RosInterface library is used such that V-REP is instantiated as a node in ROS. As a node V-REP will be able to publish transform data and sensor data from the simulation and also be able to subscribe to any topics which might facilitate controls. V-REP also provides for the simulation of a Velodyne VPL-16 LIDAR sensor. Important to note that the lidar sensor is a vision sensor in V-REP and thus utilizes the GPU, with the only other aspect of the simulation which uses GPU being the renderer.

3 Simulation Setup

A Pioneer P3DX is used as the base robot, to which a VPL-16 lidar has been added on to the top platform.

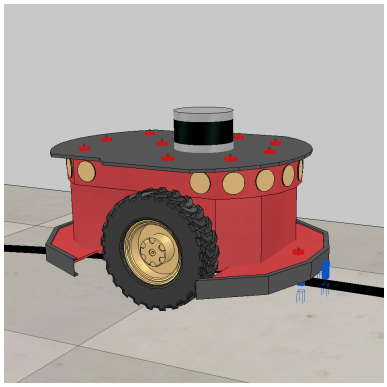

Figure 2: Robot in V-REP

This robot begins in a small maze-like environment. In V-REP a basic line following algorithm is implimented using three downward facing vision sensors and a black line on the floor to follow.

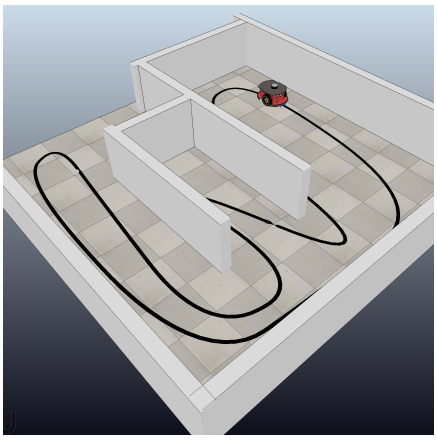

Figure 3: Robot in Environment

While the simulation is running it will publish data to two topics. One topic is for the pointcloud data generated by the lidar and the other for the current pose of the robot. The V-REP ROS node will publish these two topics using the rosInterface library.

4 SLAM

Simultaneous Localization and Mapping (SLAM) is a type of algorithm used in robotics for generating maps of a robot's environment and determining the robot's position within that map. Generally a Kalman Filter or Particle Filter is used to provide the estimation of the robot's pose given existing sensor data and the history of observed landmarks or features within the environment. This section will detail

4.1 Feature Extraction

5 Appendix

5.1 V-REP as a ROS Node

As a ROS node, V-REP can communicate with other ROS nodes using ROS services, publishers and subscribers. This is accomplished using the RosInterface API provided by Coppelia Robotics. These instructions were created for an Ubuntu 16.04 LTS system running ROS Kinetic and V-REP 3.4.0.

To setup RosInterface with V-REP

1. Download the V-REPs stub generator and install the software it needs.

```
sudo apt-get install -y git cmake
    python-tempita python-catkin-
    tools python-lxml xsltproc
git clone -q https://github.com/
    fferri/v_repStubsGen.git
```

2. Add path of V-REP stubs generator to the search path of python

```
export PYTHONPATH=$PYTHONPATH:$PWD
```

3. Create the catkin workspace and initialize it

```
mkdir -p ~/project_ws/src
cd ~/project_ws
catkin init
```

4. Download RosInterface and build it

```
cd src/
git clone --recursive https://
    github.com/fferri/
    v_repExtRosInterface.git
    vrep_ros_interface
catkin build
```

5. Source project workspace

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

6. Add VREP_ROOT to the bashrc, assuming it is located in the home folder. Then copy the RosInterface library which was compiled earlier to the V-REP folder.

```
echo 'export VREP_ROOT="/home/user
    /V-REP_PRO_EDU_V3_3_2_64_Linux
    /"' >> ~/.bashrc
source ~/.bashrc
cp -iv devel/lib/
    libv_repExtRosInterface.so "
    $VREP_ROOT/"
```

7. Setup alias for v-rep

```
echo 'alias vrep="$VREP_ROOT/vrep.
    sh"' >> ~/.bashrc
```

8. This setup procedure can be validated by running ROS and V-REP, then listing the active ROS nodes. Each command is ran in its own terminal.

```
roscore
vrep
rosnode list
```

To validate the functionality of the RosInterface the predefined scenes rosInterfaceTopicPublisherAndSubscriber.ttt and controlTypeExamples.ttt can be used.