Motion planning of a fixed wings Uav through an hybrid approach based on artificial potential fields and RRT.

Edoardo Ghini

ghiniedoardo@gmail.com

Gianluca Cerilli

gianlucer@gmail.com
Dipartimento di Ingegneria dell'Universita' di Roma La Sapienza

May 31, 2018

I. Introduction

This report presents a work made for an international challenge[1] that every year involves several academic teams.
The following structure has been respected :
in section II, we will present the problem statement and in section III we will give technical informations on the hardware at our disposal, finally in section IV, the adopted solution will be discussed.

II. Problem Statement

All the joining teams to the challenge will have to compete on several tasks concerning on actual problems in the governance of Unmanned Aerial Vehicles.
Each team will bring its prototype of UAV on a common flight ground and try to score the greatest number of points among all the tasks proposed which are:

1. Autonomous Flight
2. Obstacle Avoidance
3. Object Detection
4. Object Classification
5. Object Localization
6. Air Delivery

This will be the first participation at a competition of this kind for the team Sapienza, in fact the previous challenges in which our university has already been involved required a human controlled guidance system. Therefore the autonomous flight constraint has to be faced without previous insights on the matter.

i. Starting Points

The aeronautical research department, in the past decade, has spent a big effort working to an autonomous guidance system, that has been designed and implemented by master students in their thesis.
This system allows to control the complex aerodynamics of the vehicle from an higher level of abstraction, its main high level control mode uses a series of way points that describes a trajectory.

In this way the aircraft will likely follow a path made of lines that intersects subsequent waypoints.
The system is designed to work on an on-board computer and to communicate with a ground station that will continuously send to and receive from the judges's server, telemetrical data and mission objectives.
We will use a fixed-wigs radio controlled aircraft model[2] commonly used in hobby modelling as shown in Figure 1.

Figure 1: *YAK scaled aero model*

[1] The challenge is the AUVSI-SUAS hosted in united states in summer 2018.
[2] It is a scaled reproduction (1:3 ratio) of the YAK 112

ii. Our task

This paper describes the work concerning the area of **automation** and **robotics** of an atomic part of the full Sapienza Flight team, which counts several members coming from an aeronautical background.
With this premises, the autonomous guidance problem that comprehends also the obstacle avoidance clause, has been engaged.
The competition will be organised in such a way that during the various missions of the challenge each team will receive pose informations about fixed and mobile obstacles that will be **virtual**.
Then the judges will check the success or failure in avoiding obstacles according to the **telemetry** data that they will receive from each team.
In other words, we where supposed to design a **path planning** algorithm that will bring the UAV from a start position to a goal position without collisions.
Additionally we contributed to the integration of the ground control station with the interoperability protocols that allows an efficient communication with the judge's server, both for receiving mission information, obstacle position and sending the telemetry of the aircraft.

III. Hardware components

All the components described in this section, left aside the propulsive module, will work with two different voltages (around 3 and 5 Volts). For this reason two converters have been adopted, in particular the *Tracopower TSR 1-2433* and the *Recom R-78B5.0-1.5*.

i. Propulsion

The engine chosen is a **DLE55** (in Figure 2).

1. 50 cc monocylinder petrol engine
2. reservoir capable of containing 950cc of petrol
3. Equipped with a 7, 4V LiPo battery

Figure 2: *DLE55 motor*

ii. On board computing

The autonomous guidance architecture, implemented in Matlab and Simulink, is converted in compiled high speed C, Cpp language.
It will be run by Arduino Due and there will be also a Raspberry pi Model 3 that will run a Kalman filter in order to process all signals from sensors.

iii. Sensing instruments

Concerning the sensors which are mounted on the aeromodel we have a GPS receiver, namely *Lassen IQ* and an attitude and heading reference system (*MTi* model).
We also find a pressure and temperature sensor which is a *BMP280* and an interestingly complex system to measure the fluid flow velocity (in this case the fluid measured is the air around the plane body) that is the *Honeywell SSC*.

iv. Communication devices

Ensure secure and continuous communication between the aeromodel and the ground station is a crucial point, in fact we employed two capable antennas, the *Xbee Pro S2C*. They will be placed one on the lower body of the aircraft and the other on the ground.
These reliable antennas works at 2.4 GHz and are omnidirectional, they support an operational range of 1.5 Km and in our case, some estimations made clear to use a 96 dB gain in the antenna on the ground.

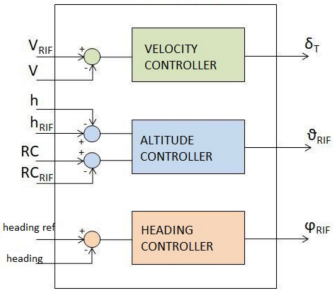

Figure 3: *External loop*

v. Auto pilot framework

The system supports six modes :

- Boot and pre-flight checklist
- Calibration of sensor
- Ready before mission start
- Manual guidance
- Stand by (transition state)
- Autonomous guidance

Talking about the last mode, there are two main approaches supported by the **Auto Pilot** : *Vectorial* guidance and *Waypoint-based* navigation.

The vectorial guidance is controllable through four desired values:

- **ALT**: altitude
- **RC**: rate of climb (roll angle ϕ)
- **IAS**: velocity
- **HDG**: heading (yaw angle ψ)

An oversimplified conceptual map is shown in Figure 3 Figure 4.

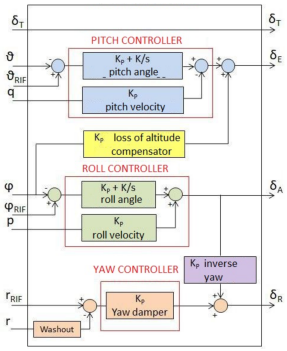

Figure 4: *Internal loop*

iv

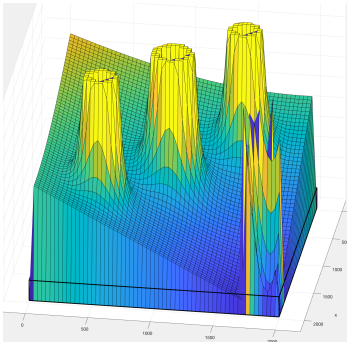

Figure 6: *Example of the potentials distribution on a map with three obstacles and the goal on the lower right corner*

IV. Hybrid planner

We implemented two different strategies, since both present some shortcomings and advantages, we tried to take the best of the two worlds.

i. RRT

RRT (Rapidly-exploring Random Tree) is a probabilistic planner that randomly builds a space-building tree.
It has been used offline considering some motion primitives (produced by some specific velocity inputs) to define a path for the UAV, biasing it towards the unexplored areas closest to the goal. One interesting aspect of this probabilistic approach is that it will eventually find a way to reach the goal if it exists. In Figure 5 we can see how the algorithm explores all the branches of the tree and expanding unexplored nodes and discarding colliding ones it will succeed to find the path between the two obstacles.

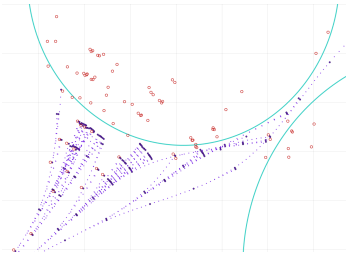

Figure 5: *Behaviour of RRT search near obstacles*

ii. Artificial potentials

Artificial Potential Fields (APF) have been considered for two main reasons:

- the world can be approximated to a "world of spheres" (since there are cylindrical obstacles that we consider in 2D), that prevents the UAV entering the basin of attraction of some local minima
- when the UAV faces an obstacle, RRT must expand many branches before getting around it. This leads to a significant slowdown

Using APF the UAV can react as soon as entering the range of influence of the obstacle, with a smooth movement. This result has been made possible thanks to the implementation of vortex fields, replacing repulsive actions with actions forcing the robot to get around each obstacles, in practice generating a tangential gradient that follows a circle around the obstacle that will guide the aircraft avoiding harmful gradients pointing against the natural direction of motion as shown in Figure 7.

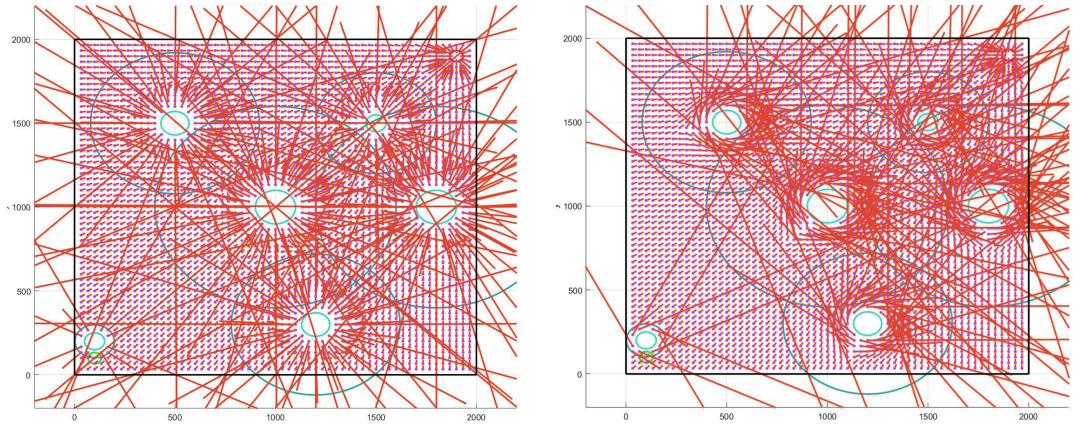

Figure 7: *Map with vectors pointing in the gradient of the artificial potential fields: the difference between the two images is that in the right the vortex was implemented while in the left wasn't.*

iii. Implementation

We consider the kinematic model of a fixed-wing UAV flying at a constant altitude. So, we can neglect the pitch angle, obtaining:

$$\dot{x} = u_v cos\theta\psi$$

$$\dot{y} = u_v sin\theta\psi$$

$$\dot{\psi} = -\frac{g}{u_v} tan\phi$$

$$\dot{\phi} = u_\phi$$

in which the tangential velocity u_v and roll rate u_ϕ are the control inputs. As described above, for the path planning we use a mixed approach between RRT and Artificial Potential Fields methods.

At the beginning, a random value is generated: if this value is greater or less than a factor (that is decremented after each cycle), the RRT generates respectively a position biased to the goal or a random one.
Depending from the five primitives of the UAV and it expands a node that represents the closest possible position of the UAV to the new configuration.
If this new position collides with an obstacle, the node is not added to the tree and becomes unavailable for next nodes generation.

In a similar way, if all the children of a parent collide with some obstacle, both the children and the parent become unavailable. These five primitives are generated and ordered according to another factor obtained from the APF method.

Particularly, depending from the configuration of the UAV and from the range of influence of the obstacles, the total vortex field acting on the UAV is computed.
Since the resultant vector of the vortex fields is directed toward the goal, we compute the direction

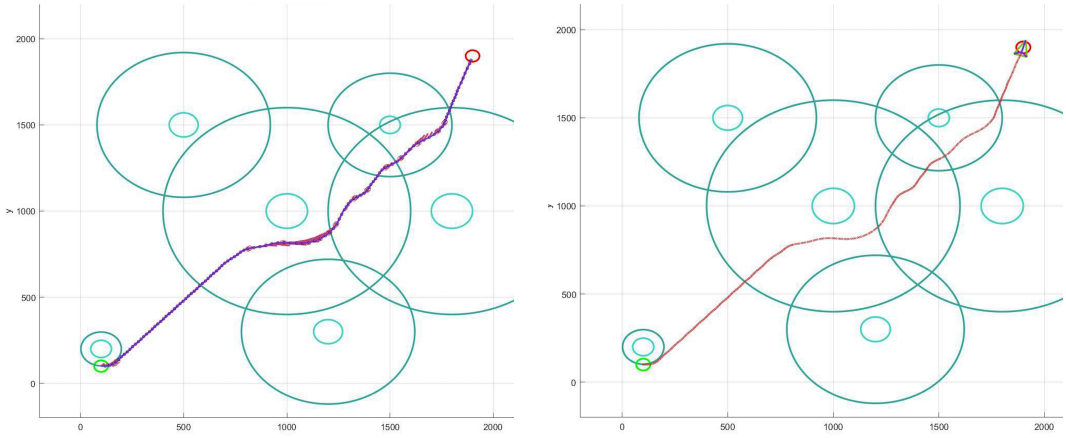

Figure 8: *On the left is shown the dynamic execution of the planning algorithm, on the right there is the path followed by the UAV after that the planning has been completed.*

of each primitive and select the one that forms the smallest angles with the direction of the vortex. Thanks to the vortex fields the UAV get around the obstacles avoiding local minima and proceeds faster to the goal.

V. Interoperability integration

In order to achieve an efficient communication we where nearly obliged to make use of the python **API** written by the challenge organisers. In this way the communication can be handled with *asynchronous* **HTTP** requests between each team client and the judge's server.
Since our team has chosen a code base in MATLAB, we implemented an interface that allows to invoke the API functions directly from MATLAB code.
Then every significative data received will be saved in a compatible file (in our case in a .mat file) that will be read and used in the execution cycle of the Auto Pilot infrastructure.

VI. Conclusion

In the end, we consider the experience that we have made a good way to face problems spread on a large spectrum.
In retrospection, it was a very time-taxing challenge and in our opinion it demands a lot more time and effort from the team to be able to confront the other teams around the world.

In such a way they already come from past years experiences and results, and they alse have an accumulated knowledge about the issues that we, instead, have had to solve for the first time. We hope that this work could be a base for the next challenges of our university and that the competition in June will score good results.