IMAV 2017 Virtual Challenge Guide

1. Introduction

A simulation environment was created to represent the IMAV 2017 indoor challenge to speed up the development process of the drone Navigation, Guidance and Control algorithms developed by the teams. This guide provides some help on how to set up and use the tools.

The simulation is built on ROS packages developed by the Technische Universität Darmstadt (repository here) and the Technical University of Munich (repository here).

2. Environment setup

a) Operating System

The simulation requires a Linux environment (latest Ubuntu is recommended, testes with 16.04.1). It can be a virtual system on a Windows host system, but please note that the recommended virtualization environment is VMware, because Hyper-V and VirtualBox does not perform well enough for the 3D simulation and visualization.

b) ROS, Gazebo, Repository

Run setup.sh by:

```
chmod +x setup.sh
./setup.sh
```

It will ask for your password because of the sudo.

Also, add to ~/.bashrc:

```
export ROS_IP=[your ip]
```

c) Simulink

To connect to the ROS network on the virtual machine or on an external PC from MATLAB/Simulink, please refer our documentation.

d) Notes

There is an OpenGL issue with VMware. If Gazebo crashes with the message:

```
VMware: vmw_ioctl_command error Invalid argument.
```

Then execute:

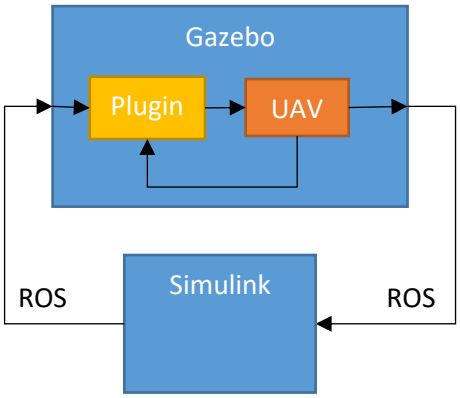
```
echo "export SVGA_VGPU10=0" >> ~/.bashrc
```

This changes the environment variable to use an earlier version of OpenGL. More info here.

Also, there is a bug in gazebo_ros_pkgs, which makes the simulation crash if you try to remove a camera object from the scene. More info here. You can pull the latest ros-simulation:kinetic-devel branch and build from source, or wait until they roll it out to the release branch. Please note that this does not cause issues during the simulation, only when you try to remove (or respawn) the drone.

3. Simulation

The simulation represents the indoor challenge at IMAV 2017. Gazebo handles the physics of the rigid body (UAV), while the attached plugins take care of the sensor readings and stabilization of the system. The IMU data is used directly from the simulation in the inner loop (without noise), and the forces and torques are exerted on the body mimicking the effect of the propellers.



The outer loop (involving Simulink) handles the high-level planning based on the image and sensor data acquired by the virtual instruments. This is done through ROS by subscribing the corresponding topics, and publishing the velocity commands.

a) Launching the simulation

The simulation can be run by

```
roslaunch imav_2017 imav_indoor.launch
```

The robots can be spawned by:

```
cd ~catkin_ws/src/IMAV_2017_Virtual_Challenge/urdf
./spawnrobot.sh 1
```

You can spawn multiple drones by changing the ID (the argument of script).

To remove a drone from the simulation run:

```
rosservice call gazebo/delete_model '{model_name: quadrotor_[ID]}'
```

Please note that in the current gazebo_ros_pkgs release this will crash the simulation (see Notes above).

b) Controlling the drone from Simulink

The drone can be controlled in Gazebo by sending ROS topics from Simulink, thanks to the ROS blocks from the Robotics System Toolbox.

Several example models are available for you to experiment with that and build up a drone control software that will allow to meet the mission elements:

- *manualControl*: control the drone manually on the X, Y, Z and Yaw axis
- *simpleMovement*: make the drone reach a target (the QR code) by defining a trajectory with the Signal Builder block
- *pathFollowing*: make the drone reach a target using the Pure Pursuit block from the Robotics System Toolbox
- *pathFollowingWithObstacleAvoidance*: make the drone avoid obstacles using the Vector Field Histogram block from the Robotics System Toolbox

For the moment, these models control the drone with ID 1, but you can change the topics of the *Subscribe* and *Publish* blocks to control another drone.

To connect Simulink with the ROS server, follow these steps:

1. Open Simulink
2. In the Tools menu, navigate to Robotics Operating System, and click on Configure Network Addresses
3. In ROS Master Node, set Network Address to Custom
4. In ROS Master Node, fill the Hostname/IP Address field with the IP of your virtual machine
5. In ROS Master Node, let the Port field with the default port ROS is running on (11311)
6. Test the connectivity using the Test button

If a model can control the drone but does not receive any data from the drone, make sure you have set the *ROS_IP* environment variable in the VM before running ROS.