UAV Project

機械所 洪得瑜 309611087

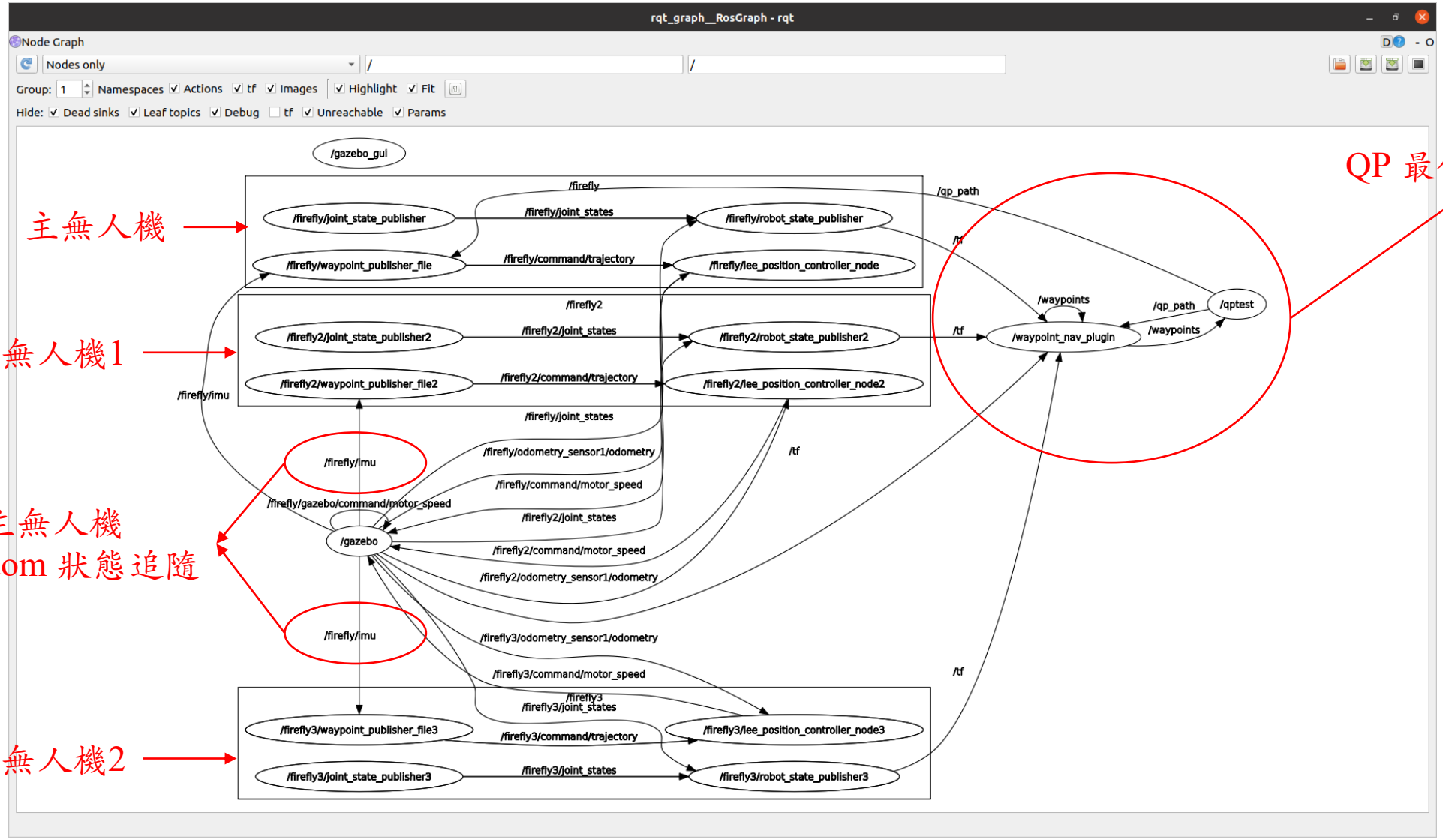
UAV Final project

- In this project, each team has to complete the following part:

1. (10%) Select 10 points and visualize them on Rviz.
2. (40%) Implement QP trajectory planning using points you selected in part 1, and visualize trajectories on Rviz.
3. (10%) Run UAV‘s gazebo simulation and make it fly in trajectories you planned with the following topic: “/${arg mav_name}/command/trajectory". Notice that the heading of it should be towards its velocity direction.
4. (40%) Construct a UAV formation team with one leader and two followers, the heading direction of them should be the same, and followers should remain behind the direction the leader is facing. (Coordinate transformation is needed)

- Final Project 目標
- 在Rviz中設置10點位置由QP軌跡規劃產生最優路徑,將路徑點發布在節點上,由主無人機接收路徑規劃軌跡飛行,在增加2部追隨無人機,且追隨無人機依照主無人機狀態飛行,及固定相對位置在主無人機位置。

rqt_graph


rqt_graph__RosGraph - rqt
Node Graph
Nodes only
Group: 1 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/gazebo_gui
/firefly
/firefly/joint_state_publisher
/firefly/joint_states
/firefly/robot_state_publisher
/firefly/waypoint_publisher_file
/firefly/command/trajectory
/firefly/lee_position_controller_node
/qp_path
/tf
/firefly2
/firefly2/joint_state_publisher2
/firefly2/joint_states
/firefly2/robot_state_publisher2
/firefly2/waypoint_publisher_file2
/firefly2/command/trajectory
/firefly2/lee_position_controller_node2
/firefly/imu
/waypoints
/qptest
/waypoint_nav_plugin
/firefly/joint_states
/firefly/odometry_sensor1/odometry
/firefly/command/motor_speed
/firefly/gazebo/command/motor_speed
/firefly2/joint_states
/gazebo
/firefly2/command/motor_speed
/firefly2/odometry_sensor1/odometry
/firefly/imu
/firefly3/odometry_sensor1/odometry
/firefly3/command/motor_speed
/firefly3
/firefly3/joint_states
/firefly3/waypoint_publisher_file3
/firefly3/command/trajectory
/firefly3/lee_position_controller_node3
/firefly3/joint_state_publisher3
/firefly3/joint_states
/firefly3/robot_state_publisher3
主無人機
Follow 無人機1
根據主無人機
IMU 及 odom 狀態追隨
Follow 無人機2
QP 最佳化軌跡

Rviz wapoints to QP path

1.修改發佈/waypoints到指定坐標系world.

2.任意點選10點點座標後由Publish Waypoints 到/waypoints節點上

3.而/waypoints有發布點座標在由qptrajectory節點接收座標位置

4.透過qptrajectory節點計算QP路徑結果,並將規劃結果發布到/qp_path節點上

```
void waypoint_pub(std::vector<trajectory_profile>& qp_path){
    nav_msgs::Path data_pub;
    data_pub.header.frame_id = "world";
    data_pub.header.stamp = ros::Time::now();
    for(int i = 0; i < qp_path.size(); i++){
        geometry_msgs::PoseStamped qp_point;
        qp_point.pose.position.x = qp_path[i].pos[0];
        qp_point.pose.position.y = qp_path[i].pos[1];
        qp_point.pose.position.z = 1; // set default hight
        data_pub.poses.push_back(qp_point);
    }
    qp_pub.publish(data_pub);
};

void waypointCallback(const nav_msgs::Path::ConstPtr& msg){
    std::vector<geometry_msgs::PoseStamped> data = msg->poses;
    std::vector<geometry_msgs::PoseStamped>::const_iterator it = data.begin();
    /* QP trajectory data */
    qptrajectory plan;
    path_def path;
    std::vector<trajectory_profile> path_data;
    std::vector<trajectory_profile> path_new;
    /* pop out raw path data */
    for(it; it != data.end(); it++){
        geometry_msgs::Point pos;
        pos.x = it->pose.position.x;
        pos.y = it->pose.position.y;
        pos.z = it->pose.position.z;
        //ROS_INFO("x = %f, y = %f z = %f", pos.x , pos.y, pos.z);
        /* push in path data*/
        trajectory_profile tmp;
        tmp.pos.x() = pos.x;
        tmp.pos.y() = pos.y;
        tmp.pos.z() = pos.z; // Modify to const
        tmp.vel << 0,0,0;
        tmp.acc << 0,0,0;
        path_data.push_back(tmp);
    }
    ROS_INFO("Take waypoint data!");
    /* QP path generate */
    int setpoint = path_data.size();
    const int t = 1; /* set const interval time */
    double sample = 0.01;
    double max;
    ROS_INFO("Totoal setpoint : %d", setpoint);
    /* push to new generat path */
    for(int i = 0; i < setpoint - 1; i++){
        path.push_back(segments(path_data[i], path_data[i+1], t));
    }
    path_new = plan.get_profile(path, path.size(), sample);
    max = path_new.size();
    ROS_INFO("New path point total : %f", max);

    /* pub waypoint */
    waypoint_pub(path_new);
};
```

QP trajectory move and spawn UAV

1. 由主無人機節點接收/qp_path路徑,並重新計算desire yaw
2. 重新將/qp_path資料型態改為trajectory_msgs::MultiDOFJointTrajectory重新發布
3. 而發布的路徑為命令與UAV已作Topic連結,即發布到指定UAV節點控制無人機.

```
/* Publish QP trajectoy */
void traj_pub(std::vector<WaypointWithTime>& waypoints){
    trajectory_msgs::MultiDOFJointTrajectoryPtr msg_traj(new trajectory_msgs::MultiDOFJointTrajectory
    msg_traj->header.stamp = ros::Time::now();
    msg_traj->points.resize(waypoints.size());
    msg_traj->joint_names.push_back("base_link");
    int64_t time_from_start_ns = 0;
    for (size_t i = 0; i < waypoints.size(); ++i) {
        WaypointWithTime& wp = waypoints[i];

        mav_msgs::EigenTrajectoryPoint trajectory_point;
        trajectory_point.position_W = wp.position;
        trajectory_point.setFromYaw(wp.yaw);
        trajectory_point.time_from_start_ns = time_from_start_ns;

        time_from_start_ns += static_cast<int64_t>(wp.waiting_time * kNanoSecondsInSecond);

        mav_msgs::msgMultiDofJointTrajectoryPointFromEigen(trajectory_point, &msg_traj->points[i]);
    }
    wp_pub.publish(msg_traj);
    ROS_INFO("Successful poblish to node!");
};


void pathcallback(const nav_msgs::Path::ConstPtr& msg){
    std::vector<geometry_msgs::PoseStamped> data = msg->poses;
    std::vector<geometry_msgs::PoseStamped>::const_iterator it = data.begin();
    std::vector<WaypointWithTime> waypoints;
    geometry_msgs::PoseStamped temp;
    //const float DEG_2_RAD = M_PI / 180.0;
    /* store waypoint vector*/
    for(it; it != data.end(); it++){
      if(it == data.begin()){
          waypoints.push_back(WaypointWithTime(                0.03,   // Time
                                               it->pose.position.x,   // x
                                               it->pose.position.y,   // y
                                               it->pose.position.z,   // z
                                                       init_yaw));   // 0
          temp.pose.position = it->pose.position;
       } else {
          double dx = it->pose.position.x - temp.pose.position.x;
          double dy = it->pose.position.y - temp.pose.position.y;
          double theta = std::atan2(dy, dx);

          // ROS_INFO("heading %f", theta);
          waypoints.push_back(WaypointWithTime(                0.03,   // Time
                                               it->pose.position.x,   // x
                                               it->pose.position.y,   // y
                                               it->pose.position.z,   // z
                                                           theta));  //
          temp.pose.position = it->pose.position;
       }

    }
    // Publish traj
    traj_pub(waypoints);
};
```

Follower UAV

1.Spawn UAV 在 mav_with_waypoint_publisher.launch file 中增加無人機模型

```
<group ns="firefly2">
  <include file="$(find rotors_gazebo)/launch/spawn_mav2.launch">
    <arg name="mav_name" value="firefly2" />
    <arg name="model" value="$(find rotors_description)/urdf/mav_generic_odometry_sensor.gazebo" ,
    <arg name="enable_logging" value="$(arg enable_logging)" />
    <arg name="enable_ground_truth" value="$(arg enable_ground_truth)" />
    <arg name="log_file" value="$(arg log_file)"/>
  </include>
  <node name="lee_position_controller_node2" pkg="rotors_control" type="lee_position_controller_n
    <rosparam command="load" file="$(find rotors_gazebo)/resource/lee_controller_firefly.yaml" />
    <rosparam command="load" file="$(find rotors_gazebo)/resource/firefly.yaml" />
    <remap from="odometry" to="odometry_sensor1/odometry" />
  </node>
  <node name="hovering_example2" pkg="rotors_gazebo" type="hovering_example2" output="screen"/>
  <node name="waypoint_publisher_file2" pkg="rotors_gazebo" type="waypoint_publisher_file2" outpu
  <node name="robot_state_publisher2" pkg="robot_state_publisher" type="robot_state_publisher" />
  <node name="joint_state_publisher2" pkg="joint_state_publisher" type="joint_state_publisher" />
</group>

  <group ns="firefly3">
  <include file="$(find rotors_gazebo)/launch/spawn_mav3.launch">
    <arg name="mav_name" value="firefly3" />
    <arg name="model" value="$(find rotors_description)/urdf/mav_generic_odometry_sensor.gazebo" ,
    <arg name="enable_logging" value="$(arg enable_logging)" />
    <arg name="enable_ground_truth" value="$(arg enable_ground_truth)" />
    <arg name="log_file" value="$(arg log_file)"/>
  </include>
  <node name="lee_position_controller_node3" pkg="rotors_control" type="lee_position_controller_n
    <rosparam command="load" file="$(find rotors_gazebo)/resource/lee_controller_firefly.yaml" />
    <rosparam command="load" file="$(find rotors_gazebo)/resource/firefly.yaml" />
    <remap from="odometry" to="odometry_sensor1/odometry" />
  </node>
  <node name="hovering_example3" pkg="rotors_gazebo" type="hovering_example3" output="screen"/>
  <node name="waypoint_publisher_file3" pkg="rotors_gazebo" type="waypoint_publisher_file3" outpu
  <node name="robot_state_publisher3" pkg="robot_state_publisher" type="robot_state_publisher" />
  <node name="joint_state_publisher3" pkg="joint_state_publisher" type="joint_state_publisher" />
</group>
```

Follower UAV

2.並修改 CMakeLists.txt 增加編譯路徑

```
add_executable(waypoint_publisher_file2 src/waypoint_publisher_file2.cpp)
target_link_libraries(waypoint_publisher_file2 ${catkin_LIBRARIES})
add_dependencies(waypoint_publisher_file2 ${catkin_EXPORTED_TARGETS})

add_executable(waypoint_publisher_file3 src/waypoint_publisher_file3.cpp)
target_link_libraries(waypoint_publisher_file3 ${catkin_LIBRARIES})
add_dependencies(waypoint_publisher_file3 ${catkin_EXPORTED_TARGETS})


add_executable(hovering_example2 src/hovering_example2.cpp)
target_link_libraries(hovering_example2 ${catkin_LIBRARIES})
add_dependencies(hovering_example2 ${catkin_EXPORTED_TARGETS})

add_executable(hovering_example3 src/hovering_example3.cpp)
target_link_libraries(hovering_example3 ${catkin_LIBRARIES})
add_dependencies(hovering_example3 ${catkin_EXPORTED_TARGETS})
```

Follower UAV

3.在waypoint_publisher_file2.cpp 及 wapoint_publisher_file3.cpp 訂閱主無人機位置及姿態資訊/firefly/imu 及 /firefly/odometry_sensor1/position 計算出相對固定位置及姿態飛行

```
    // firefly2 relative position setup
    follower_goal.x = -1;
    follower_goal.y = -1;

    // firefly3 relative position setup
    follower_goal.x = -1;
    follower_goal.y =  1;

// Relative position function call
geometry_msgs::Point LeadertoFollower(geometry_msgs::Point &follower_goal, geometry_msgs::Point &lea
    float temp_x = follower_goal.x;
    float temp_y = follower_goal.y;
    geometry_msgs::Point follower_res;
    Eigen::Quaterniond q(cos(leader_theta/2), 0, 0, sin(leader_theta/2));
    Eigen::Quaterniond q_normalized(q.w()/q.norm(), q.x()/q.norm(), q.y()/q.norm(), q.z()/q.norm());
    Eigen::Quaterniond v(0, temp_x, temp_y, 0);
    Eigen::Quaterniond v_new = q_normalized * v * q_normalized.inverse();
    follower_res.x = leader_point.x + v_new.x();
    follower_res.y = leader_point.y + v_new.y();
    return follower_res;
  }
```

Result of Video

Link : https://www.youtube.com/watch?v=Ktfvc9yTbzQ

Github : https://github.com/DavidHong1997/UAV_Final.git