Principles of Safe Autonomy ECE 498 SM
Lecture 2: RightHook Simulator

Pulkit Katdare and Tianqi Liu
Lecture outline

- RightHook Simulator
- FastX, Docker
- ROS
RightHook Simulator

- RightHook Demo
- Closed loop testing environment with deterministic results
- Real world testing is expensive and time consuming
Virtual Machine

- Virtual machines (VM) are emulation programs of operating systems (OS)
- VM provides virtual hardware to run multiple instances of different OS
- We use VMs in this class to get access to GPU resources
Virtual Machine

- Virtual machines (VM) are emulation programs of operating systems (OS)
- VM provides virtual hardware to run multiple instances of different OS
- We use VMs in this class to get access to GPU resources.
FastX

▶ We want visualization for simulation scenarios
▶ FastX offers a virtual desktop of EWS Linux virtual machine
▶ Before accessing our VM through FastX, make sure you are connecting to illinois.net WiFi or using a EWS machine (either actual or virtual) or connecting through university VPN.
▶ When launch a new VM session, choose MATE(VirtualGL).
01 is the VM number. You will be assigned a number later.
Docker

▶ Docker offers virtual environment on operating system level
▶ Applications run in docker containers, which allow us to package all dependencies in one docker image and ensures compatibility on different OS.
▶ Applications are isolated from each other and the operating system
▶ RightHook simulator runs inside docker container.
Why Dockerize Everything?

▶ Easy to develop/deploy applications on different platforms
▶ Performance increase
▶ Container orchestration (Kubernetes)
Image from: https://aws.amazon.com/docker/
Robotics Operating System (ROS)

▶ Developed in 2007, ROS was developed by Stanford Artificial Intelligence Lab (SAIL) to build modular software stack for robotics project
▶ ROS is very suitable in cases wherein you have multiple robot modules that are needed to run in sync with each other
Software Architecture

- **Node**: Executable code is called nodes

- **Topics**: Communication protocol in ROS

- **Messages**: Data structure expected by ros-topics is called ros-messages
Example:
Example:

- **v1-sensors**
  1. Camera
  2. Controller

- **v2-sensors**
  1. Camera
  2. Controller
  3. Lidar
Vehicle 1 sensors
- actuation
- camera

Vehicle 2 sensors
- actuation
- camera
- LIDAR

ROS Topics
Vehicle 1 sensors
- camera
- actuation

Vehicle controller 1
- Publisher
- Subscriber

Vehicle 2 sensors
- LIDAR
- camera
- actuation

Vehicle controller 2
- Publisher
- Subscriber

ROS Nodes
- ROS Topics
Publisher/Subscriber?

- Let’s look at some MP0 node

- A car initially moving with 4.5 m/s has to autonomously stop near a stop sign
Demonstration
Visualizing topics and nodes (rqt_graph)
Roslaunch

- Important for large projects with multiple nodes
- In form a .xml file
Writing Scripts in RightHook

▶ Mount Docker

```bash
rh_sim/minimaps:c000140725e017ab00810eea6ab55e1cc9310182
```

▶ Ros network setup

```bash
export ROS_MASTER_URI=http://172.17.0.2:11311
export ROS_IP=172.17.0.1
export ROS_HOSTNAME=172.17.0.1
```
Running scripts

- Git clone the repository
  https://gitlab.engr.illinois.edu/GolfCar/mp-release.git
- Go into the cloned repository
- `catkin_make`
- `./setup.sh`
- `./run.sh`
- `source devel/setup.bash`
- `roslaunch mp0 run_mp.launch`
Writing scripts in RightHook

- Running a particular scenario
  ```
curl http://172.17.0.2:8080/connected_launch -X POST -d "42"
  ```

- Run the Simulator
  ```
  rosrun rh_msgs advance_step_loop.py
  ```
  OR
  ```
  roslaunch mp0 run_mp.launch
  ```
rosbag

- Rosbag is a set of tools for recording from and playing back to ROS topics.
- By calling the rosbag API, we can record different types of ROS messages
- rosbag record <topic_name>