Brief Documentation for Swarm Playground

Please visit https://github.com/ZJU-FAST-Lab/EGO-Planner-v2 for software updates. This link will be available after the paper is published.

Install Required Software

Install ROS (Robot Operating System) following https://www.ros.org.
If you are using ubuntu **16.04**, please install armadillo solver for solving quadrotor dynamics in simulation and ros joy package for using Xbox Controller:

```
sudo apt-get install -y libarmadillo-dev ros-kinetic-joy
```

On ubuntu **18.04**, armadillo is installed by default. You only need to install ros joy package:

```
sudo apt-get install -y ros-melodic-joy
```

On ubuntu **20.04**, ros joy package is the only requirement as well:

```
sudo apt-get install -y ros-noetic-joy
```

Play in Swarm Playground

Please watch the videos in *main_ws/src/*, *formation_ws/src/*, *interlaced_flight_ws/src/*, and *tracking_ws/src/* to start the code.

Computing Time

If you want to know the computing time, please fix your CPU frequency to its maximum using **cpufreq** tool. That's because the computing time of our planner is always around 1 ms, which is too short to let the CPU feel necessary to increase the frequency from the default power-saving mode.
Install **cpufreq**:

```
sudo apt-get install -y cpufrequtils
```

Set CPU to its maximum frequency:

```
sudo cpufreq-set -g performance
```

Print current frequency:

```
sudo cpufreq-info
```

If everything goes well, you will see the current frequency close to 5 GHz.
Then the code should be compiled in release mode using

```
catkin_make -DCMAKE_BUILD_TYPE=Release
```

Trouble Shooting

If any pcl relevant packages are not found, execute

```
sudo apt-get install ros-<ros-distribution>-pcl*
```

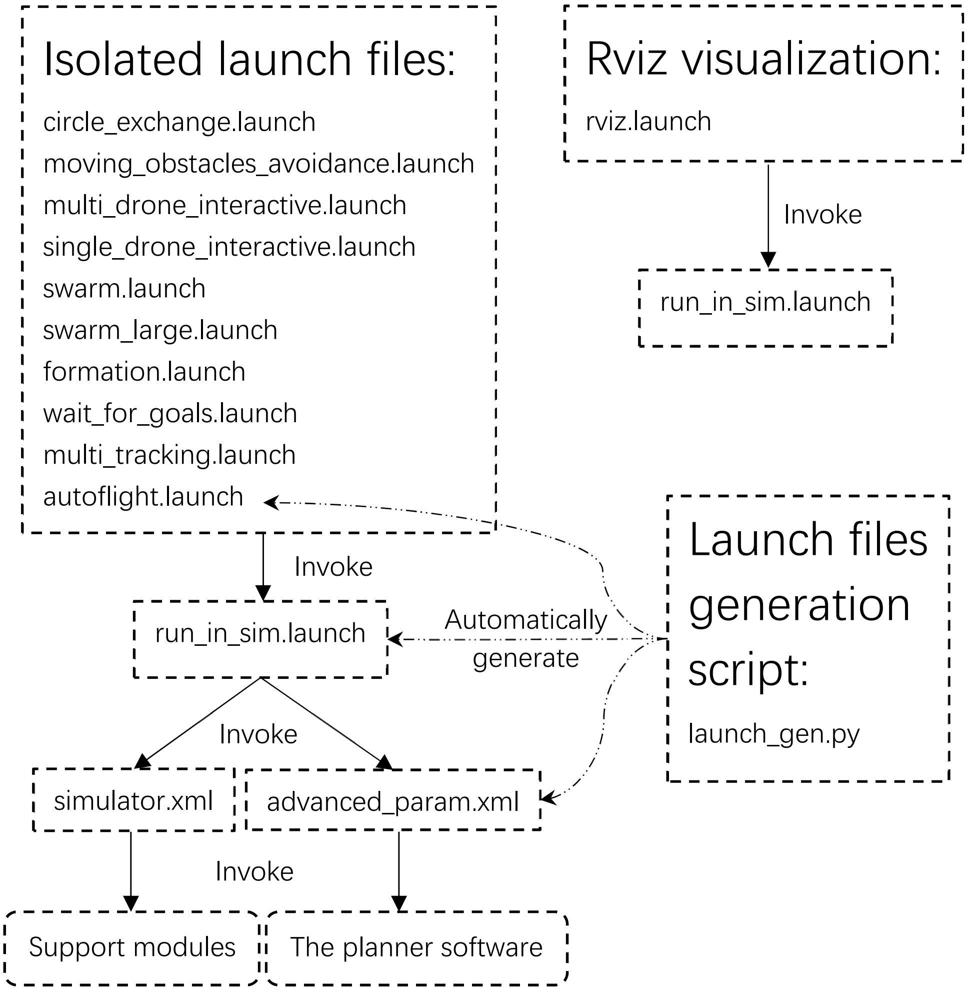
If any opencv relevant packages are not found, execute

```
sudo apt-get install ros-<ros-distribution>-opencv*
```

Replace the marker “<ros-distribution>” with your ros distribution on your system, which is “kinetic” on ubuntu 16.04, “melodic” on ubuntu 18.04, and “noetic” on ubuntu 20.04.

The Launch File Architecture

Launch files are nested in *src/planner/plan_manage/launch/*. Here is the architecture.



Isolated launch files:
circle_exchange.launch
moving_obstacles_avoidance.launch
multi_drone_interactive.launch
single_drone_interactive.launch
swarm.launch
swarm_large.launch
formation.launch
wait_for_goals.launch
multi_tracking.launch
autoflight.launch

Rviz visualization:
rviz.launch

Launch files generation script:
launch_gen.py

Commonly Used Parameters

In the files which invoke *run_in_sim.launch* :

map_size_x, map_size_y, map_size_z	The map size should be big enough to let the given goals fall inside the map.
init_x, init_y, init_z	The initial position of the drone.
target_x, target_y, target_z	The goal position of the drone.
drone_id	Starting from 0, each drone should be assigned a unique drone id.

In the file *run_in_sim.launch* :

max_vel, max_acc, max_jer	Maximum system dynamics allowed.
flight_type	Set to 1 to receive goals from ros topics, 2 to use given waypoints in launch files.
point_num	(valid if flight_type is 2) The number of waypoints to chase.
point0_x ~ point4_z	waypoints to set. You can add more waypoints following these naming rules.

In the file *advanced_param.xml* :

grid_map/resolution	Map resolution, typically set to half of the drone radius.
grid_map/obstacles_inflation	Obstacles inflation value, the real inflation equals: [ceil(**obstacles_inflation** / **resolution**) + 1] * **resolution**.
optimization/weight_*	Weights of each penalties.
optimization/obstacle_clearance	A distance that the trajectory is required to stay away from inflated obstacles.
optimization/swarm_clearance	Minimum allows distance between drones.

License

GPLv3

Maintenance

If you encounter any issues, please raise them in the GitHub repository or contact Xin Zhou directly.
GitHub repository: https://github.com/ZJU-FAST-Lab/EGO-Planner-v2
Email: iszhouxin@zju.edu.cn