LAMBARKI EL ALLIOUI, Yassine

TPS RIO 2022

Engineering Degree

Telecom Physique Strasbourg

Second Year Internship Report

“Smooth path planning in constrained environments for unmanned aerial vehicles”

HERRERO Jesus Garcia jgherrer@inf.uc3m.es

Llerena Juan Pedro jllerena@inf.uc3m.es

01/06/2021 - 31/08/2021

uc3m | Universidad Carlos III de Madrid

Acknowledgements

First of all, I am very thankful to the GIAA team who welcomed me in the best way, it was a real pleasure to discuss interesting subjects with you and exchange ideas about drone problematics. Second, I would like to express my gratitude to both Lllerena Juan Pedro and Mr. HERRERO Jesus Garcia, my internship supervisors, for their guidance, encouragement and positive critics during my internship. Thanks to your monitoring, I could explore many areas in the UAVs field and I have deepened my methodologies and techniques to tackle many drone problems and specially in path planning.

I would like to extend my thanks to the incredible interns whose company was a blessing this three month internship. It was a joy to work with you and discuss each other's bugs and difficulties.

Finally, I wish to thank my parents for their support and encouragement throughout these years.

Abstract

The use of UAVs is becoming common in many fields with less human interaction. Thus, it is important for autonomous UAVs to find the optimal path planning methods to move from the location to the target. To assure the right functioning, such methods require a map or a graph of the mission environments so that the UAV can be aware of the constraints and its location in the map. The aim of a path planning method is not only to find an optimal path but also to provide a safe collision-free trajectory. In this project, we will explore methods to detect obstacles like houses and avoid those no fly-zones while providing a safe optimal route from source to destination. This paper explores methods to detect no fly-zones like houses and provides path planning techniques like Artificial Potential Field, A star, Dijkstra, and Dynamic Programming method. We will also compare the cost of each method to elect the best one. In a second time, the trajectories will be simulated in AirSim to study the execution of the path planning methods in a simulated environment. Furthermore, we will discuss some problems like solving the minimum local potential field or smoothing trajectories. In addition, a number of open research problems faced in this project will be explored to provide deep insights to the readers.

Résumé

L'utilisation de drones devient courante dans les domaines qui demandent moins d'interactions humaines. Ainsi, il est important pour les drones autonomes de trouver des méthodes de planification de trajectoire optimale pour se déplacer de l'emplacement à la cible. Pour assurer un bon fonctionnement, de telles méthodes nécessitent une carte des environnements de mission afin que l'UAV puisse connaître son emplacement et les contraintes sur la carte. L'objectif d'une méthode de planification de trajectoire n'est pas seulement de trouver une trajectoire optimale, mais également de fournir une trajectoire sûre et sans collision. Dans ce projet, nous explorerons des méthodes pour détecter les obstacles comme les maisons et éviter ces zones d'exclusion aérienne tout en fournissant un itinéraire optimal et sûr de la source à la destination. Cet article explore des méthodes pour détecter les zones d'interdiction de vol comme les maisons et fournit des techniques de planification de chemin comme : le champ de potentiel artificiel, A star, Dijkstra, et la programmation dynamique. Nous comparerons également le coût de chaque méthode pour élire la meilleure. Dans un second temps, les trajectoires seront simulées dans AirSim pour étudier l'exécution des méthodes de planification de trajectoire dans un environnement simulé. De plus, nous aborderons certains problèmes comme le minimum local ou le lissage de trajectoire. En outre, un certain nombre de problèmes de recherche ouverts rencontrés dans ce projet seront explorés pour fournir des informations approfondies aux lecteurs.

Contents

Table of figures

List of abbreviations

UC3M : Universidad Carlos III de Madrid

GIAA : Applied Artificial Intelligence Group

UAV : Unmanned Aerial Vehicle

QGC : QGroundControl

UE4 : Unreal Engine 4

A* : A star

DP : Dynamic Programming

DL : Deep Learning

1. Overview of Universidad Carlos III de Madrid

1.1 Presentation of the university

Universidad Carlos III de Madrid (UC3M) was established by an Act of the Spanish Parliament on the fifth of May 1989, within the framework of the University Reform Act of 1983. From the outset it was intended to be a relatively small and innovative public university, providing teaching of the highest quality and focusing primarily on research. UC3M's mission is to contribute to the improvement of society through teaching of the highest quality and cutting-edge research in line with stringent international guidelines. The University aspires to excellence in all its activities, with the aim of becoming one of the top universities in Europe. The university actively encourages the personal development of all those connected to the higher education community. All their activities are guided by the values of merit, ability, efficiency, transparency, fairness, equality and respect for the environment.

Figure 1: Universidad Carlos III de Madrid

Figure 2: Campus of Colmenarejo

1.2 Presentation of the GIAA laboratory

The Applied Artificial Intelligence Group, GIAA, is composed of a team of physicists, telecommunications and prestigious computer engineers, recognized for their nationally recognized for their ability to solve engineering problems by integrating the latest artificial intelligence techniques: machine learning, evolutionary learning, evolutionary computing, data analysis, objective optimization, fuzzy systems and intelligent agents.

The group has a long history of work providing consulting to companies and developing countries customized solutions for prediction, optimization, data fusion and signal and image processing. Their research activities and lines are:

- Automatic Learning Techniques.
- Data and Business Analysis.
- Evolutionary Computation and Multi-objective Optimization.
- Agents and Multi-agent Systems: web, information retrieval, electronic commerce, sensor management.

- Computer Vision.
- Internet of Things (IoT).
- Data Fusion Systems and Contextual Information.
- Surveillance Systems.
- Air Traffic Control (ATC).
- Coastal Surveillance and Maritime Traffic.
- Indoor localization systems.
- Inference in adaptive, non-linear dynamic systems.
- Unmanned vehicles.
- People detection, tracking and action recognition.
- Behavior extraction through data analysis.

1.2.1 Equipment

The investment in the provision of infrastructure for the research group amounts to approximately 750 000 €. The research laboratory has been equipped with high performance computer systems, cameras, and communication network, etc. The wireless network system is formed by an Aruba WiFi tracking system and by an Ubisense Ultra Wide Band network. Both systems are portable and can be used in any test environment of about 400 m^2.

2. Presentation of the project

2.1 Introduction and motivation

Drones or UAVs are unmanned aircraft piloted automatically or remotely, for civilian use or for the benefit of the armed or security forces - police, customs - of a State. Depending on the required capacities, their mass varies from a few grams to several tonnes. Their autonomy can reach up to several dozen hours (compared to the typical two hours of autonomy for a hunter).

Advances in IT and technology have caused some drones to be new-world weapons. They are also used for intelligence and electronic warfare (including jamming or interception of communications). Their missions are then the ISR (Intelligence, surveillance, and reconnaissance) or ISTAR (for “Intelligence, Surveillance, Target”). Acquisition and Recognition ”).

Their civilian applications include traffic control, maritime and environmental surveillance, air search and rescue operations, the collection of data of meteorological interest or in difficult environments (in CBRN risk zones - “nuclear, radiological, bacteriological and chemical ”- for example), relaying information, taking aerial photographs and even soon the direct acquisition of photogrammetric data...

With the extent of use of drones and UAVs in recent years, the regulations are becoming more strict. Thus, for privacy reasons, it is important for UAVs to avoid NFZ (no-fly-zones) during their path planning. The NFZ could be airports, military bases, nuclear facilities, and private properties. All of these can be considered as constraints that UAVs should avoid while flying in a specific area.

According to all of the above, the implementation of a path planning method with obstacle avoidance is necessary to assure an optimal trajectory and a legal flight for UAVs.

2.2 Problematic

To create such a system it is important to first detect the constraints and obstacles. In this study, we will only take into consideration buildings and private properties as obstacles. But the methodology can be generalized and used for other types of obstacles.

First, how can we detect buildings, or more precisely, how can we detect buildings' footprints on a map? How can we situate those obstacles geographically so that the drone could avoid them?

What path planning method can be used for this type of obstacle? To what extent can it be accurate and efficient?

And finally, is the trajectory generated realistic? Can we fly the drones using that trajectory in a simulated environment to discuss its accuracy ?

3. Hardware and Software overview

3.1 UAV or Drone

An **Unmanned Aerial Vehicle** (UAV) is defined as a "powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely. The first drones were created for military use during the first half of the 20th century, but as they have evolved their uses have multiplied and stood out in multiple areas such as Topography and mapping, Agriculture, Security and emergency situations, and Logistics. Within the UAVs, we can find different types depending on the type of wings they have. They are divided into two main types: Fixed-wing and Rotary wing.

Fig 3: Fixed wing and Rotary wing drones

Drones offer a reliable means of representing rotations in 3-dimensional pitch, roll, and yaw. From these main rotations and the horizontal displacement, in the case of fixed-wing UAVs, and the vertical displacement, in the case of rotary-wing drones, all

the movements that the aircraft can perform are determined. Pitch, roll, and yaw (rotations in the transverse axis, longitudinal axis, and normal axis, respectively) are visualized in Figure 4.

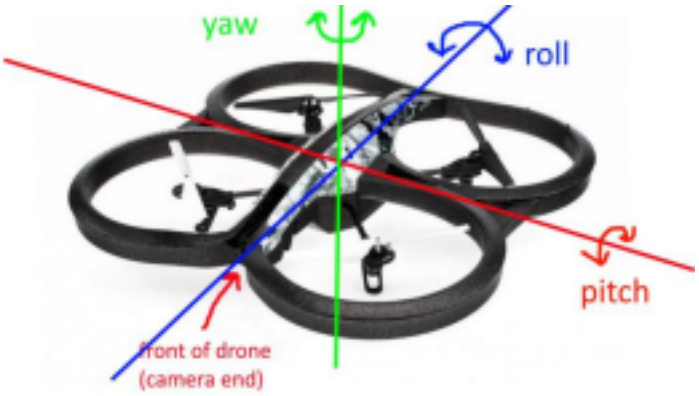


Fig 4: Visualisation of pitch, roll and yaw.

Drones can be equipped with a lot of materials, especially sensors, from Radars and Sonars to Lidars and Cameras that facilitate navigation. The processing of the images by these cameras allows the development of a multitude of applications using different artificial vision techniques in a profitable way since the cameras are cheap compared to other types of sensors.

Newer computer vision applications that run nearly to completely on the drones themselves are promising in opening new markets for drone manufacturers as well as their hardware and software suppliers. Such applications further expand the

capabilities of drones to include real-time availability of results, enabling faster decision-making by users.

One of the key functions to these real-time applications for drones is **self-navigation**. Currently, most drones are flown manually, and battery capacity limits flight time to around 30 to 40 minutes. Vision-based navigation, already offered in trendsetting consumer drones, conversely enables them to chart their own course from point of origin to destination. Such a drone could avoid obstacles, such as buildings and trees, as well as more generally be capable of calculating the most efficient route.

3.2 Software

a. Unreal Engine 4

Released in May 1998, Unreal Engine is a free-to-use game development engine owned by Epic Games. It can create a large variety of 3D, 2D, and VR game styles. The platform is known for its impressive graphical and lighting capabilities. It also includes a library of materials, objects, and characters. In addition to supported coding languages, Unreal Engine offers a visual editor, called *Blueprints*, for creating the rules of the game, which requires no coding experience. The Unreal Engine 4 (UE4) editor will allow users to simulate a customized environment in which we can visualize a drone interacting with the physical laws of the real world in a way that achieves a realistic and very accurate simulation. The main advantage of UE4 over other simulators that could have been used such as Gazebo, jMavSim, or FlightGear is that it achieves greater realism in photography, which will be very useful when interacting with visual sensors, which is what is of most interest for the task to be developed in this project.

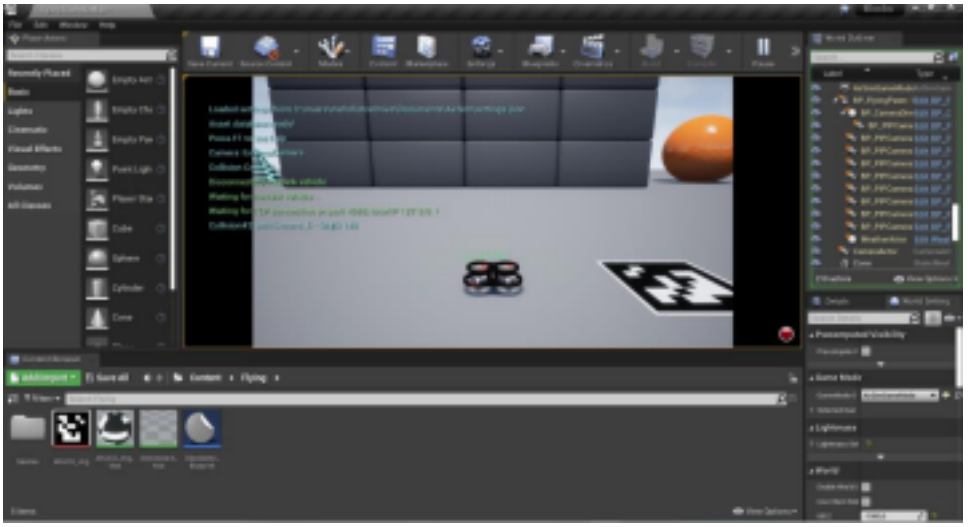


Figure 5: Drone simulation with UE4.

● **Installation:**

First, we must download the installer from the official site of Unreal Engine, after that we will have to create an Epic Games account to link it to Github. Once done these steps, we will have to execute the installer and finally launch the platform.

b. Airsim

AirSim (Aerial Informatics and Robotics Simulation) is an open-source, cross platform simulator for drones, ground vehicles such as cars and more, built on Epic Games's Unreal Engine 4 as a platform for AI research as well as Unity. It supports software-in-the-loop simulation with popular flight controllers such as PX4 & ArduPilot and hardware-in-loop with PX4 for physically and visually realistic simulations. It is developed as an Unreal plugin that can simply be dropped into any Unreal environment. Similarly, we have an experimental release for a Unity plugin. AirSim exposes APIs so you can interact with the vehicle in the simulation programmatically. You can utilize these APIs to recover images, get state, control the vehicle, etc. The APIs are exposed through the RPC, and are accessible via a variety of languages,

including C++, Python, C# and Java. These APIs are also available as part of a separate, independent cross-platform library, so you can deploy them on a companion computer on your vehicle. This way you can write and test your code in the simulator, and later execute it on the real vehicles. To obtain a model as close as possible to reality of the physical phenomena affecting the vehicle AirSim focuses on the following three points: Gravity, Magnetic field and Air pressure and density.

- **Installation:**

To install Airsim, we must follow these instructions:

1. Create a folder that we called “Workspace”.
2. Open the bash git and clone https://github.com/Microsoft/AirSim.git and then we execute the command to download the repository.
3. Open as Administrator the program "x64 Native Tools Command Prompt for VS 2019" (I worked on Windows, and instead of terminal we have the command prompt).
4. In the terminal that opens, execute cd C:\Users\USERNAME\Desktop \Workspace \AirSim
5. build.cmd
6. update_from_git.bat

Normally, after opening UE4 and in Settings → Plugins, we should see Airsim active. **c. PX4**

PX4 is an open-source flight control software for drones and other unmanned vehicles. The project provides a flexible set of tools for drone developers to share technologies and innovations to create tailored and customized solutions for drone applications. PX4 is hosted by Dronecode, a Linux Foundation non-benefit. All hardware and software is open-source and freely available to anyone under a BSD license and users can modify the autopilot based on their own special requirements. In this project PX4 will interact with AirSim to simulate the drone, it will act as if it were

the "brain" of the drone, in order to carry out the process. From the PX4 console, it will be possible to see information and send orders to carry out the drone.

- **Installation:**

The installation of PX4 on Windows can be done following these steps:

1. Download the Pixhawk PX4 toolchain for Windows 10, version 0.8
https://github.com/PX4/windowstoolchain/releases/tag/v0.8
2. After running it and executing the installation, Go to the folder C:PX4 and create a folder called "home".
3. In this folder, we must run the command run-console.bat
4. A Linux terminal based on cygwin64 will open, there we will execute git clone https://github.com/PX4/Firmware.git
5. cd Firmware/
6. git checkout v1.10.1
7. make px4_sitl_default none_iris
8. Compile the first time and then use it to start PX4.

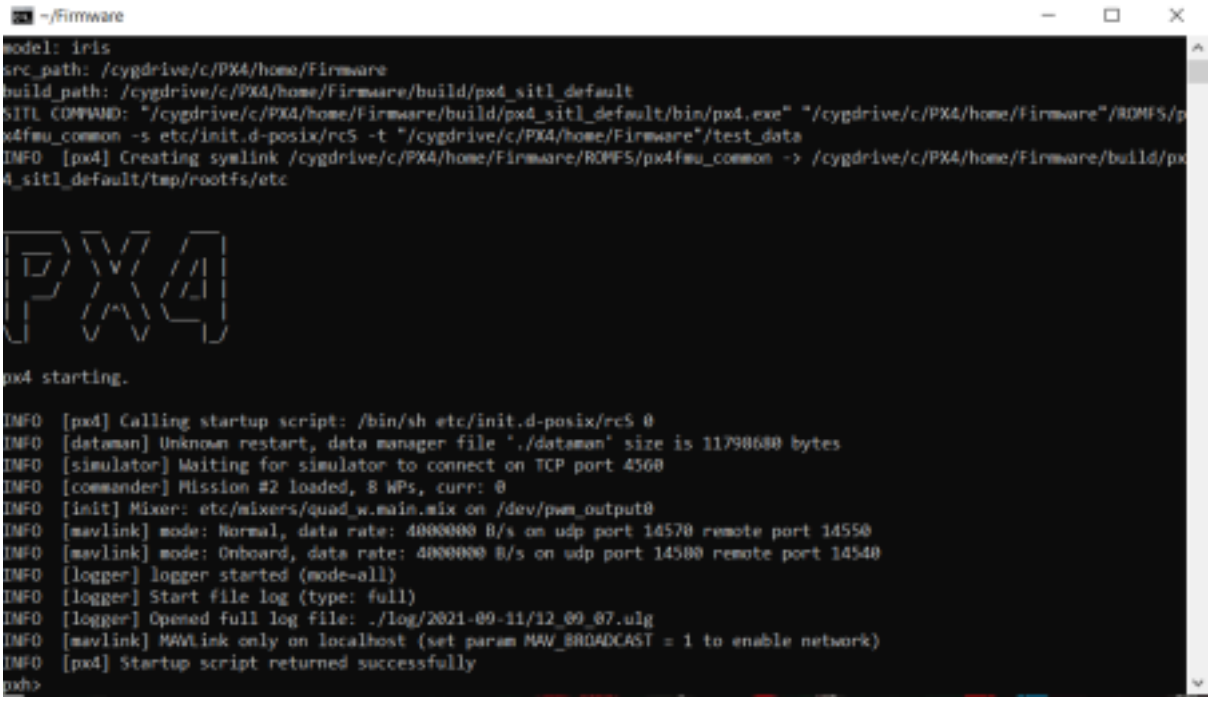


Figure 6: PX4 launching

d. QGroundControl

QGroundControl is a mission planner that provides full flight control and tuning as well as mission planning for any MAVLink enabled drone. Its primary goal is ease of use for professional users and developers. All the code is open-source source, so you can contribute and evolve it as you want. It permits planning autonomous missions, with courses that include going through a series of given points or overflying an area describing the desired path. It displays in real time the position, speed, orientation, flight time, battery status and satellites to which the drone is connected.

QGroundControl will connect with PX4 and AirSim forming a software-in-the-loop system in which through the mission planned in QGC and the drone simulated in PX4, the behavior of the drone can be seen in the AirSim window in Unreal Engine 4. QGC will act as the ground station of the drone.

● **Installation:**

The installation of QGroundControl is simple, we have to download the installer from: https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html and then run it with allowing access if the Windows Defender Firewall appears.

Figure 7: QGroundControl Interface

4. Proposed solution

4.1 Building detection

Problematic: Detecting houses from aerial satellite images

4.1.1 Line detection

In brief, here we will present a building detection system from single monocular satellite imagery. Each input is segmented at several levels. The borderline definition of such fragments combined with line sections identified on the initial picture is utilized to create a set of quadrilateral building hypotheses. For each hypothesis, a probability score is computed that represents the evidence of true building according to the image gradient field and line segment definitions.
The problem of characterization of buildings within the urban/suburban zones could be an exceptionally complicated one that has numerous applications in a vast areas. Automatic creation of 3D urban city maps can be an imaginative way for giving geometric information for varieties of applications such as civilian emergency situations, natural disaster management (flooding, earthquakes, and landslides), military situations (active engagement of forces, counter-terrorism and peace keeping measures), urban planning, airport hazard analysis, and statistical geographic localization (such as health, crime, and past natural disasters).
Here we will apply a combination of image processing algorithms in order to extract specific image informations that will be used and combined later. The outputs of this section are: The gradient field of the image, including both gradient magnitude and direction. • A set of straight line segments (location, extent, orienta- tion) found in the image. • A set of image segments that represent similar image regions at various sensitivities.

Gradient field

Here we will generate the gradient field of the image with both the direction and magnitude for each pixel.

- To reduce the noise, we apply a 3×3 pixel Gaussian low-pass filter.
- We convolve 2×2 pixel horizontal and vertical masks across the image in order to compute the horizontal and vertical gradients
- Finally, the gradient magnitude and direction are estimated from the horizontal and vertical gradients at each pixel

Straight line extraction

Using the gradient field computed, we will extract a set of straight-line segments from the image. To achieve this goal, we will use the Burns line detector which follows this procedure:

1. Partition pixels into regions based on the gradient orientation values.
2. Form line support regions from groups of 4-connected pixels that share the same gradient orientation region.
3. Eliminate line support regions that have an area smaller than a specified threshold.
4. Repeat steps 1, 2, and 3 by shifting the gradient bins to produce a second set of line support regions.
5. Use a voting scheme to select preferred lines from the two sets (original and shifted) of candidate lines.
6. For each line support region, compute the line represented by that region by performing a least-squares fit.

Line linking and filtering

At this level, the goal is to link collinear line segments that are separated by very small gaps, as the following:

1. Sort the lines.
2. Utilize a divide-and-conquer strategy to productively determine adjacent sets of lines.
3. Test each match of adjacent lines to decide whether they should be connected. All 5 criteria, which are outlined in Figure 1, must be fulfilled for a combination of lines to be linked.

We also use two processes of line filtering to remove inefficient lines. One is based on the line length because long lines help to form good footprint assumptions, and the second one employs the gradient fields, separating regions of high contrast from low contrast.
Finally, we combine both the image segments and the line detected using a Canny edge detection algorithm to generate some rooftop hypothesis. This method is very efficient but its accuracy depends on the choice of parameters and the image processing algorithms used during the process.

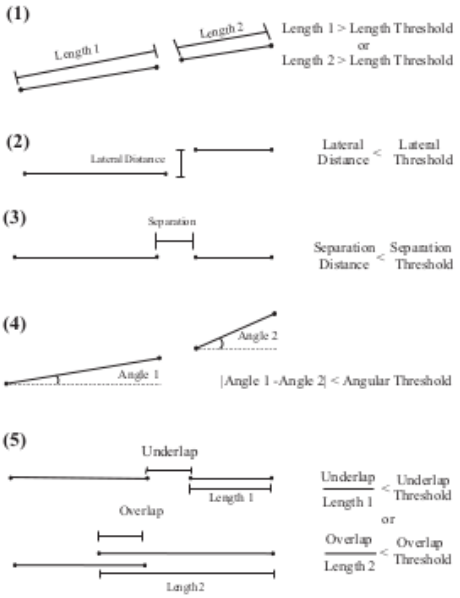


Fig 8: Line filtering and linking processes

Fig 9; Detected lines

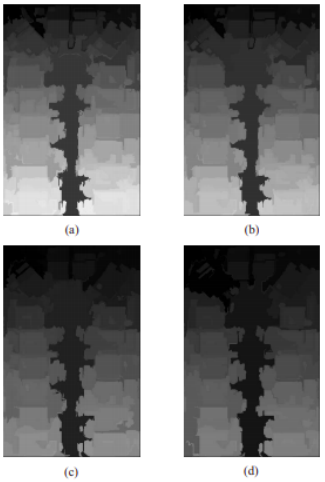


Fig 10 : Gradient field

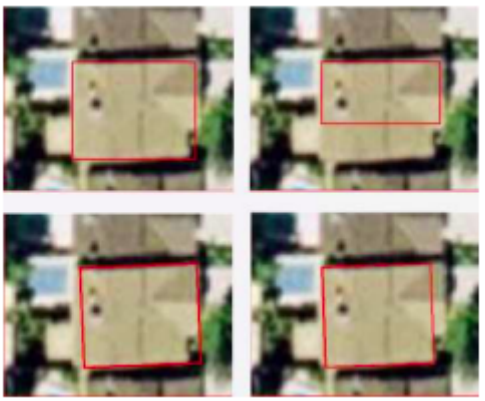


Fig 11 : Rooftop hypothesis

4.1.2 Deep learning

It is possible to generate building footprints by manual tools to draw outlines of each building. But, it is a labor process. Here, we will use the ArcGIS API to train a deep learning model to extract the footprints from satellite imagery.

ArcGIS [1] is a mapping and analysis system that provides contextual tools for mapping and spatial analysis. You only have to enter aerial images to train the model then finally you can have your images with the border outlines.

Fig 13 : A subset of detected building footprints

Even if this method using DL has proven its efficiency, it is very hard to use and requires a large amount of data in order to train the DL models (> 30Gb) and also more CPU power.

4.1.3 OpenStreetMap

OpenStreetMap (OSM) is a collaborative online mapping project which aims to build a free geographic database of the world (allowing for example to create maps under

[1] https://developers.arcgis.com/python/sample-notebooks/automate-building-footprint-extraction-using-instance-segmentation/

free license), using the GPS system and other free data. It was started in July 2004 by Steve Coast at University College London.

Through the use of computer resources based on the Internet which allow the intervention and collaboration of any volunteer user, OpenStreetMap is based on geomatics 2.0, voluntary geographic information and neogeography, the tools of which make up the GeoWeb.

The data can be downloaded using various tools linked to the OSM project, the mastery of which may require some expertise in the world of geographic information and the structure of OSM data.

Turbo overpass

Overpass-turbo is a site that offers a multilingual graphical interface on top of the Overpass API in order to make it easier for users to learn. An integrated map allows you to select the area of interest. A wizard is used to automatically generate the Overpass API language code for simple requests. A preview of the recovered data is available on the map and it is possible to export them to geographic data formats (GPX, KML, GeoJSON).

With turbo overpass you can run Overpass API queries and analyze OSM data interactively on a map. There is a built-in wizard that makes creating queries super easy.
More information about turbo overpass and creating Overpass queries can be found on the OSM wiki.

Overpass request

The Overpass API allows you to query OSM data according to your own search criteria. For this, there is a specific query language.

In addition to the normal Overpass API requests, the following shortcuts can be used in turbo overpass:

{{bbox}} - bbox coordinates of the current map view

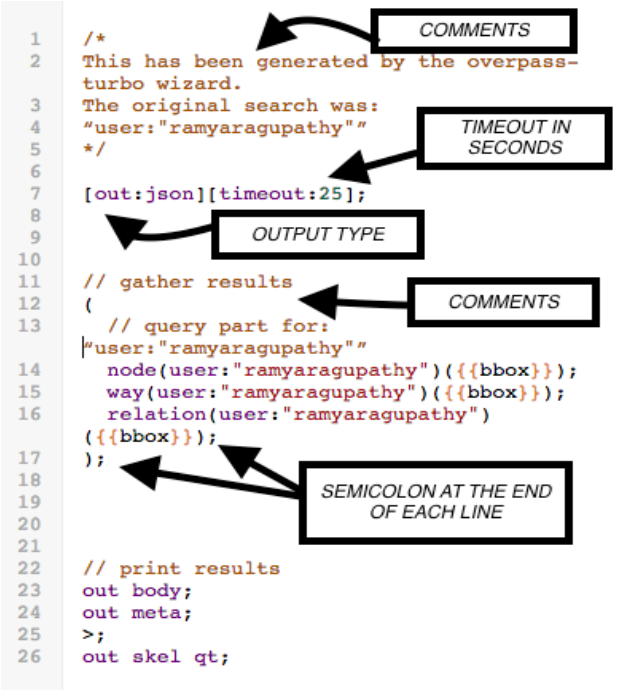
{{center}} - coordinates of the center of the map

{{date:...}} - elapsed time interval in ISO 8601 date-time-string format (eg "24 hours")

{{style:...}} - define a MapCSS stylesheet

Arbitrary shortcuts can be defined by specifying {{shortcut = value}} somewhere in the script.

You can find more Overpass shortcuts, more information on the above and usage examples in the OSM wiki.




```
1  /*
2  This has been generated by the overpass-
   turbo wizard.
3  The original search was:
4  “user:"ramyaragupathy"”
5  */
6
7  [out:json][timeout:25];
8
9
10
11 // gather results
12 (
13   // query part for:
   “user:"ramyaragupathy"”
14   node(user:"ramyaragupathy")({{bbox}});
15   way(user:"ramyaragupathy")({{bbox}});
16   relation(user:"ramyaragupathy")
   ({{bbox}});
17 );
18
19
20
21
22 // print results
23 out body;
24 out meta;
25 >;
26 out skel qt;
```

Fig 14 : Overpass QL

Use case

In order to detect the building in this Colmenarejo map, we will first look for the geographical coordinates of this squared area. In our request, we should enter those

coordinates as a parameter of the "way" function. Then, we will specify that we are only looking for a "house" type of building.

We can see in the figure the outlines of the building and when we click on one of them we can find important information like the house number, post code, address of the street, the type of building and how many levels it contains.

```
way(40.54473,-4.01731,40.54810,-4.00889)[building="house"];
/*added by auto repair*/
(._;>;);
/*end of auto repair*/
out;
```

Fig 15 : the query script necessary to detect houses in the Colme area

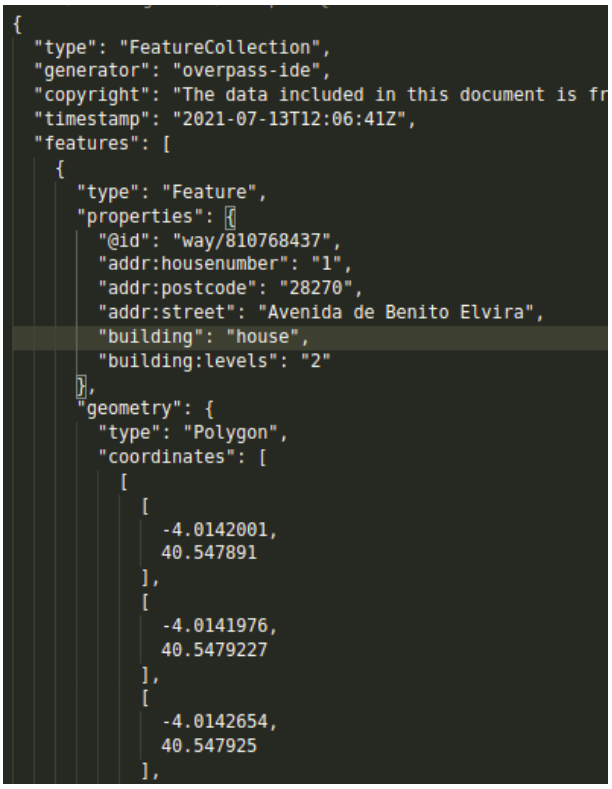
Fig 16: The detected houses

Fig 17: a zoom on a detected house

Export and data formatting

The output of this request and data corresponding to the building in this area will be exported as a geoJSON file containing the information about each house and the

coordinates of the edges of those buildings. Using a small python script we will format those geoJSON to save only the information that is important for us : the coordinates. Between the coordinates of each building, we insert triple “a” word to distinguish between houses.



```
{
  "type": "FeatureCollection",
  "generator": "overpass-ide",
  "copyright": "The data included in this document is fr
  "timestamp": "2021-07-13T12:06:41Z",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "@id": "way/810768437",
        "addr:housenumber": "1",
        "addr:postcode": "28270",
        "addr:street": "Avenida de Benito Elvira",
        "building": "house",
        "building:levels": "2"
      },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -4.0142001,
              40.547891
            ],
            [
              -4.0141976,
              40.5479227
            ],
            [
              -4.0142654,
              40.547925
            ],
```

Fig 18: the GeoJson file

```
 1 -4.0142001 40.547891
 2 -4.0141976 40.5479227
 3 -4.0142654 40.547925
 4 -4.0142663 40.547894
 5 -4.0142325 40.5478924
 6 -4.0142346 40.5478581
 7 -4.014223 40.5478575
 8 -4.0142243 40.5478361
 9 -4.014136 40.5478329
10 -4.0141345 40.5478552
11 -4.0141142 40.5478546
12 -4.0141119 40.5478877
13 -4.0142001 40.547891
14 aaa
15 -4.013506 40.546393
16 -4.0134182 40.5464557
17 -4.0134485 40.5464788
18 -4.0134625 40.5464692
19 -4.0134863 40.5464869
20 -4.0135012 40.5464764
21 -4.0135382 40.5464502
22 -4.0135508 40.5464413
23 -4.0135305 40.5464248
24 -4.0135383 40.5464192
25 -4.013506 40.546393
26 aaa
```

Fig 19: the formatted data

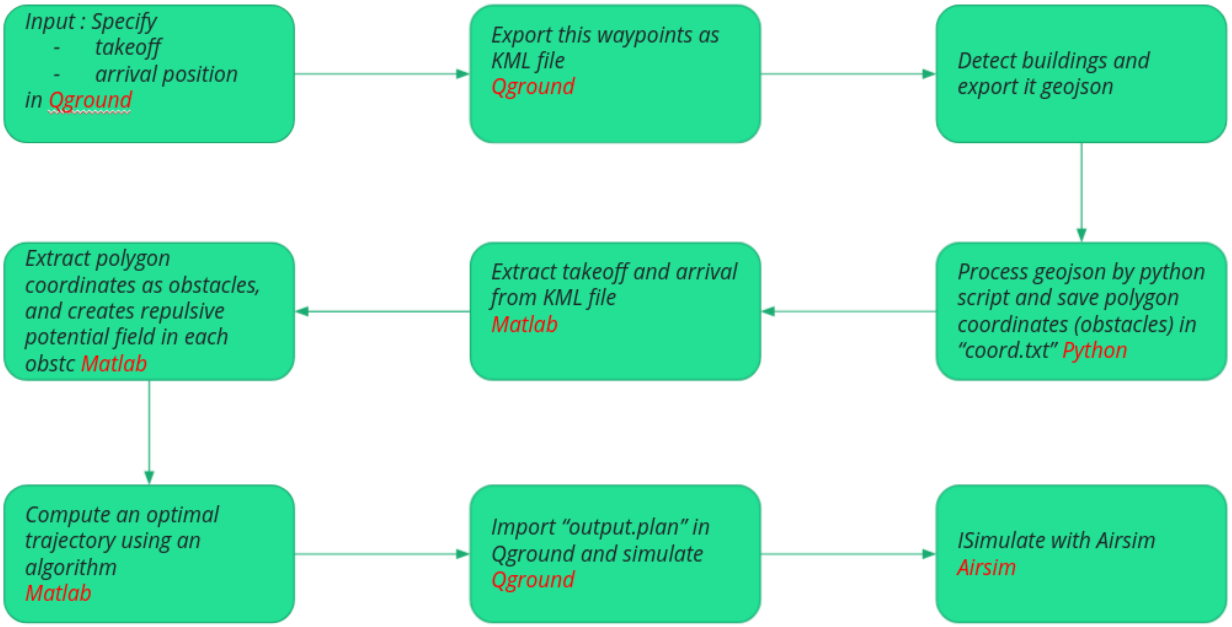
```
import geojson
geojson.geometry.Geometry.__init__.__defaults__ = (None, False, 10)
with open('exp.geojson') as f:
    data = geojson.load(f)
allobs = data['features'] #Your first point


for i in range(len(allobs)):
    obs = allobs[i]['geometry']['coordinates'][0]
    for j in range(len(obs)):
        print(obs[j][0],obs[j][1])
    print('aaa')
```

Fig 20 :

the python script for formatting data

4.2 Workflow

Once we know how to perform the house detection, it becomes easy to put in place the workflow as the following :



4.3 Path planning

The definition of the path planning problem is very straightforward: "find a collision-free motion between an initial (start) and a final configuration (goal) within a specified environment". The simplest situation is when the path is to be planned in a static and known environment; however, more generally, the path planning problem can be formulated for any robotic system subject to kinematic constraints, in a dynamic and unknown environment.

Path planning algorithms are usually divided according to the methodologies used to generate the geometric path, namely:

- roadmap techniques
- cell decomposition algorithms

- artificial potential methods.

In this study, we will study and compare, the artificial potential methods with APF and cell decomposition algorithms using Dijkstra, Astar and Dynamic Programming.

4.3.1 Artificial Potential Field APF

The goal and obstacles act like charged surfaces and the total potential creates the imaginary force on the robot. This imaginary force attracts the robot towards the goal and keeps it away from an obstacle as shown. The robot follows the negative gradient to avoid the obstacle and reach the target point.

Obstacles and the goal are “active” entities which exert respectively virtual “repulsive” and “attractive” forces onto the robot which is “passive”. The robot starting with null velocity tends to approach the goal deviatong the trajectory according to the position of detected obstacles. The heading direction of the robot depends on the sum of two vectors corresponding to the goal attraction and the repulsion due to the combined action of one or more obstacles.

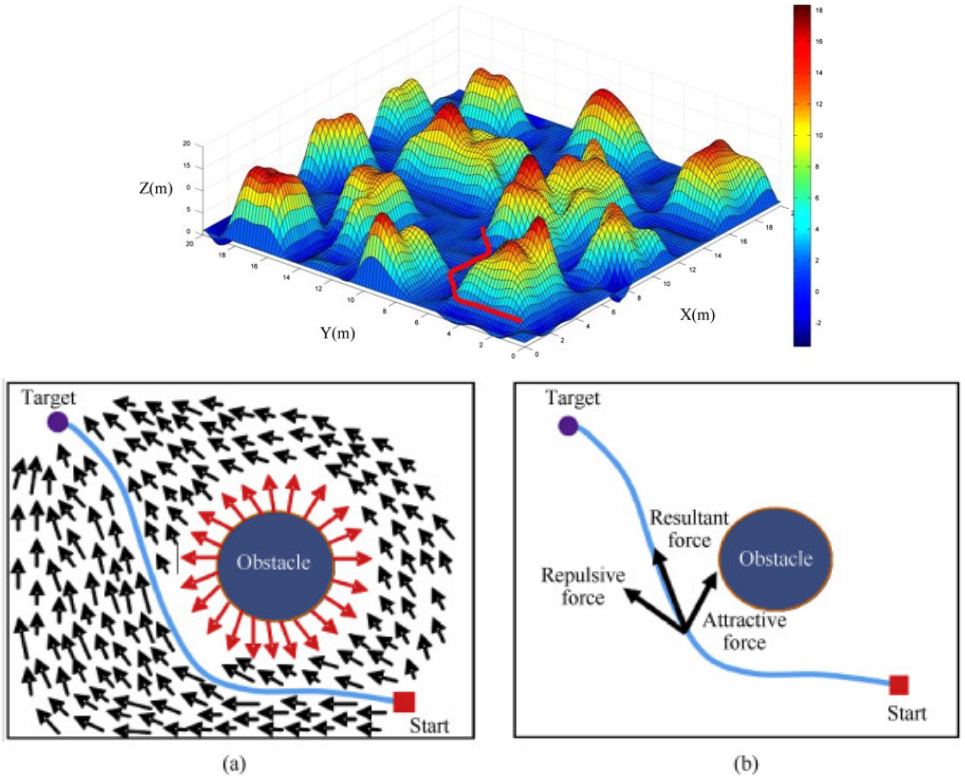


Fig 21: Visual heuristic of the APF

4.3.1.a APF simulation in our case

After data formatting and obstacle defining, we create a repulsive field for each house. Then, we create an attractive field around the finishing position. Finally, we sum both potentiel fields and we deduce the optimal path.

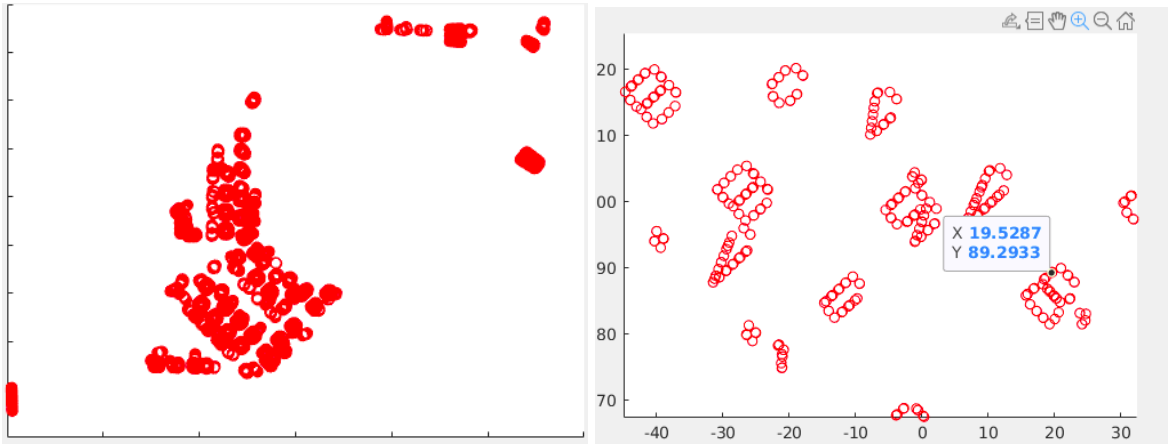


Fig 22: The repulsive fields generated

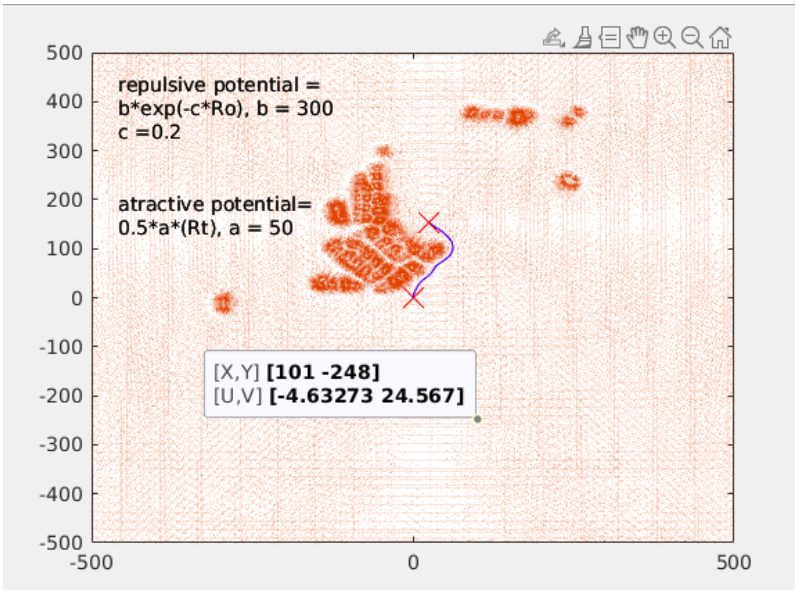
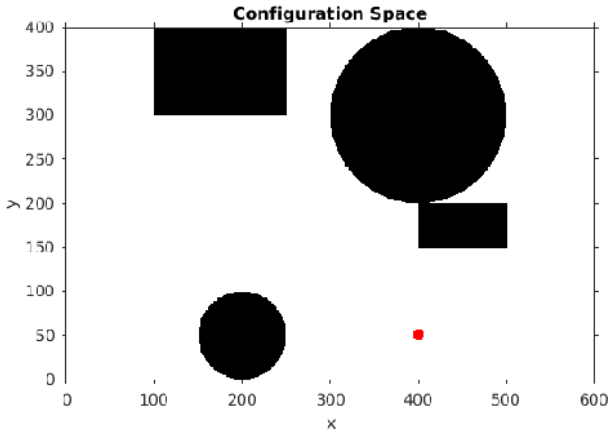
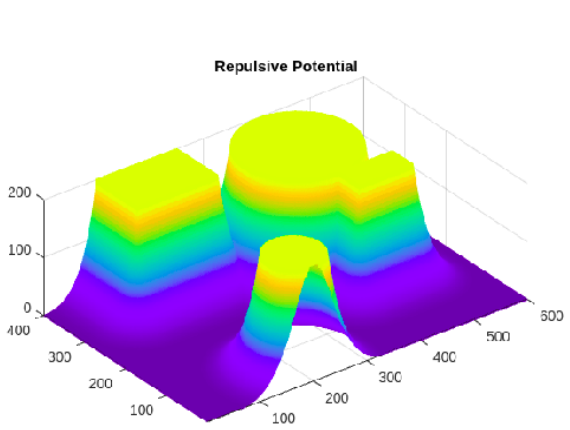


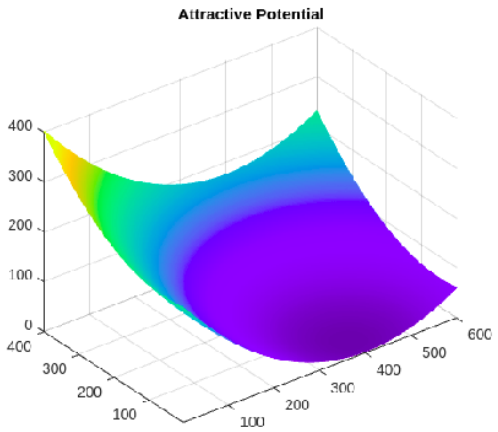
Fig 23: the total PF and the final trajectory generated

4.3.1.b APF in 3d

As an additional explanation, here we will show to what extent the APF method is very simple and efficient to use even in a 3d map or in real time (SLAM).
At first, in the configuration space, we define static obstacles that we create in the script. Then a global repulsive potential field of all obstacles is computed and added to an attractive one around the goal which gives us a total potential field. And so if we put an object into the starting position it will naturally reach the final destination.







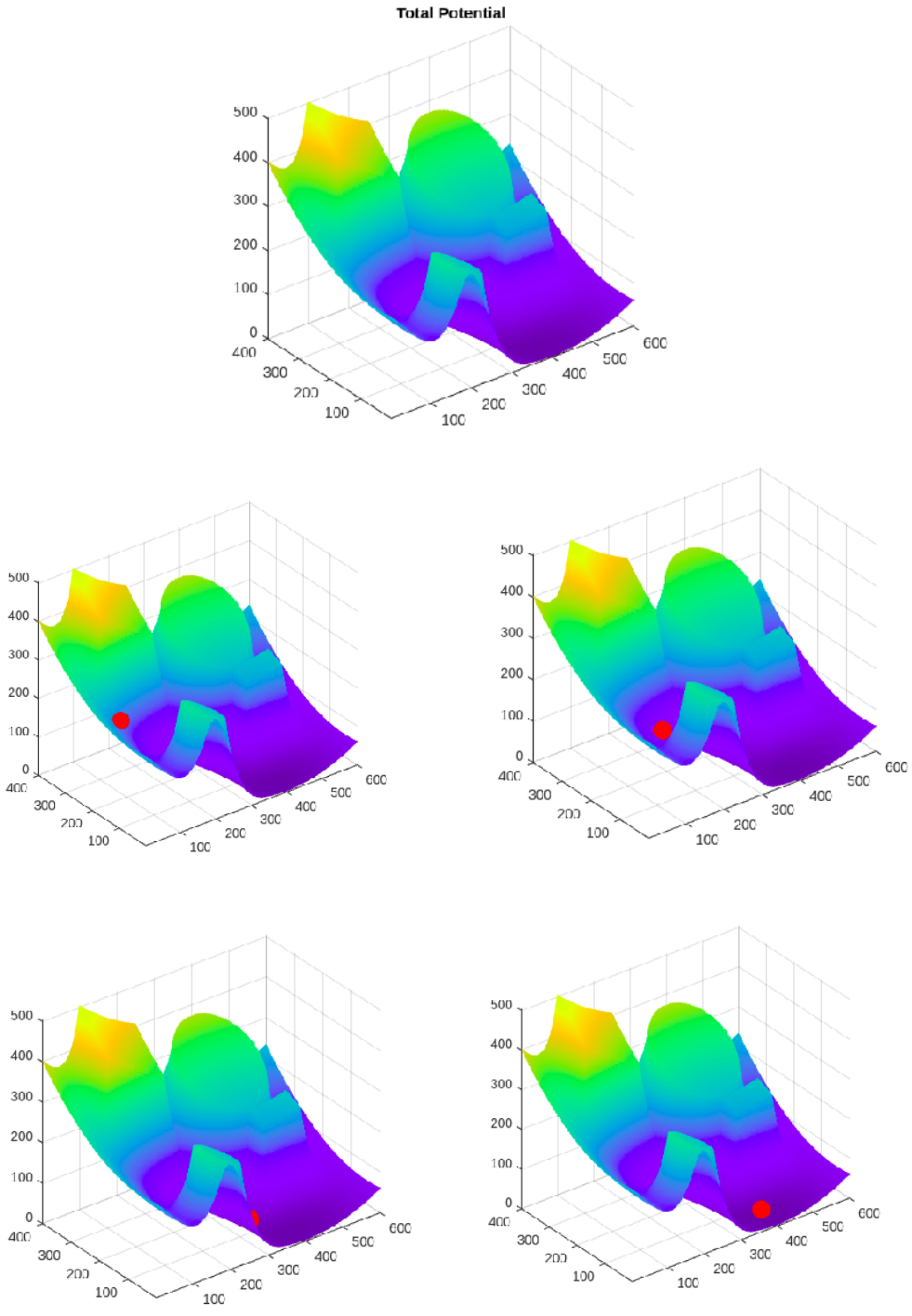


Fig 24: APF steps in 3d environments

4.3.2 Comparative of three cell decomposition algorithms:

Dijkstra

- Well known
- Shortest Path between nodes
- Discover neighbour nodes
- This is a greedy algorithm to find the minimum distance from a node to all other nodes. At each iteration of the algorithm, we choose the minimum distance vertex from all unvisited vertices in the graph,

A star

- Most effective
- It introduces a heuristic function to the dijkstra algorithm
- The principle of the A* algorithm is :
    - starting from a specific node of a graph, it constructs a tree of paths starting from that node
    - expanding paths one step at a time, until one of its paths ends at the predetermined goal node

Dynamic Programming

- break up a problem into multiple related sub-problems.
- calculate the distance of the goal from all the points in the map.
- Distance to further points (problem) is computed based on the pre-computed distance to nearer points (sub-problem).

4.4 Path Generation using Matlab

4.4.1 House detection and node sampling

First, we define the green obstacles that represent the buildings detected and after it we sample the mep and generate random nodes or keypoints. The final trajectory computed will be a series of special key points.

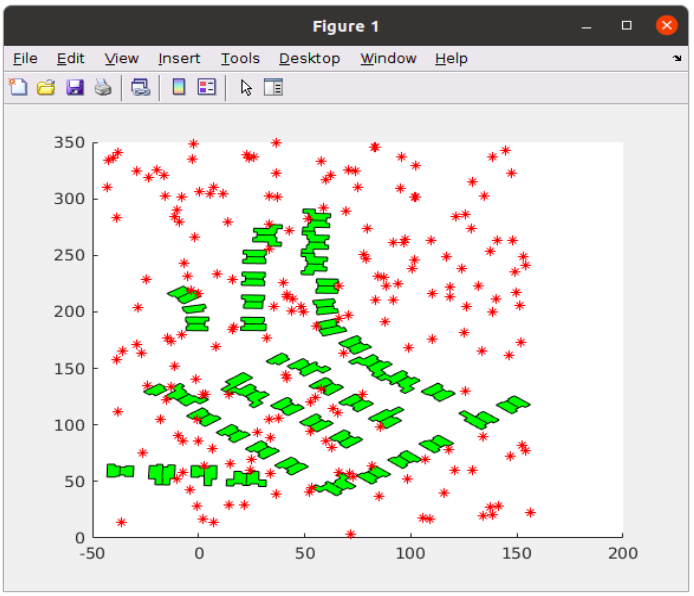


Fig 25: Houses defined as polygon obstacles in green and node sampling of the map with red points

4.4.2 Result : Shortest Path

We compute the algorithms to generate the path corresponding to each mpath planning method.

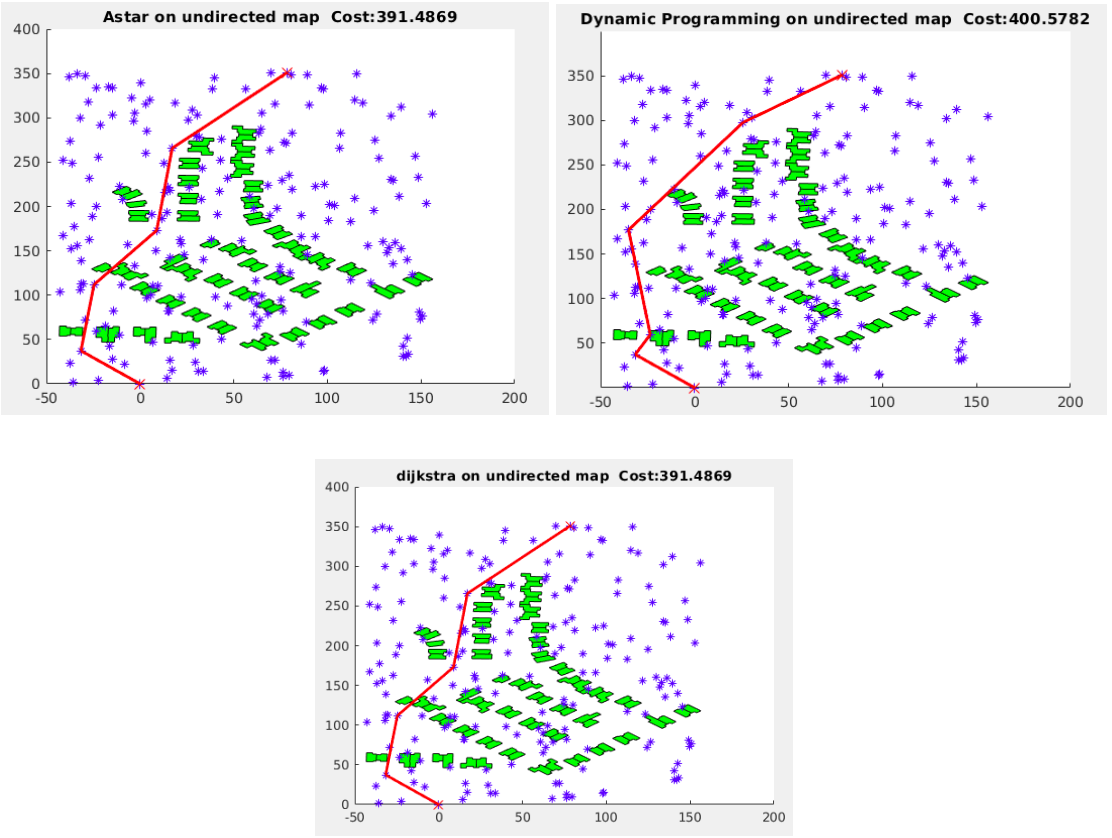


Fig 26: The path generated by the three algorithms

4.5. Smoothing

Path planning and trajectory planning are significant issues within the field of UAVs and, more by and large, within the field of Robotics. Undoubtedly, the trend for robots and autonomous machines is to operate at increasingly high speed. Such speed may affect the UAV's motion since extreme performances are required from the control system. Thus, specific care ought to be put in producing a trajectory that may be executed at high speed, but at the same time safe for the robot, in terms of dodging over the top increasing velocities of the actuators and vibrations of the mechanical structure. Such a trajectory is characterized as smooth.

Most of the trajectories generated consist of straight lines and sharp turns. In the figure, we can see that the robot has to pass by sharp turns at A, B, C, D, and E. Such a path is prohibited because the UAV needs to slow down at each sharp turn and it can not make it suddenly. For example, in the case of a delivery drone, there will be

multiple hard stops that can cause damage to the delivered items. Furthermore, from an external point of view in the same environment, such a trajectory looks unnatural and it's harder to predict the uav's next position in order to avoid collision. Thus, a smooth and continuous path is necessary for easier navigation.

At a first level, we used the Bezier curves to smooth the trajectory. Bézier curves are parametric polynomial curves developed first to design automotive body parts. These curves are formed by the bezier control points that are computed from a polynomial interpolation as shown in the figure.

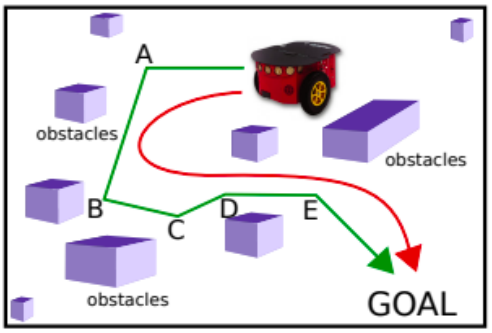


Fig 27 : sharp turns

The Bézier curve of the n+1 control points $(\mathbf{P}_0, ..., \mathbf{P}_n)$, is the set of points defined by the graphic representation of the polynom $P(t) = \sum_{i=0}^{n} B_i^n(t) \cdot \mathbf{P}_i$, where $t \in [0\ ;\ 1]$ and B_i^n the Bernstein polynoms.

The points $\mathbf{P}_0, ..., \mathbf{P}_n$ form the « **control polygon** of Bézier ».

Exemples

- For $n = 3$:

$$1^3 = ((1-t)+t)^3 = (1-t)^3 + 3(1-t)^2 t + 3(1-t)t^2 + t^3$$

Then we add the n+1 control points for each polynom coeffecient which gives us,

$$P(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3$$

with $t \in [0\ ;\ 1]$. This is the Bézier curve of degree 3.

Fig 28: Mathematical approach of Bezier curves

In our case and for a better result, we will use a generalisation of the bezier curves called the B-splines and we will control the parameter as possible so that the trajectory doesn't diverge from the sharp turns. In the figure, we can see the normal A* trajectory and on pink the smooth version of it.

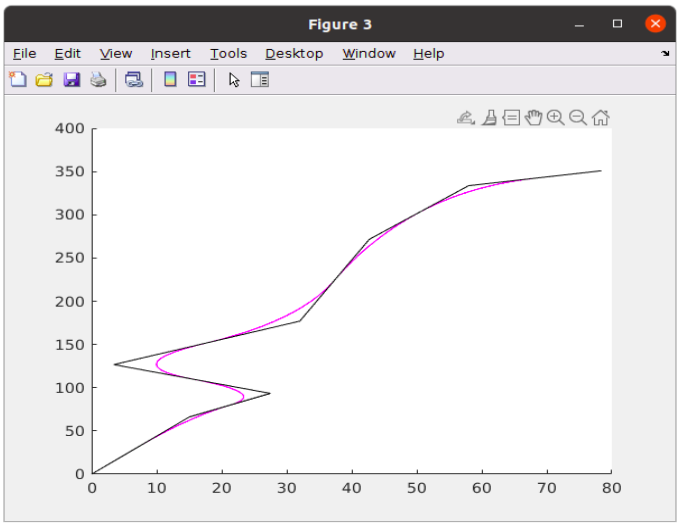


Fig 29: Smoothing of the A star trajectory

5. Results

Cost

In the code script, we added a cost variable which represents the euclidean distance of each path.
Dijkstra : 382.68
A star : 387.03
Dynamic Programming : 379.20
We can see that in this map the DP is the best path planning method to use.

6. Simulation

Matlab

After path planning, we generate trajectories as multiple keypoints. Those keypoints' coordinates will be exported as a .plan file.

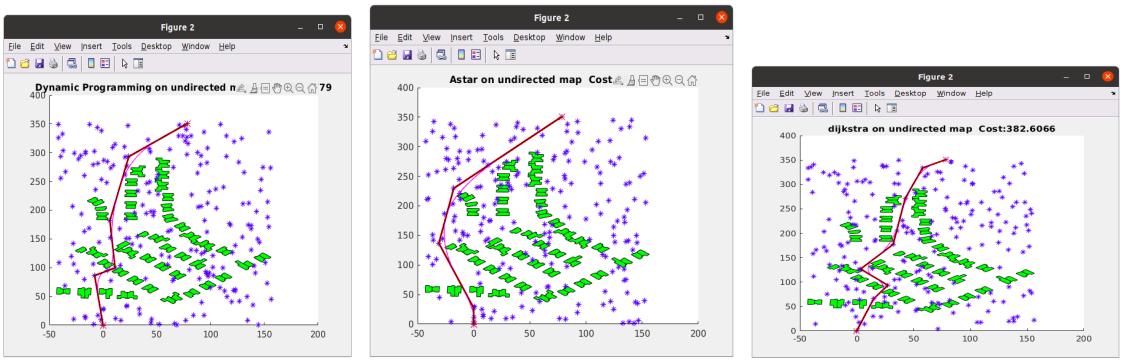


Fig 30 : Path planning in Matlab

QGroundControl

The input file for the mission planner QGC that contains the trajectory is the .plan file.

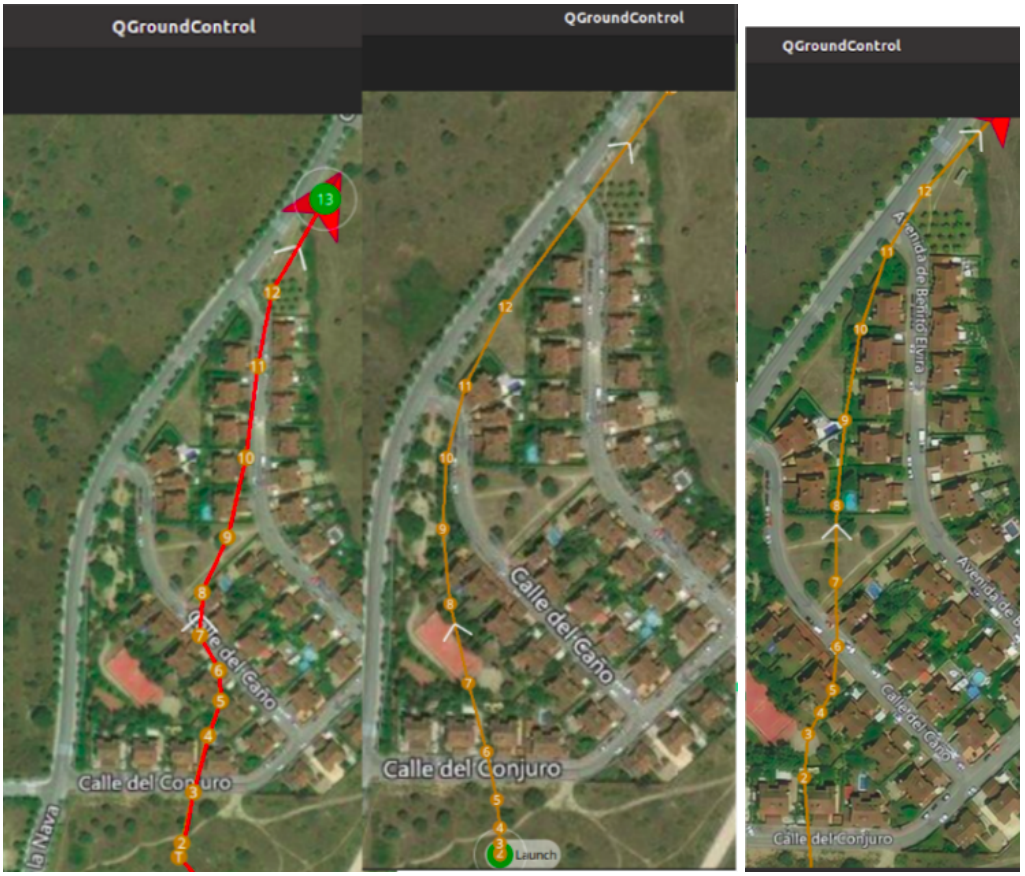


Fig 31 : Mission planning with QGC

Airsim

At this level, we simulate the trajectory from the take-off position to the starting position to the final goal. This simulation gives us important information about the velocity and speed of the drone, the time necessary for this trip ... A good improvement can be developing an automated system to add a 3D version of this colmenarejo map to unreal engine in order to see if the drone will in fact avoid the obstacles or not.

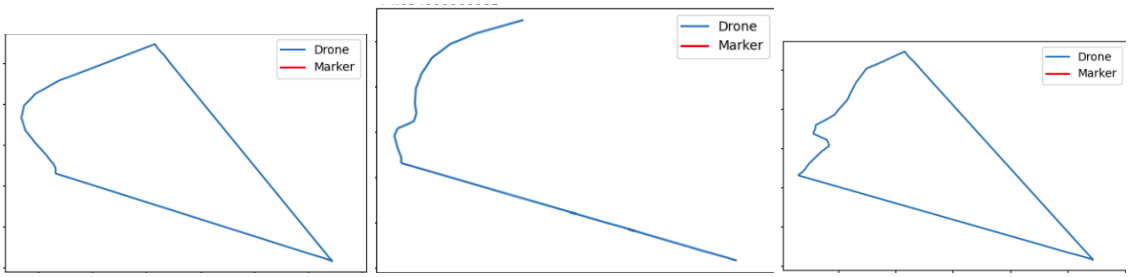


Fig 32 : Path simulation in Airsim

7. Conclusion

In this study, we analysed different ways of building footprint detection. Deep learning methods using neural networks are very specific and precise and can be very interesting to use in both local and global path planning but they require a large amount of data to train its models, and thus its efficiency is questionable. On the other hand, line extraction methods can be efficient and simple to use, but they are less precise than DL models. Finally, OpenStreetMap offers various services and a large amount of data to use. The overpass turbo platform is very interesting to query information from the OSM database and can be used to solve different problems.

After extracting houses from the map and defining their coordinates, we decided to test and compare multiple path planning methods. Starting with APF, which is known as the most simple and efficient method to use. But this method has one big issue which is the local minimum trap that is hard to solve. To overcome such problems, we decided to use cell decomposition algorithms like A star, dijkstra and DP. We finally introduced a cost variable to compare between the trajectories generated by these algorithms.

We finally decided to use a bezier spline to smooth the trajectories in order to have a more realistic output which will make the navigation easier for the drone.

8. Improvements

- ## Deadlock of APF

  The artificial potential field (APF) give simple and efficient trajectories. Be that as it may, this strategy has one big issue which can trap an uav before reaching its goal called the local minimum problem. It is sometimes unavoidable when an object moves in unknown environments, because the object cannot foresee local minima before it detects obstacles forming the local minima. The avoidance of local minima has been an active research topic in the potential field based path planning.

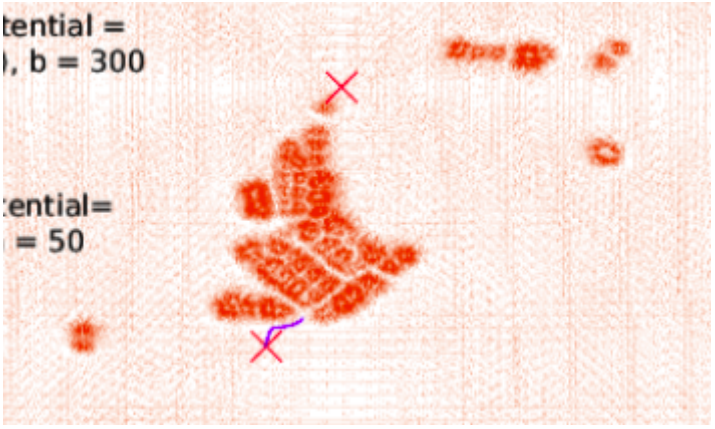


Fig 33 : Minimum local point

- ## Add security margin

  After the smoothing step, I realised that the trajectory doesn't avoid all the building's edges. Thus, it is necessary to either change the smoothing parameters to avoid such things, or add a security margin to the building. In

order words, a 40x40 box house would be represented by a 50x50 with a 10' margin before applying the path planning methods.

Fig 34 : Security margin for after-smoothing obstacle avoindance

• 3D path planning

In this study, we only focused on 2D environments. In order to expand the same methodology to a 3D environment, we should take into consideration the height parameter while exporting data from OSM and while generating the path trajectory.

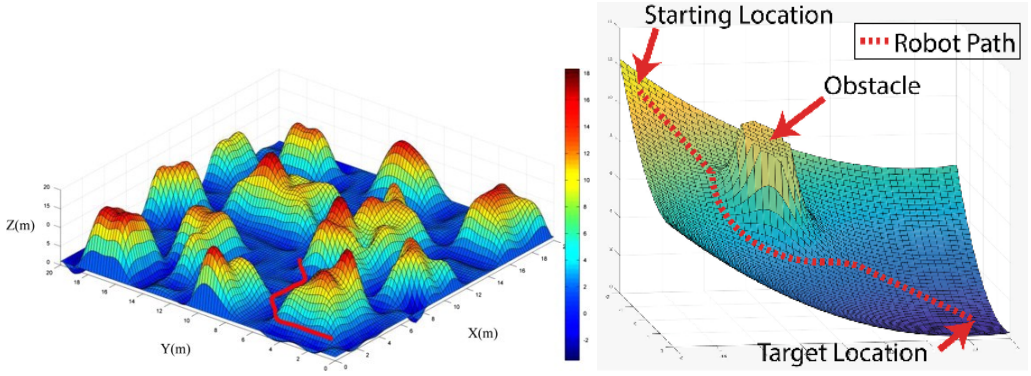


Fig 35 : 3D path planning

• Real-time path planning and SLAM

SLAM (synchronous localization and mapping) is a strategy utilized for UAVs that lets you construct a map and localize your vehicle in that map at the same

time. SLAM calculations permit the vehicle to map unknown environments. Engineers utilize the map data for path planning and obstacle avoidance.

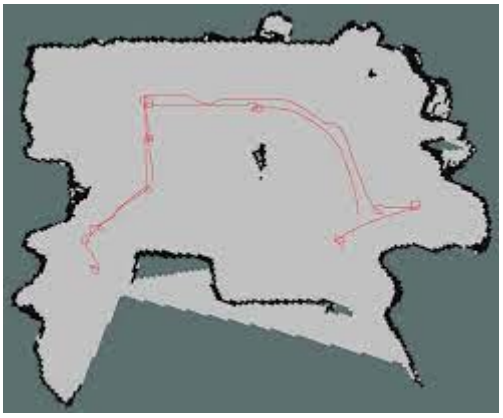


Fig 36 : SLAM

9. Bibliography

Note: the code developed and used in this project can be found at : https://github.com/YasLbk/stageuc3m

- ☐ OpenStreetMap Wiki visited on 15 June 2021.
- ☐ overpass turbo visited on 17 June 2021.
- ☐ Home - AirSim visited on 3 June 2021.
- ☐ (PDF) Automatic Building Detection in Aerial and Satellite Images visited on 20 June 2021.
- ☐ Automate Building Footprint Extraction using Deep learning visited on 22 June 2021.
- ☐ https://github.com/fuzailpalnak/building-footprint-segmentation visited on 27 June 2021.
- ☐ A review: On path planning strategies for navigation of mobile robot visited on 08 July 2021.
- ☐ UAV path planning using artificial potential field method updated by optimal control theory visited on 11 July 2021.
- ☐ Comparison of Path Planning Algorithms for an Unmanned Aerial Vehicle Deployment Under Threats visited on 15 July 2021.
- ☐ A Path Planning Algorithm for UAV Based on Improved Q-Learning visited on 17 July 2021.
- ☐ Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges visited on 27 July 2021.
- ☐ Bézier curve visited on 12 August 2021.
- ☐ cubic B-spline visited on 15 August 2021.
- ☐ An efficient approach to 3D path planning visited on 17 August 2021.