HiFi Sim Setup

Sim Team

Updated: March 25, 2024

1 Updating the Sim

If you have not yet set up the sim, move on to section 2. This section contains dates for when some files in sim have been updated so you can replace them in the PX4 workspace (use the documentation below if you have forgotten the directories).

- Model SDF - 03/29/24
- World SDF - 03/29/24

2 Installing Dependencies

2.1 Install Gazebo-Classic

Use these commands in the terminal to install the correct version of gazebo:

```
sudo apt remove gz-garden
sudo apt install aptitude
sudo aptitude install gazebo libgazebo11 libgazebo-dev
```

2.2 Installing the PX4 Workspace, ROS2, and the Micro XRCE-DDS Agent

Follow the directions in this link to install the PX4 workspace for ROS2: PX4 Documentation Link
Please make sure to run the "make px4_sitl" command to see if it compiles completely; you can move on to setting up ROS2 once it does. Please make sure you are installing ROS2 Humble and not Foxy, and even if you have ROS2 Humble already installed, it is recommended to at the very least install the rest of the dependencies listed in the documentation. Finally, stop after installing the Micro XRCE-DDS Agent & Client. Do not run the command to make px4_sitl with gz_x500, as that uses Gazebo Ignition instead of Gazebo Classic.

3 Setting up the PX4 Workspace

3.1 Adding our Drone Models

First, please pull from our git repository HiFi branch. Once you have done so, navigate to the **PX4 Models and Worlds** folder and copy both the grass_plane and hex_drone folders into this directory:

/PX4-Autopilot/Tools/simulation/gazebo-classic/sitl_gazebo-classic/models

Then, copy the **M-Air.world** and **hex_drone.world** files into the **worlds** directory inside **sitl_gazebo-classic** parent folder in the PX4-Autopilot Workspace.

3.2 Jumping Around more of the Workspace

Because Git was giving us a hard time, some files are stored in our Google Drive. Please navigate to this directory in our Google Drive and download the file named **6012_gazebo-classic_hex_drone**:

/M-LADIS/Simulation Sub-Team/High Fidelity Sim/Backups/PX4 File Backup Once you have done so, navigate to these directories in the PX4-Autopilot workspace and paste the **6012_gazebo-classic_hex_drone** file into them:

/PX4-Autopilot/ROMFS/px4mu_common/init.d-posix/airframes
/PX4-Autopilot/build/px4_sitl_default/etc/init.d-posix/airframes

Do not worry, once you add this airframe configuration file to the build directory, you shouldn't have to repeat the process anymore even if you rebuild the workspace. Now, navigate to this directory and open the **sitl_targets_gazebo-classic.cmake** file:

/PX4-Autopilot/src/modules/simulation/simulator_mavlink

Once opened, add **hex_drone** inside the set(models) list, as well as **M-Air** to the set(worlds) list. Also make sure to add them in alphabetical order to reduce the chance of any issues. Once you finish with setup, you can go ahead and run this command in the terminal, making sure you are in the PX4-Autopilot directory:

```
make px4_sitl gazebo-classic_hex_drone
```

You could also run this command to see the drone in M-Air:

```
make px4_sitl gazebo-classic_hex_drone__M-Air
```

If the workspace isn't building (throws errors), try installing some dependencies using these commands:

```
sudo apt-get install libgstreamer-plugins-base1.0-dev gstreamer1.0-plugins-bad
gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly -y
```

To test even more of PX4's functionality, you could run this command in a separate terminal to see the drone take off:

```
commander takeoff
```

4 Running the Simulation with the Launch File

4.1 Running the Agent

First, run this command in a separate terminal:

```
MicroXRCEAgent udp4 -p 8888
```

If you haven't already, please run the simulation (in a separate terminal) using one of the *make px4_sitl* commands from section 2.2. Also, to prevent issues with certain plugins, install the ros-gazebo package (even if you think you have it installed) using this command:

```
sudo apt install ros-humble-gazebo-ros-pkgs
```

4.2 Running the Launch File

Open a new terminal, navigate to the directory of our git repository (M-LADIS-sim HiFi branch). Then, navigate through this filepath:

/M-LADIS-sim/'PX4 Aux'/ws_sensor_combined

Once in this directory, run the following commands:

```
colcon build
source install/setup.bash
ros2 launch ros roslauncher.launch.py
```

It is possible that colcon build will not complete successfully, especially if the workspace has recently been altered but not tested (remember, colcon build is essentially compiling the code). If you need to urgently use the simulation, you could see which files are causing the issues and move those files out of the workspace, taking care to also comment out the *add_executable* lines in the corresponding *CMakeLists.txt* file. It is also possible that the name of the launch file has changed since this documentation has been updated. If colcon build runs but the ros2 launch command doesn't, please navigate to the directory listed below and take note of the name of the launch file, replacing it with the last argument in the ros2 launch terminal command:

/M-LADIS-sim/'PX4 Aux'/ws_sensor_combined/ros/launch