03 Weekly report 4

October 12, 2021

Information: *The weekly report supposed to be finished in week 7*

Written by: *Zihao Xu*

Last update data: *Oct.13.2021*

1 Get control of basic movements

1.1 Existing controller

1.1.1 Trajectory controller

The current simulation script is using the *lee_position_controller_node* in the **RotorS**. The controller requires full information of the positions, velocities and accelerations with timestamps. As mentioned in last report, I don't think this controlling method is appropriate for the obstacle avoiding task. Therefore, some other controlling methods need to be investigated or created.

1.1.2 Controller for joystick usage

The **RotorS** package actually provides two built-in controllers. In the examples provided by the **RotorS** package, the other controller is used for joystick usage. Considering the way that a joystick controls the UAV, this controller can probably control the very basic movements in an efficient way.

1.2 Investigation to the C++ scripts

To find out how to make use of the second controller, I looked carefully into the original scripts and added detailed comments. Here are some important notes.

- According to the header file *rotors_control/common.h*, there should be commands to directly controll the movement described by roll, pitch, yaw rate and thrust. Similar to what we used to control the trajectory, the corresponding topic is *command/roll_pitch_yawrate_thrust*.

[ ]:

```
static const std::string kDefaultCommandMotorSpeedTopic =
    mav_msgs::default_topics::COMMAND_ACTUATORS; // "command/motor_speed";
static const std::string kDefaultCommandMultiDofJointTrajectoryTopic =
    mav_msgs::default_topics::COMMAND_TRAJECTORY; // "command/trajectory"
static const std::string kDefaultCommandRollPitchYawrateThrustTopic =
    mav_msgs::default_topics::COMMAND_ROLL_PITCH_YAWRATE_THRUST;
    // "command/roll_pitch_yawrate_thrust"
static const std::string kDefaultImuTopic =
    mav_msgs::default_topics::IMU; // "imu
```

```
static const std::string kDefaultOdometryTopic =
    mav_msgs::default_topics::ODOMETRY; // "odometry"
```

- The input message type for this controller is *mav_msgs::EigenRollPitchYawrateThrust*

mav_msgs::EigenRollPitchYawrateThrust

Type	Name
double	pitch
double	roll
Eigen::Vector3d	thrust
double	yaw_rate

- The controller is based on the implementation from T. Lee et al paper
- The required parameters for starting the node are the same as that of the *lee_position_controller_node.*

2 Get the visual information

In the last meeting, Jianwen noted that I could probably get the visual input from the image topics. Here're some conclusions after trial.

- Current UAV model uses two different cameras acting like two eyes.
- The corresponding topics are */hunter/vi_sensor/left/raw* and */hunter/vi_sensor/left/raw*
- Due to the location of cameras, the front motor occurs in the view, which seems need to be fixed for further development.

3 Summary and future work

In the past two weeks, I

- Read through the controller scripts and find out how to use the controller to control basic movements.
- Tested the current visual information in several gazebo environments.

In the next two weeks, I'm planning to

- Edit the launch file of UAV simulation and change the controller node.
- Modify the UAV model to put the cameras in better positions.
- Build a subscriber for the visual information and check what I can do with the visual information (which kind of vision-based obstacle avoidance).