

ME3720 Lab 1

April 19, 2024

1 Introduction

The lab objective is to get a basic understanding of a feedback control system and how to tune it. It is comprised of three sections. The first is to develop code and tune a PID controller for depth/altitude using the vertical thrusters. The second is to develop code and tune PID gains for a heading controller with the horizontal thrusters. The third is to conduct accurate waypoint navigation using the available thrusters and actuators. There are two vehicles available to use in simulation, the Fusion and the REMUS.

2 Overview

The simulation approach is a combination of MATLAB and VMWare with an Ubuntu and Gazebo/UUV Simulator running in the remote Operating System (see figure 1). A MATLAB m-script will be written using the MATLAB ROS Toolbox that permits publishing and subscribing messages that creates the desired AUV motion in the Gazebo environment.

3 Setup

Follow these steps for starting/managing the processes inside the Virtual Machine running Ubuntu:

1. Inside the VM Ubuntu OS, open a terminal window. Type the following commands:

```
>> echo $ROS_IP  
>> echo $ROS_HOSTNAME
```
2. They should both be the same ipaddress (see figure 3).
3. If they aren't do the following:
Open a new terminal window and type

```
>> ifconfig
```

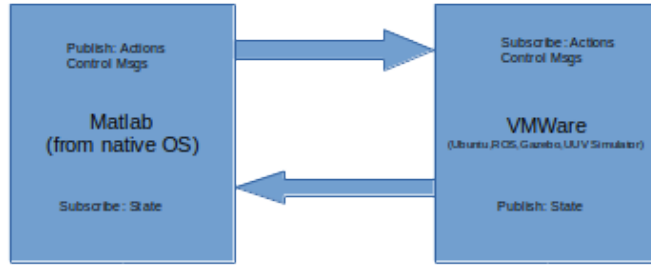
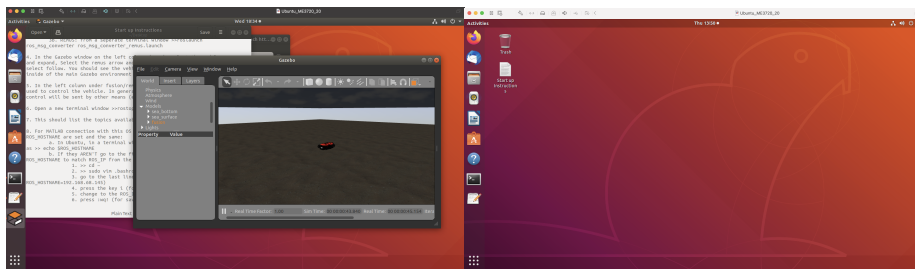


Figure 1: MATLAB-Virtual Machine Information Flow.



(a) Ubuntu with processes running

(b) Ubuntu without processes running

Figure 2: Two examples of Ubuntu on startup of the virtual machine

If the top line (ens33: inet 192.168.68.145), do this `>> sudo ifconfig ens33 192.168.68.145`

Your machine should look the same as figure 4

enter `>> source .bashrc`

4. Check `$ROS_IP` and `$ROS_HOSTNAME` they should be the same now.
5. There may be three windows open when you start the VMWare Ubuntu OS. They are:

Gazebo window with the Fusion AUV in the undersea environment.

A terminal window with three processes running

A script for launching the underwater world (within Gazebo)

A script for launching the fusion/remus vehicle into the world

A script for launching a ROS process for converting messages published by MATLAB into messages that Gazebo uses to move the AUV.

A text file with the instructions for launching the above scripts.

6. If your screen looks like figure 2b. do the following (for the fusion AUV):

Open a terminal window and three tabs within the window and start the following processes:

`>> roslaunch uuv_gazebo_worlds auv_underwater_world.launch`

`>> roslaunch fusion_description upload.launch mode:=default x:=0 y:=0 z:=-10 namespace:=fusion`

`>> roslaunch ros_msg_converter ros_msg_converter_fusion.launch`

7. **NOTE: The password, if it goes into screen saver mode, is cavr.**

Follow these steps within MATLAB in you native OS:

1. Start MATLAB

2. Within the MATLAB Command Window:

Set a variable 'ipaddress' to the ipaddress for ROS_HOSTNAME on the remote OS (ipaddress = '192.168.68.145')

Start the roscore (rosinit(ipaddress))

Type 'rostopic list' in the MATLAB command window and you should see a list of the available topics. This should include a list of the available topics with fusion (or REMUS). See figure 5.

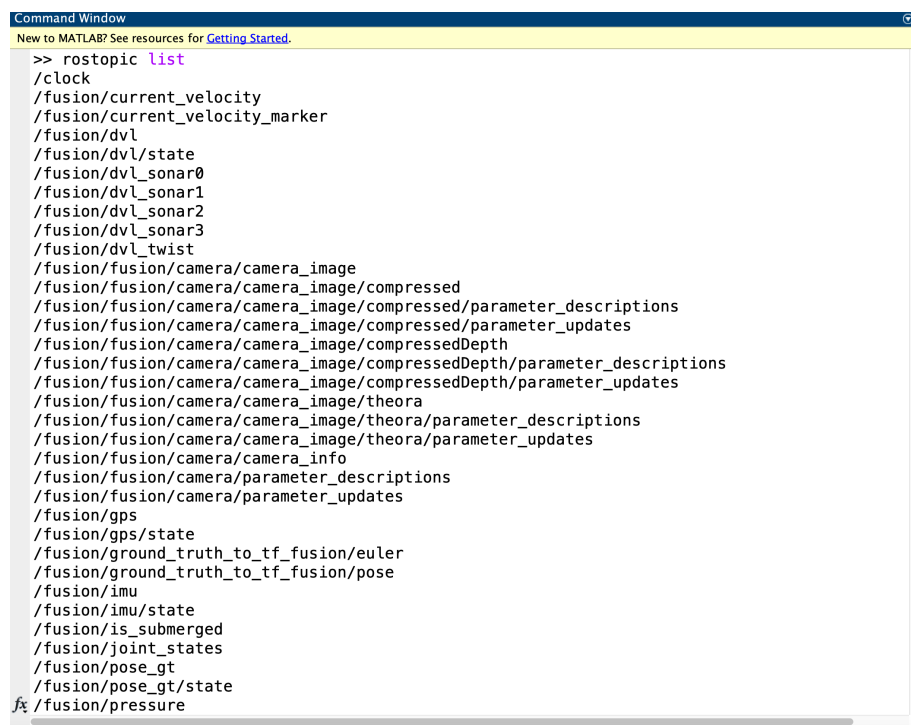
Now you are ready to write your MATLAB scripts. You will need to write code for subscribing and publishing to ROS messages. The subscription messages will allow you to receive state messages from Gazebo and the publication

```
cavr@ubuntu: ~  
File Edit View Search Terminal Help  
cavr@ubuntu:~$ echo $ROS_IP  
192.168.68.145  
cavr@ubuntu:~$ echo $ROS_HOSTNAME  
192.168.68.145  
cavr@ubuntu:~$
```

Figure 3: Both the ROS_IP and ROS_HOSTNAME should be the same ipaddress.

```
cavr@ubuntu: ~  
File Edit View Search Terminal Tabs Help  
/home/cavr/cod... x /home/cavr/cod... x /home/cavr/code... x cavr@ubuntu: ~ x  
cavr@ubuntu:~$ ifconfig  
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.68.145 netmask 255.255.255.0 broadcast 192.168.68.255  
    inet6 fe80::20c:29ff:fe4a:2357 prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:4a:23:57 txqueuelen 1000 (Ethernet)  
    RX packets 8926 bytes 708985 (708.9 KB)  
    RX errors 0 dropped 166 overruns 0 frame 0  
    TX packets 22132 bytes 5711559 (5.7 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 2326400 bytes 328033590 (328.0 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 2326400 bytes 328033590 (328.0 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
cavr@ubuntu:~$
```

Figure 4: Result of changing the ip address using the ifconfig command

A screenshot of a MATLAB Command Window. The window has a title bar that says "Command Window" and a yellow banner at the top that says "New to MATLAB? See resources for [Getting Started](#)." The command prompt shows the command "rostopic list" has been executed, resulting in a list of ROS topics. The topics are listed as follows:

```
>> rostopic list
/clock
/fusion/current_velocity
/fusion/current_velocity_marker
/fusion/dvl
/fusion/dvl/state
/fusion/dvl_sonar0
/fusion/dvl_sonar1
/fusion/dvl_sonar2
/fusion/dvl_sonar3
/fusion/dvl_twist
/fusion/fusion/camera/camera_image
/fusion/fusion/camera/camera_image/compressed
/fusion/fusion/camera/camera_image/compressed/parameter_descriptions
/fusion/fusion/camera/camera_image/compressed/parameter_updates
/fusion/fusion/camera/camera_image/compressedDepth
/fusion/fusion/camera/camera_image/compressedDepth/parameter_descriptions
/fusion/fusion/camera/camera_image/compressedDepth/parameter_updates
/fusion/fusion/camera/camera_image/theora
/fusion/fusion/camera/camera_image/theora/parameter_descriptions
/fusion/fusion/camera/camera_image/theora/parameter_updates
/fusion/fusion/camera/camera_info
/fusion/fusion/camera/parameter_descriptions
/fusion/fusion/camera/parameter_updates
/fusion/gps
/fusion/gps/state
/fusion/ground_truth_to_tf_fusion/euler
/fusion/ground_truth_to_tf_fusion/pose
/fusion/imu
/fusion/imu/state
/fusion/is_submerged
/fusion/joint_states
/fusion/pose_gt
/fusion/pose_gt/state
/fusion/pressure
```

Figure 5: MATLAB screen capture of some of the fusion ROS message topics.

messages will be actions that will be converted into messages that Gazebo will use for motion.

The topics for publishing are the actuators of the AUV (either Fusion or REMUS): For Fusion they are as follows:

1. `/bow_port_thruster` - (positive rpms sends the vehicle counterclockwise)
2. `/bow_stbd_thruster` - (positive rpms sends the vehicle counterclockwise)
3. `/vert_port_thruster` - (positive rpms sends the vehicle downward)
4. `/vert_stbd_thruster` - (positive rpms sends the vehicle upward)
5. `/aft_port_thruster` - (positive rpms sends the vehicle clockwise)
6. `/aft_stbd_thruster` - (positive rpms sends the vehicle clockwise)
7. `/aft_vert_thruster` - (positive rpms sends the vehicle up (pitches the vehicle))

All these messages are of type `Float64`. The maximum RPMs for the Fusion thrusters is between `[1200,-1200]`. The Fusion subscription topic is `/fusion/pose_gt` this is a standard `nav_msgs/Odometry` message. See what is in the message at http://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html

While sending these messages to the topics listed above they are converted to the following topics within Ubuntu. I mention this because there is a tool called `rqt` that can be used to directly control the commands to the thrusters. This allows you to reset the vehicle pose or test control manually. See figure 6. We'll go over this in class.

The topics for feedback control for the REMUS are as follows:

1. Thrusters (Propellers) - ROS message type `std_msgs/Float64`
 - `/fwd_vert_thruster` (-5000 to 5000 rpm)
 - `/fwd_horiz_thruster`
 - `/aft_vert_thruster`
 - `/aft_horiz_thruster`
 - `/aft_thruster` (0-2000 rpm)
2. Fins (Actuators) - ROS message type `std_msgs/Float64`
 - `/rudder_fin` (-1.4 to 1.4 radians deflection)
 - `/pitch_fin`
3. State (Coming from Gazebo) - ROS message type `geometry_msgs/PoseStamped`
 - `/remus/pose_gt`

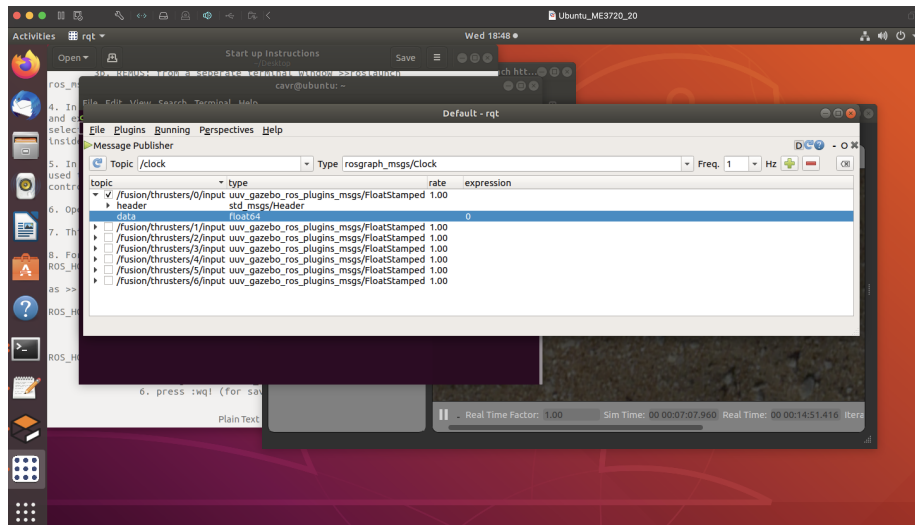


Figure 6: rqt permits setting thruster or actuator values directly to vehicle.

With these ROS topics you have the ability to do feedback control with the AUV. There is a MATLAB shell script (me3720_pid_lab_shell.m) available on Sakai that gives an example of how to subscribe and publish to ROS topics. Start with developing an AUV depth controller with the vertical thrusters by subscribing to the state message and publishing to the appropriate message(s). The goal is the development an effective hovering PID depth controller. Within MATLAB, plot the performance of the PID controller and optimize the K_p , K_i and K_d gains.

The second objective is to use the actuators/thrusters for developing a hovering heading control again optimizing the PID gains. It requires simultaneously hovering while bringing the AUV to the desired reference heading.

The third objective is navigation in the horizontal plane using the appropriate actuators/thrusters. The goal is to accurately drive a box. For all of the lab objectives you are welcome and encouraged to use all support available with MATLAB PID (and other) functions.

4 Writeup

Each group will submit a detailed report on the project. It will include a description of the approach for determining the PID gains and a progression of the results. A minimum of 8 runs for each objective is required. Give justification for your final selection of gains. Provide critical analysis - this can include limitations and analysis of the controllers and the simulation. Collaboration within the group is encouraged but I expect that each member of the group will have a working laptop/desktop implementation. Include individual areas of

responsibility within the report.