A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Development and testing of methods for drones control

Paolo Leopardi

Last update: October 17, 2023

Contents

Acronyms

CPP Coverage Path Planning

GCS Ground Control Station

QGC QGroundControl

ROS Robot Operating System

SITL Software-In-The-Loop

UAV Unmanned Aerial Vehicle

1 Introduction

The project is officially called (italian) *Sviluppo e sperimentazione di metodologie di controllo per droni* and it lies in the field of agricultural robots. The aim of these months will be the implementation of a real drone which is able to move in a area and take informations autonomously through some device mounted on the robot (thermal imager). The project can be splitted in two main parallel directions:

- Navigation (path planner, control schemes, etc.)
- Implementation (drone construction, autopilot selection, etc.)

2 Navigation

To successfully cover the area of interest the quadcopter has to navigate following a certain logic and take into account different factors such as the shape and the dimension of the area, presence of obstacles, vehicle used and so on. There is a class of algorithm called Coverage Path Planning (CPP) which is well suited for this aim.

2.1 Coverage Path Planning

Given an area of interest the CPP problem consist of planning a path which covers the entire target environment considering the vehicle's motion restriction and sensor's characteristics, while avoiding passing over obstacles [1]. These algorithms can be classified into two main categories: offline and online [2]. Offline algorithms need a previous knowledge of the search area, online algorithm instead are based on real-time data acquisition.

Another classification is based on the decomposition method thaat can be classified in:

- Cellular decomposition
  - Exact decomposition
  - Approximate decomposition
- No decomposition

Cellular decomposition methods are based in dividing the surface into cells: in the exact decomposition the workspace is splitted in sub-areas whose re-union exactly occuped the target area [1]. In the approximate decomposition the area is usally divided using a grid where the size of the squares is typically determined for example by the footprint of the camera mounted on the robot. In no decomposition techniques, as the name suggest, isn't applied any type of decomposition. Taking in to account the aim of the project, i.e. the Unmanned Aerial Vehicle (UAV) has to collect data in different positions of an area, the best solution is the approximate decomposition technique beacuse we don't need to cover every centimetre of the area (like an autonomous lawn mower), we need to determine the amount of waypoints that guarantees an exhaustive data collection compared to the target area.

3 Implementation

The implementation step is the physical construction of the UAV, this involves the selection of all the elements, both hardware (e.g. platform and its components) and software (e.g. autopilot flight stack).

3.1 Autopilot selection

Autopilot selection is made by evaluating possible pros and cons which every autopilot fligh stack brings with it. Three possible solution were evaluated:

1. INAV [3]
2. PX4 [4]
3. Agilicious [5]

There are a lot of reason which can determine the choice of one solution instead of another, a preliminary evaluation is made considering the informations available on the web (official documentation and other sources). These parameters have been accounted:

- configuration
- missions definition
- future developments

Configuration denotes the level of complexity needed to configure flight controller for the first flight, *missions definition* takes into account how to define missions, and *future development* indicates compatibility with other framework, software and so on.

INAV's configuration seems easy as PX4, the main difference is the guide: for INAV you can follow some videos on Youtube at this link, for PX4 it's necessary to follow sections from *Basic Assembly* to *Flying* in the official documentation. Agilicious doesn't have a section related to the configuration steps for the first real flight like the above mentioned.

INAV provide a Ground Control Station (GCS) which is capable of define only waypoints which the UAV has to visit, as shown for example here. PX4 typically use QGroundControl (QGC) as GCS[1], here different missions can be defined and it is worth to note that there is also survey missions which seems particularly suited with the aim of this project. Agilicious doesn't not provide a GCS for missions definition, but it has a module called `reference` which implements different ways of generating reference trajectories.

I wasn't able to find any documentation regarding interfacing between INAV and Robot Operating System (ROS), PX4 has a subsection dedicated to ROS communication with PX4. In addiction PX4 has a MATLAB package called UAV Toolbox Support Package for PX4 Autopilots [6]. Agilicious has a very good structure for future developments beacause you can change controller or estimator by simply modify a `yaml` file. It's not provided a way to integrate GPS measurements. An interface for ROS called `agiros` is provided. Both PX4 and Agilicious docs propose a simulator.

In conclusion the better idea should be to try the autopilot in this order: PX4, Agilicious, INAV.

[1] QGC supports only PX4 and Ardupilot

3.2 PX4 configuration

Before first flight PX4 Autopilot needs some steps to follow to configure the autopilot, this one are documented in PX4 documentation's section called Standard Configuration. The procedure is quite straightforward but some problems may arise during these steps.

Troubleshooting

Firmware version

QGC provides au automatic way to flash the latest firmware[2], however all version 13 express same problem with our specific hardware. More specifically the problem is related to the Wi-Fi module because with the firmware version `v1.13.x` the autopilot is unable to connect with QGC. So I found that version `v1.12.3`[3] fixes this problem.

Autotune

Having downgraded to the version `v1.12.3` determined the impossibility to use the autotune procedure because this is available from `v1.13.0`.

3.3 Optitrack configuration

After some outside experiments (in which human pilot successfully drove the quadcopter) we decided to take the next flight test in an indoor scenario; this beacause an indoor environment is safer if compared to the outdoor one in terms of damaged caused by the drone's crashing.

Before flying, the communication between Optitrack and flight controller needs to be configured, we can think the Optritrack as the indoor counterpart of the GPS. To configure the Optritrack with PX4 there also a dedicated section named Using Vision or Motion Capture Systems for Position Estimation, this one provides all the necessary steps to configure the communication. Please note that there is also a dedicated subsection for Optitrack system.

Troubleshooting

Parameters

Having used an older firmware version, some parameters[4] have been replaced with others; these ones are listed in the table below. The first column shows the actual name of the parameters the second columns shows the counterpart on the firmware version used in this project.

PX4 docs naming	`v1.12.3` naming
`EKF2_EV_CTRL`	`EKF2_AID_MASK`
`EKF2_HGT_REF`	`EKF2_HGT_MODE`
`EKF2_GPS_CTRL`	`EKF2_AID_MASK`

Another set of parameters are the ones used for the preflight check, Disabling these prevents the drone from checking the correct operation of the corresponding sensors:

- `SYS_HAS_BARO`

[2] At the time of writing this report, i.e. September 2023, the last stable release is `v1.13.3`.
[3] Firmware releases available here.
[4] Full parameter list here.

- SYS_HAS_GPS
- SYS_HAS_MAG

4 Simulation

To test some CPP algorithms is worth to implement an environment which allows to drive a drone autonomously in a safe way, without any risk of collision. PX4 offers different simulators which allow to develop Software-In-The-Loop (SITL) simulation [7]. More in details, in the section named *MAVROS Offboard control example (Python)* [8] there is a useful example on how to setup PX4, Gazebo and MAVROS to run a simulation.

To implement a complete pipeline which allows to develop CPP algorithms, simulate the quadcopter and analyse the result MATLAB has been employed beside the simulator structure which exploit PX4, Gazebo and MAVROS, mentioned above. MATLAB is used to determine the area, develop the CPP algorithms and pass the waypoint to the simulator. After the simulation phase, which is executed in Gazebo environment the results are visualized and analysed in MATLAB again.

4.1 MATLAB code

```
%% create target area

basemap = "satellite";
geobasemap(basemap)
geoplot(43.116118, 12.525492)
geolimits([43.11, 43.12], [12.51, 12.53]);
set(gcf, 'WindowState', 'maximized');

target_area = drawpolygon("Color", 'y') % 1 column = latitude, 2 column = longitude

target_area = polyshape(target_area.Position(:,2), target_area.Position(:,1))



figure
plot(target_area)
title('Target area')
xlabel('Longitude')
ylabel('Latitude')
grid on


target_area_verticesMeters = latLonToMeters(target_area);
target_area_meters = polyshape(target_area_verticesMeters(:,2), target_area_verticesMeters(:,1));

figure
plot(target_area_meters)
title('Target area (local coordinates in meters)')
xlabel('X (m)')
ylabel('Y (m)')
grid on
```

References

[1] Tauã M Cabreira, Lisane B Brisolara, and Ferreira Jr Paulo R. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1):4, 2019.

[2] Georgios Fevgas, Thomas Lagkas, Vasileios Argyriou, and Panagiotis Sarigiannidis. Coverage path planning methods focusing on energy efficient and cooperative strategies for unmanned aerial vehicles. *Sensors*, 22(3):1235, 2022.

[3] iNavFlight. iNav. Available at: https://github.com/iNavFlight/inav, 2023. Accessed: 12 July 2023.

[4] PX4 Autopilot Development Team. PX4 Autopilot. Available at: https://px4.io/, 2023. Accessed: 12 July 2023.

[5] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, et al. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science robotics*, 7(67):eabl6259, 2022.

[6] MathWorks. PX4 Support Package. Available at: https://it.mathworks.com/help/supportpkg/px4/, 2023. Accessed: 12 July 2023.

[7] PX4 Autopilot Development Team. PX4 Documentation: Simulation. Available at: https://docs.px4.io/main/en/simulation/, 2023. Accessed: 17 October 2023.

[8] PX4 Autopilot Development Team. PX4 Documentation: MAVROS Offboard Python Example. Available at: https://docs.px4.io/main/en/ros/mavros_offboard_python.html, 2023. Accessed: 17 October 2023.