Practical seminar task assignment: Leader - Follower

Multi-robot Systems (MRS) group at Czech Technical University in Prague

September 2020

MRS Summer School 2020

Leader-follower operation without communication

In this task, we will explore a state-of-the-art system for coordinated motion of multiple robots without direct communication. The system consists of two parts - blinking UV markers, which are fitted to each robot, and onboard cameras capable of UV-only sensing. Using this unique approach, the robots are capable of determining the relative position of their neighbors without the need to exchange any information via a communication link. This allows the UAVs to operate in environments with dense electromagnetic interference or coordinate motion of multiple robots with mutually incompatible localization systems (e.g. GPS and SLAM).

Installation

1) The installation requires the open-source MRS system, or if you prefer to install it separately, you will need the uav_core and simulation repositories.
2) You will also need to install the UVDAR core and the UVDAR plugin for the Gazebo simulator.
3) Further install the summer_school_supervisor, which will be enforcing the rules of the competition for this task.
4) Finally, you will need to install the trajectory_loader. This package will help you load a trajectory for the leader UAV, which will come in handy for testing in simulations.

Task overview

You are given a system consisting of two autonomous unmanned aerial vehicles (UAVs). Both vehicles are equipped with the MRS control pipeline and the UVDAR mutual localization system.

One of the UAVs is designated as a formation leader. It will follow a smooth, pre-defined trajectory. This UAV is out of your control and you should not interfere with the onboard control software.

The second UAV is designated as a follower. This is the vehicle, that you will be directly interacting with.

Your task is to develop a control routine for the follower such that it keeps up with the leader for as long as possible. We have prepared some C++ code to get you started. Your entry point to the system is the `src/follower.cpp` file. It creates a library, which is periodically called by the supervisor node to provide new control commands to the UAV. The file already contains the following methods:

- initialize - called only once on startup. This method is suitable for loading parameters from a configuration file. This way, you can tweak the parameters without needing to compile the package with every change. The default configuration file is `config/follower.yaml`.
- dynamicReconfigureCallback - use is optional, recommended for fine tuning of system parameters on the fly. Allows you to tweak system parameters while the program is running using the rqt_reconfigure. The default dynamic configuration file is `cfg/Follower.cfg`
- receiveOdometry - called every time a new odometry message is received. This will provide you with the latest information on the state of the follower UAV (position, orientation, velocity)
- receiveTrackerOutput - called every time a new message is received. This will provide you with the actual state of the UAV, which already compensates for external disturbances. Use this instead of odometry, if you want to control the UAV relative to its actual position.
- receiveUvdar - called every time a new UVDAR message is received. This will provide you the estimate of position of the leader UAV, based on the markers visible by the onboard cameras.

- createReferencePoint - called periodically by the supervisor node. You are expected to provide the onboard controller a new reference point through this method.
- createReferenceTrajectory - called periodically by the supervisor node. You are expected to provide the onboard controller a series of reference points through this method.
- createSpeedCommand - called periodically by the supervisor node. Alternative to the reference point commands. Allows you to have a deeper level of control by using velocity commands instead of position reference.
- getCurrentEstimate - uses a simple Kalman filter to estimate the position and velocity of the leader UAV. Filtering the raw UVDAR data will allow you to smooth out the control commands. Aggressive manoeuvres generate tilt, which may result in loss of visual contact with the leader.

These methods will be called by the `summer_school_supervisor` node running onboard the follower UAV. **Do not modify the supervisor node! The real UAVs will use the default one during the experiments!**

Simulation and a Reference solution

The provided code also includes a very crude solution. This solution will take the latest known leader pose, add a desired offset (loaded from a config file), and set it as the reference point for the follower. You may try running this code by opening the `tmux_scripts/two_drones_uvdar` folder and running the script:

```
./start.sh
```

Upon running the script, you should see 3 windows opening up: * Gazebo - realistic simulator with 3D graphics visualization * RVIZ - lightweight visualization tool for graphic representation of ROS messages * UVDAR simulation - two mostly black screens, which represent the view of the cameras onboard the follower UAV

How to confirm everything works

- In the Gazebo GUI, check that the time at the bottom of the window is running.
- After a few seconds, you should see two drones. The drones will take off automatically.
- After the takeoff, you should see markers in one of the UVDAR camera views. These represent the blinking LED lights on the leader drone.

Starting the task

You may notice, that your terminal opened multiple tabs. Consult the first page of the MRS Cheatsheet if you need help navigating the tabs and panes. **After the UAVs take off, your input has to be provided manually.**

- Navigate to the `goto_start` tab, hit up arrow key and then enter. There is a pre-filled series of commands, which will load the Leader trajectory and take both UAVs to their starting position.
- Switch to the `follower` tab, hit the up arrow and then enter. The pre-filled command will launch the `summer_school_supervisor`. This is the program, which will periodically call your code.
- Finally, switch to the `start_challenge` tab, hit the up arrow and pres enter. The leader will begin tracking the trajectory, and the score counter will start.
- Now, you can navigate back to the `follower` tab, and observe the simulation. Once the visual contact is broken, the process in the follower tab will print out the score.

Implementation tips

You may notice that the reference solution does not produce a smooth control input for the follower UAV. Also, the following may break after a while. The jumps in follower motion are caused by multiple contributing factors: * The camera as well as the target do not stay completely still, even during hovering. * Aggressive control manoeuvres will also move the camera more, and make it harder to estimate the leader position accurately. * Setting a single reference point to the controller will cause the UAV to decelerate and stop moving once the destination has been reached.

Let's improve the follower code

There are a few steps that may help you. It is not necessary to follow them. You may skip this section completely and craft a solution on your own.

- Experiment with the desired offset. You may get better results just by adjusting the distance between the leader and the follower. You can use the config/follower.yaml if you want to change the settings without the need for compilation. (You will still need to restart the node.)
- Experiment with the control action rate, also adjustable in the config file.
- Estimate the leader's velocity. An estimator based on the Kalman filter is already provided in the code, but in the default solution it is unused. Filter parameters may be loaded from the config file as well.
- You can give the follower three types of control commands: ReferencePoint (this is done in the default solution), ReferenceTrajectory and SpeedCommand. Inside each command, there is a boolean variable called `use_for_control`. If you set this value to `true`, the supervisor node will attempt to use this command for control.
- If multiple commands are available at the same time, only the highest ranking command in the following hierarchy will be used: SpeedCommand > ReferenceTrajectory > ReferencePoint.
- In computer vision, distance to an object tends to be more difficult to estimate than its bearing. Therefore, the position tracking works much better if the line of sight from camera is directly perpendicular to the direction of leader's motion. If the leader moves along the line of sight from the camera, retrieving the position becomes much more difficult.
- You are not required to perfectly copy the leader's trajectory. You only have to track the leader for as long as possible.

Things to avoid

- Collisions (wow!) - When using the ReferencePoint or ReferenceTrajectory control, the UAVs will automatically avoid collisions. The avoidance is done by moving the follower into a higher altitude. This mechanism will be triggered once the distance between the vehicles is less than 3 m. During this time, all of your commands will be overriden. Triggering the collision avoidance will not result in a score penalty, however, the leader may evade you in the meantime. If you are using the SpeedCommand for control and get within 3 m of the leader, the follower fill fallback to the MPC control and perform the avoidance manoeuver as well.
- Height changes. The leader will always fly at a height of 3 m. Control commands with height below 2 m and above 4 m will be discarded by the follower.
- Erratic position changes. Position reference, which is over 15 m apart from the current UAV position will be discarded.
- Pushing physical limits of the UAV. Velocity command larger than 5 m/s will be discarded.
- Violating these restrictions 10 times in a row will terminate the challenge.