5 ROS Code for Human Subject Tests

5.1 Package Descriptions

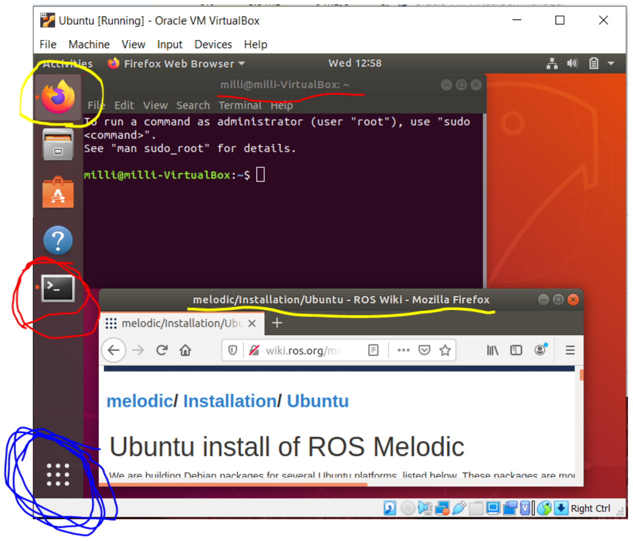
Below is a summary of the packages used in the human subject experiment.

- vr_exp_ros - This package contains the main scripts and launch files for running the code.
  - Launch files:
    - run_waypointcontrol_experiment.launch - launches the waypoint swarm controller, Tanvas client, sets up websocket, and saves all topics to a bag file (the bag file section has been commented out). It takes 2 arguments for the id, complexity (referring the the environmental complexity, either 'high' or 'low').
    - run_ergodic_experiment.launch - launches the ergodic swarm controller, Tanvas client (if control is either direct or shared and can incorporate a user input), dist_autonomy scripts for creating the autonomous distribution (if the control is either shared or auto), sets up websocket, and saves all topics to a bag file (the bag file section has been commented out). It takes 3 arguments for the id, complexity (referring the the environmental complexity, either 'high' or 'low'), and control (either direct, shared, or auto).
    - run_nocontrol_experiment.launch - launches websocket and saves all topics to a bag file (the bag file section has been commented out). It takes 2 arguments for the id, complexity (referring the the environmental complexity, either 'high' or 'low').
  - Scripts:
    - main.py - This script creates the /target_distribution for the ergodic controller for the specific direct, shared, and auto conditions set as a rosparam in the launch file. It subscribes to /input, /visual_dist', and /obj_dist.
    - swarm_listener.py - Listens to output of decentralized swarm ergodic controller and transforms drone positions to the appropriate workspace coordinates and publishes to Unity.
    - waypoint_control_swarm.py - This script publishes the swarm poses so that each drone follows the list of coordinates that were published to the /input topic in a loop.
- dist_autonomy - This package creates the visual occlusion and object detection distributions and publishes them to topics /visual_dist and /obj_dist, both of type Target_dist.msg, a custom message in the dist_autonomy package. To create the visual occlusion distribution, visual_occlusion.cpp subscribes to the player's current position (topic /person_position of type person_position.msg in the user_input package). Within visual_occlusion.cpp, the assumed sight capabilities of the player are defined (we assume the player can see 45 degrees in each direction and 10 unity units in front) and the locations of the buildings are hard-coded. The code references a ROS parameter to find out whether the 'low' or 'high' complexity environment is being used. obj_detect_dist.py subscribes to the /obj_dist topic of type object_position.msg in the dist_autonomy package and creates gaussian peaks around the detected objects. If the same object is re-detected, the location of the gaussian distribution is updated. If it has been 15 seconds since the object was last detected, it is removed from the distribution. pub_grid.py subscribes to the /target_distribution and publishes the distribution in a GridMap format for visualization in rviz.
- user_input - This package connects to a TCP socket to receive user inputs from the computer running the Tanvas and send back the player's position. All user commands are published to the /input topic from input_array.msg in the user_input package. It subscribes to /person_position topic from the person_position.msg in the user_input package. This package also contains several custom messages for the purposes of data collection: player_info.msg, treasure_info.msg, and object_position.msg. Topics with these messages are published to from unity. For it to work, a rosnode (in this case save_data.cpp) must subscribe to these topics, even if the rosnode is never run.

- file_server - This package creates the websocket for communication with unity. It depends on the rosbridge_server package and can be launched using ros_shape_communication.launch. More documentation on how to use and set up the C# websocket can be found at “https://github.com/siemens/ros-sharp”.
- rt_ergodic_control - This repo contains a python library for the RT ergodic controller for nonlinear systems. The most up-to-date code can be found here, “https://github.com/i-abr/rt_ergodic_control”. We used the code and parameters in this repo.
- network-setup.sh - bash script for setting up network settings.

5.2 Installation of Ubuntu, ROS, Required Packages, and Our Code

1. Download the Ubuntu desktop 18.04.4 LTS from ubuntu.com/download/desktop. If you are planning on installing Ubuntu on a computer with either Windows or Mac operating system, you likely will want to partition your hard drive. There are plenty of instructions online for how to accomplish this. It is a good idea to back up your hard drive beforehand because disks sometimes get accidentally erased during this process.
2. Go through the Ubuntu installer selecting the “Minimal installation” (unless you would like to install the full version) and “Erase disk and install Ubuntu” (This will only erase the portion of your disk allocated to the virtual machine, not your entire hard drive). Otherwise, select default parameters and create whatever username and password you wish. It will ask you to restart– go ahead. Flip quickly through the “What’s new with Ubuntu” and exit any pop-ups asking for a Ubuntu software update.
3. Install ROS melodic.
   (a) Open up the a terminal window by clicking the grid of nine dots circled in blue in Figure 1, then clicking on the icon that looks like the one circled in red in Figure 1. The window with the terminal is underlined in red in Figure 1.
   (b) Open up Firefox by clicking the icon circled in yellow in Figure 1. Navigate to the following webpage: http://wiki.ros.org/melodic/Installation/Ubuntu
   (c) Follow the steps 1.2-1.4 on the webpage above. You may skip Step 1.1. Please do the “Desktop Install” at step 1.4. If it says “Do you want to continue”, enter “Y” and press enter.
   (d) Before doing step 1.5, run these two commands: “sudo apt-get install python-pip” and “sudo pip install -U rosdep”. This complete step 1.5 as instructed.
   (e) For step 1.6, run the first two suggested commands: “echo “source /opt/ros/melodic/setup.bash” >> /.bashrc” and “source /.bashrc”
   (f) Complete step 1.7. If it says “Do you want to continue”, enter “Y” and press enter.
4. Set up ROS workspace
   (a) Install catkin by typing this in the terminal: “sudo apt-get install ros-melodic-catkin”
   (b) Follow the instructions at the following link to set up a workspace skipping the set up for python 3 (we will be using python 2). http://wiki.ros.org/catkin/Tutorials/create_a_workspace
5. Install necessary packages
   (a) Update apt-get: “sudo apt-get update” and “sudo apt-get upgrade”
   (b) rosbridge_server: “sudo apt-get install ros-melodic-rosbridge-server”
   (c) install python package gym: “pip install gym”
   (d) install some text editor like atom (“sudo add-apt-repository ppa:webupd8team/atom”, and “sudo apt-get install atom”)
6. Install our code


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Wed 12:58
milli@milli-VirtualBox: ~
File Edit View Search Terminal Help
melodic/Installation/Ubuntu - ROS Wiki - Mozilla Firefox
melodic/Installation/Ubu
wiki.ros.org/me
Right Ctrl

```
To run a command as administrator (user "root"), use "sudo
<command>".
See "man sudo_root" for details.

milli@milli-VirtualBox:~$
```

melodic/ Installation/ Ubuntu

Ubuntu install of ROS Melodic

We are building Debian packages for several Ubuntu platforms, listed below. These packages are mo

Figure 1: Ubuntu Desktop. blue-directory for accessing apps, yellow-firefox, red-terminal. This picture was taking in an oracle virtual machine.

(a) Go into the src folder using "cd ./src/".

(b) Clone repository from git: "git clone https://github.com/mschlafly/VR_exp_ROS.git"

(c) Move up a directory: "cd .."

(d) Rebuild workspace (this must be done after any change is make to C++ code or a package is added): "catkin_make". It should finish with no errors.

5.3 How to run the code

1. Open terminal and navigate to your catkin_ws: "cd ./catkin_ws/"

2. Source: "source devel/setup.bash" and "source src/VR_exp_ROS/network-setup.sh".

3. If cpp script or a Cmake file has been changed, "catkin_make". Depending on what you changed, you might need to "catkin clean" first.

4. Follow the instructions in Section 5.4.

5. If you have access to the Tanvas, make sure to set up the TCP server on the Windows OS before launching ROS files by following the instructions in Section 4.3.2.

6. Run the code! If you do not have access to a Tanvas, we recommend you run run_ergodic_experiment.launch with the control parameter set to auto. The ergodic launch file takes 3 arguments and the waypoint control and no control each take 2. Below are examples:

   - roslaunch vr_exp_ros run_ergodic_experiment.launch id:=11 control:=auto complexity:=low
   - roslaunch vr_exp_ros run_waypointcontrol_experiment.launch id:=04 complexity:=high
   - roslaunch vr_exp_ros run_nocontrol_experiment.launch id:=100 complexity:=low

7. Press play in unity or launch .exe file.

5.4 Set up networking

1. Decide on a static IP address to use. We used 192.168.1.2
2. Set project hostname to the static IP
   (a) Open terminal and go to the /etc file: "cd /etc/"
   (b) Open the hosts file in the text editor of your choice using sudo, for example "sudo atom hosts"
   (c) After the first two lines defining the local host, insert the following two lines to the hosts script: "192.168.1.2 darpa_station" and "192.168.1.2 test_station"
3. Connect the computer to the static IP
   (a) Click the drop-down menu in the upper-right hand corner of the screen.
   (b) Click the wi-fi symbol/drop-down menu and select "Wi-Fi Settings".
   (c) Connect to your personal router, then press the gear symbol next to it.
   (d) Next, we want to manually set the IPv4 address to 192.168.1.2. Click on the "IPv4" tab, select "Manual", then fill in the address to be 192.168.1.2 and the netmask to be 255.255.255.0
   (e) Exit the pop-up window and turn on and off the "Wi-Fi" in the upper-right hand corner of the window.
4. Change locations in which the IP address is manually defined. The IP address used to launch the websocket in the vr_exp_ros launch files will need to be changed. It will also need to be changed in Unity(refer to the Unity notes).
5. If something isn't working correctly, follow instructions in Section 4.4.

5.5 Important parameters for ergodic control and how to change them

- You may want to change the amount the ergodic control values the information from the user and the autonomy. This can be changed by editing the distribution weights in update_dist within main.py.
- Ergodic control parameters such as the number of basis functions, time horizon, and batch size for sampling can be changed when the controller is called for each agent in rt_ergodic_control/rt_erg_lib/agent.py line 38. The defaults values for these parameters can be chosen in rt_ergodic_control/rt_erg_lib/ergodic_control.py
- Control saturation limits can be set on lines 37 and 38 in rt_ergodic_control/rt_erg_lib/ergodic_control.py