Article

Not peer-reviewed version

Energy-Optimized 3D Path Planning for Unmanned Aerial Vehicle

István Nagy * and Edit Laufer

Posted Date: 30 July 2024

doi: 10.20944/preprints202407.1465.v2

Keywords: UAV; trajectory planning; trajectory optimization; 3D environment; energy map

Article

Energy-Optimized 3D Path Planning for Unmanned Aerial Vehicles

István Nagy * and Edit Laufer

Bánki Donát Faculty of Mechanical and Safety Engineering, Óbuda University, Bécsi 96/b, H-1034 Budapest, Hungary; laufer.edit@bgk.uni-obuda.hu

* Correspondence: nagy.istvan@bgk.uni-obuda.hu

Abstract: Drone technology has undoubtedly become an integral part of our everyday life these days. The business and industrial use of unmanned aerial vehicles (UAV) can provide advantageous solutions in many areas of life, and they are also optimal for emergency situations and for accessing hard-to-reach places. However, their application poses numerous technological and regulatory challenges to be overcome. One of the weak links in the operation of UAVs is the limited availability of energy. In order to address this issue authors developed a novel trajectory planning method for UAVs to optimize the energy necessity during flight. First, an "energy map" was created, which was the basis for trajectory planning, i.e., determining the energy consumption of the individual components. This was followed by configuring the 3D environment including partitioning of the work space (WS), in fact, defining the free spaces, occupied spaces (obstacles) and semi-occupied/free spaces. Then the corresponding graph-like path(s) were generated on the basis of the partitioned space, where a graph search-based heuristic trajectory planning was initiated, taking into account the most important wind conditions including velocity and direction. Finally, in order to test the theoretical results, some sample environments were created to test and analyse the proposed path generations. The method eventually proposed was able to determine the optimal path in terms of energy consumption.

Keywords: UAV; trajectory planning; trajectory optimization; 3D environment; energy map

1. Introduction:

The ever-increasing development of drone technology, including courier drones, precision agro-culture "agro-drones", military drones, drones used in emergency situations, as well as in different industrial areas, means that engineers are constantly faced with new technical challenges [1].

By definition, the Unmanned Aerial Vehicle (UAV) is an unmanned aircraft (UA) that can fly autonomously, as opposed to the Remotely Piloted Aircraft (RPA), which is semiautonomous. The Unmanned Aerial Systems (UAS) is a wider term encompassing the controller on the ground, the aircraft in the air, and the communication between them, thus the whole system. The RPA can also be part of a system called Remotely Piloted Aircraft System (RPAS), where the whole system is created from RPA (see above), plus the Ground Control Station (GCS) [2].

There are several classifications of drones, for classic UAVs, this classification can be the following: multi-rotor, fixed wing, single-rotor, fixed-wing hybrid VTOL (Vertical Take-Off and Landing), MAV (Micro Air Vehicle: < 1g), sUAS (small UAS:< 25kg), etc. More relevant to this study is the classification of "open (public) category", or "specific-category" (drones (UAVs) over 25kg). There is also the "game category" (drones under 120-gram weight), which will in this project be disregarded.

The first two categories will be considered within this study, given their flight altitude, 120 meters over the nearest point of ground, for "public category" drones. This dimension will be important in 3D map creation of the flying-environment of the drone.

 doi:10.20944/preprints202407.1465.v2

1.1. Brief Description of the Aims

As mentioned above, limited energy availability is one of the weak points of drone operation. This notion led to the development of the proposed trajectory planning method, which uses energy as effectively as possible, thus optimally during the flight. It required the creation of a drone "energy map", where the components consuming most energy were determined, so that the trajectory planning mechanism could be adjusted accordingly. Weather reports, especially wind reports of the terrain, had to be available during the drone's path planning.

The energy map creation is followed by the 3D environment configuration since this is how the drone interprets its environment. With the partitioning of the "work space" (WS), the following areas are created: "free spaces"; "occupied areas" and "semi -occupied/-free spaces". The occupied areas contain the obstacles, while the semi-occupied ones partly contain them. Depending on the used partitioning method and based on specific resolution of partitioning, the semi-occupied areas can be further partitioned. Once the environment has been partitioned, the modelling can begin. Through the modelling the drone represented as point can move on the configured WS. Obstacles, whose dimensions are increased by the radius (*R*) of the sphere drawn around the drone are called as configured obstacles (see Figure 1). In this way the physical dimensions of the drone are considered in the WS model. Naturally, extra "safety zones" can be added based on user requirements.

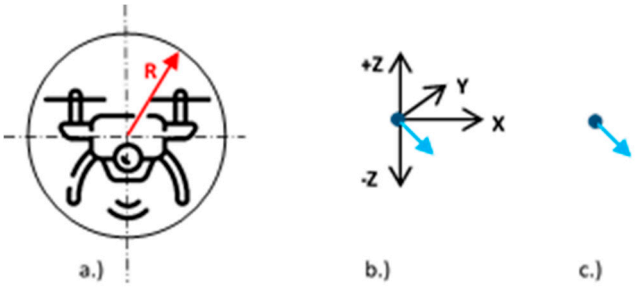


Figure 1. The process of the creation of point representation. (**a**) The drone and the sphere around the drone by "R" radius; (**b**) The drone represented as a point by its coordinates and orientation; (**c**) The drone represented as a point by its orientation (this will figure in the environment model).

1.2. Research Aspects of the Unmanned Aerial Vehicles (UAVs)

Research directions into UAVs can be grouped around three main areas, as outlined below.

- Group 1 includes publications describing drone parts and operating conditions of drones [3–5].

  The focus of these publications is the evaluation of the energy consumed by the different drone parts, based on which energy weights can be assigned to the different path segments. The created drone "energy map" revealed the most of the energy was consumed by the drive rotors. The power of these rotors is highly dependent on velocity, ascending, descending and the air resistance of the drone. The velocity can be controlled by the Electronic Speed Controller (ESC) or Flight Controller (FC).
- Group 2 incorporates literature on 3D path planning and environment modelling methods from the second research field. Space partitioning is studied e.g., in [6–10], while 3D path planning methods are presented in [11–18].

  These publications serve a good basis for designing a novel space partitioning and path planning algorithm. One of the shortcomings of the surveyed documentations is that almost none of them deal with partitioning of the WS, which may have two reasons for this as follows: a.) Drones are equipped with the environmental sensing equipment (RADAR or LIDAR sensors, cameras, vision systems, US sensors, etc.). b.) Drones fly a visible distance and are controlled via some joystick or remote controllers. GPS is standard equipment in drones.

  None of the examined works delved into the topic of energy used in aviation. In order to take this condition into account, several routes between the START and GOAL positions must be established, reviewed, and the optimal one requiring the least energy for execution, selected.

 doi:10.20944/preprints202407.1465.v2

Unfortunately, this will work in static environment, because recalculating and re-weighting the path-segments takes some time, depending on the complexity of the flight control (FC) processor.

- Last but not least, Group 3 discusses legal regulations of drone flight and civil aviation, with information that should be considered during the study (see [19]).

  Drone flight rules (policies) in the EU, or indeed the world, have not yet been fully developed. There are certain local provisions in place, but they are not completed. There are some drone categories (A, B, C), some flight altitudes, and the required permissions for drone flights (drone driving licenses). For this study, the most significant is taken into account, namely to maintain the 120 meters altitude from the nearest ground point.

2. Preliminary Studies

This section focuses on the studies about the drone's energy consumption and the work space partitioning.

2.1. Studies about the Drone Energy Consumption

There are different types of drones defined according to their power supply.

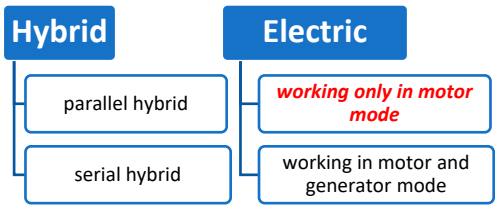


Figure 2. Classification of the drones according to power supply (the energy supply of the actual drone used in the article is colored by red).

In the case of hybrid systems, an Energy Management Strategy (EMS) system usually regulates and controls the efficient energy consumption. These EMSs can be based on different strategies, such as: Rule based systems; Intelligent based systems (Fuzzy or NN); Optimization based; etc. Classic electric UAVs, which operate only in motorized mode (where there is no recharging the battery) lack these systems, and energy savings has to be ensured by another approach. This is why energy efficient path planning was examined and improved in this project.

The energy consumption of the drone can be defined in two ways:

- practically: by direct measuring of the amps consumed by various parts of the drone.
- theoretically: with calculations of the energy consumptions of the parts.

For energy consuming calculations the following relations must be considered as a minimum. Engine power can be calculated based on forces, torques and moments on the x/y axis. Generally, Newton's Laws are used to calculate forces and moments. See Figure 3 for a better understanding of the equations.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

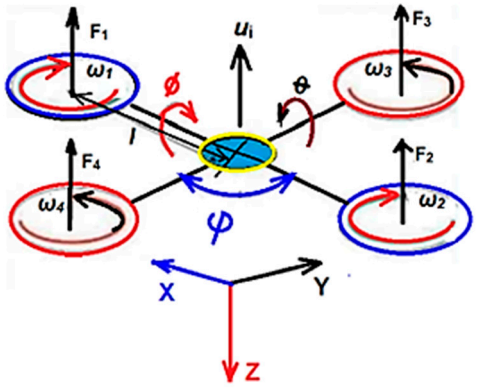


Figure 3. Forces, moments and torques used in drone flights (equations (1)-(7)), with the drone coordinate system, and roll, pitch, yaw movements were adopted from [4].

In general, the flight lift forces ($F_1 - F_4$) and M_x, M_y moments can be calculated as follows:

$$F_i = k_f \times \omega_i^2 \quad (1)$$

$$M_x = (F_3 - F_4) \times L \quad (2)$$

$$M_y = (F_1 - F_2) \times L \quad (3)$$

where "L" is the distance between 2 rotors.

The operation's conditions are given by the following relations:

1. **Hovering** motion: Equilibrium conditions for hovering: $mg = F_1 + F_2 + F_3 + F_4$; and all moments = 0;

   Equation of motion:

   $$m = F_1 + F_2 + F_3 + F_4 - mg; \quad m = 0 \quad (4)$$

2. **Rise/fall** motion: Conditions for operations: $mg < F_1 + F_2 + F_3 + F_4$ = ***rise***; $mg > F_1 + F_2 + F_3 + F_4$ = ***fall***; and all moments = 0;

   Equation of motion:

   $$m = F_1 + F_2 + F_3 + F_4 - mg; \quad m > 0 \quad (5)$$

3. **Yaw** motion: Conditions for hovering $mg = F_1 + F_2 + F_3 + F_4$; and all moments ≠ 0;

   Equation of motion:

   $$\textit{(mass * linear acceleration)} \quad m*a = F_1 + F_2 + F_3 + F_4 - mg; \quad (6)$$

   $$\textit{(}I_{ZZ}\textit{ * angular acceleration around "Z")} \quad I_{ZZ}*\alpha_Z = M_1 + M_2 + M_3 + M_4$$

4. **Pitch / Roll** motion: Conditions for hovering $mg < F_1 + F_2 + F_3 + F_4$; moments ≠ 0;

   Equation of motion:

   $$\textit{(mass * linear acceleration)} \quad m*a = F_1 + F_2 + F_3 + F_4 - mg; \quad (7)$$

   $$\textit{(}I_{XX}\textit{ * angular acceleration around "X")} \quad I_{XX}*\alpha_x = (F_3 - F_4) \times L$$

For the precise description of the forces and exact modelling of the drone, the rigid-body dynamical relations in 3D environment must be known. Initially, the reference coordinate system is set up along with the related movements. Afterward, the following effects on the drone are considered:

- aerodynamic forces: friction and drag of propellers rotating in air
- secondary aerodynamic effects: blade flapping, ground effect, local flow fields
- inertial counter torques: gravitation, acting at the center, affect the rotation of propellers.
- gyroscopic effect: change in the orientation of the drone body and plane rotation of propellers

The dynamic equation can then be formulated as follows (in essence, this is rigid body dynamics extended to 3D environment):

$$\begin{bmatrix} \vec{F} \\ \tau \end{bmatrix} = \begin{bmatrix} m_3 & 0_3 \\ 0_3 & \overrightarrow{I_3} \end{bmatrix} \begin{bmatrix} \vec{a} \\ \propto \end{bmatrix} + \begin{bmatrix} \omega \times m\vec{v} \\ \omega \times \overrightarrow{I_3}\omega \end{bmatrix} \quad (8)$$

 doi:10.20944/preprints202407.1465.v2

where: v – linear velocity (this is the origin linear velocity of the drone which will be modified by the wing velocity, see eq. (17) $v_{resulting}$), ω - angular velocity, α - angular acceleration, I - moment of inertia, a - linear acceleration, m - mass, F - total force, τ - total torque. The technical terms used earlier as well as energy consumptions concepts, are detailed in Figure 4.

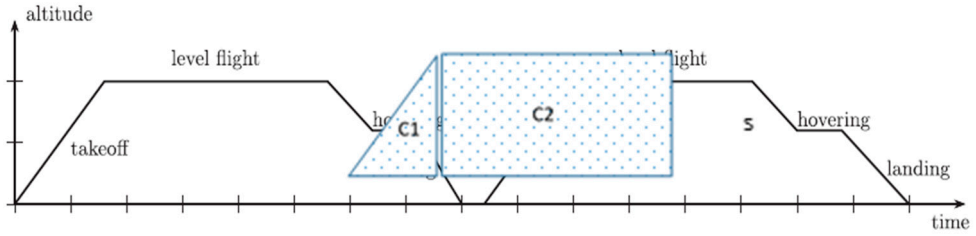


Figure 4. The idealized UAV flight pattern, where C_1 and C_2 areas are denote energy consumption, while S area is the energy saving region [20].

2.2. The Energy Evaluation

Many ideas and calculations (see equations (9)-(12)) have been applied to the energy calculations from [21], maybe with a slightly different interpretation. Generalizing, the drone's energy consumption can be summarized as follows:

$$E_{total} = E_{processors} + E_{motors} + E_{communication} + E_{peripherials} \quad (9)$$

where, $E_{(anything)} = P_{(anything)} * t$

Some constant and very low energy for ($E_{communication} + E_{peripherials}$) is assumed. The energy consumed by the processor(s), is specified depending on control system complexity:

$$E_{processors} = \sum_{i=1}^{nr.ofproc.} \int_{t0}^{t1} P_{processor(i)}(t) * dt \quad (10)$$

Furthermore, the energy consumed by motors:

$$E_{motors} = E_{takeoff}(a,w) + E_{hover}(a,w,t) + E_{move}(w,s,t,\rho) \quad (11)$$

where the dominant energy is:

$$E_{move}(w,s,t,\rho) = E_{drag}(d,\rho,k_{drag},S_{eff}) + E_{kinetic}(w,s) \quad (12)$$

This $E_{move}(w,s,t,\rho)$ is dominant and vital in different atmospheric conditions (this will be modelled later in the space model). Having completed the calculations for a particular case, the authors created the following "average energy map" of the drone, as seen in the diagram below (Phantom 4 drone pro/pro+ series).

 doi:10.20944/preprints202407.1465.v2

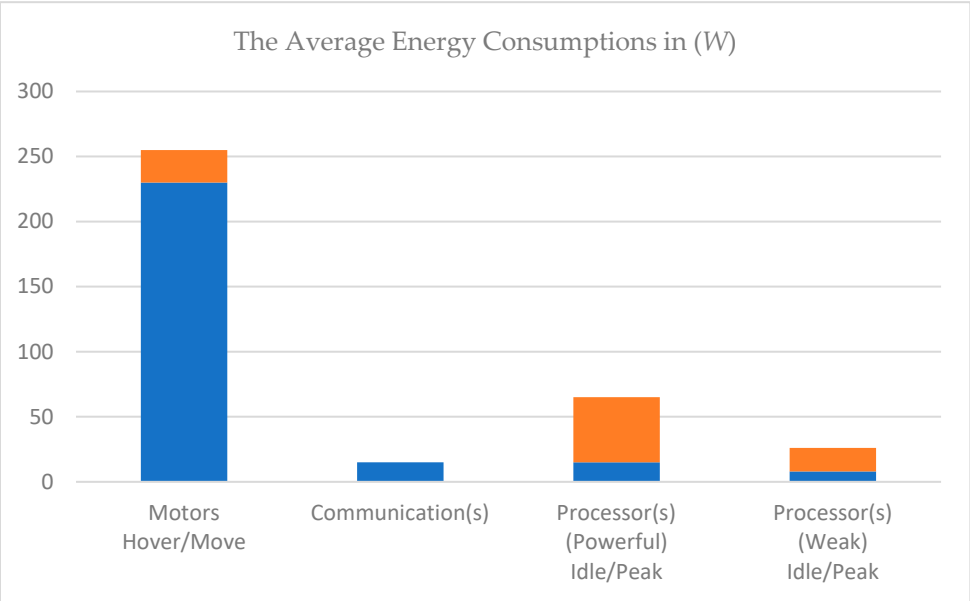


Figure 5. The generalized energy consumption of drone (hover – blue, move – orange, idle – blue, peak - orange).

These calculations were based on a "Phantom 4 Pro" drone was used. For energy calculations the simple math relations were used, where voltages are known and currents were measured: $P = U*I = E/t$. For thrust force calculations see equations (1-3), above. The two different colors of bars represent the "Hover/Move" energy of motors, and the "Idle/Peak" energy of the "Powerful" and "Weakly" loaded processors.

3. Environment Description and Work Space Partitioning

Most of the drones are equipped with modern positioning and distance-measuring sensors. Apart from this minimum requirements, even the most modestly equipped drones contain GPS and a few range measuring LEDs for obstacle detecting. Standard drones usually include the following positioning sensors: GPS, front/rear LEDs, camera system, forward/downward/rear vision systems, infrared sensing system, barometer, etc. Drones with basic equipment, based on FC complexity, can be controlled in different flight modes.

- ***P***-mode (positioning): works best when the GPS signal is strong. In addition to GPS, other sensors are used: stereo vision system, infra sensing system – obstacle avoiding, moving target tracking;
- ***S***-mode (sport): maximum maneuverability. Obstacle sensing disabled.
- ***A***-mode (altitude): only uses the barometer to sense altitude. Positioning is no longer available.

Regarding the work space partitioning, the current drone flight rules must be adhered to. The most important rules include:

- maintain the altitude of 120m
- stay at least 30m away from other people
- keep the drone in line-of-sight

From the aspects of the current study, the most crucial regulation is the first one, because it limits the volume (space) to be partitioned. On the other hand, if the drone is kept in visual line-of-sight (LoS), the WS partitioning may seem redundant, but the drone will not always be in LoS. The configuration of the WS can always be useful in the operation of unmanned aircraft, because often, if one of the sensors fails, the FC can only rely on the predefined path on the WS model.

There are numerous 3D partitioning methods, such as 4-tree (2D partitioning), 8-tree (3D partitioning, split around a point), KD-tree (partitioning along one dimension), R-tree, and the most general is the Grid (cube) rasterization.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

The basic concept of 4/8-tree partitioning: the area/space is divided into 4/8 subparts. Another division is made only on those squares/cubes, where the obstacles are located, or where part of obstacles can be found. As a result, the resolution of the division increases, as the drone comes closer to the obstacle.

A KD-Tree is a binary tree in which each node represents a K-dimensional point (hyperspace).

R(rectangle)-Tree groups nearby objects and represents them with their minimum bounding rectangle at the next higher level of the tree [13].

Voxel grid partitioning is the simplest, it is also popular, yet not truly effective, because the dimensions of the blocks (cubes) remain the same, regardless of the proximity of the obstacle. On the other hand, the calculations are straightforward, explaining why this partitioning is used in the model studied in this paper.

4. Creating the Polygonal Graph over the Work Space

Following the successful division of the work space it is easy to create the space "Occupancy Map", where the empty cubes represent the "Free Spaces" (*FS*). Based on this map, one can start with the obstacle-free route planning. A simple model of the above-mentioned method is displayed in Figure 6, here the drone is set to fly between the residential buildings.

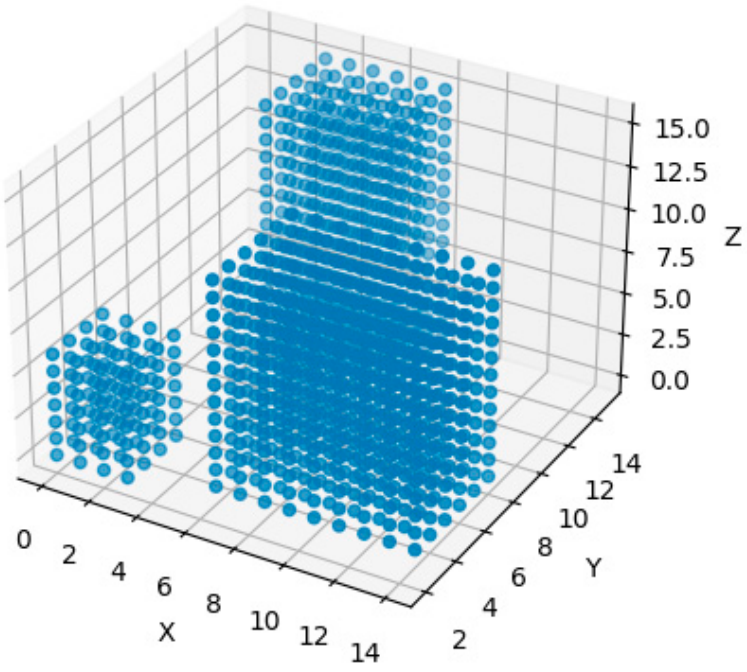


Figure 6. The scaled partitioned sample WS, with occupied blocks (blue dots) and Free Spaces (*FS*s).

The "graph-like", i.e., topological map of free environment is created by connecting the center-points of the neighbouring blocks in such a way that the connecting line cannot touch the occupied spaces (blocks marked in blue dots).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

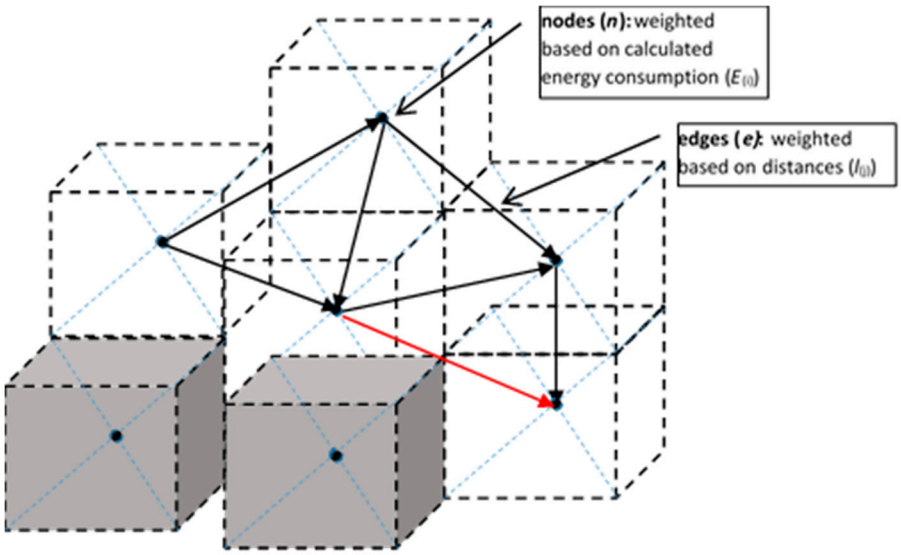


Figure 7. Creation of the graph-like, (topological) map of the space, where the center points of the FS blocks are connected. Where the connecting line crosses (or touches) the occupied block (see red line) the connection is not possible.

Figure 6 shows a scaled *15x15x15*-units WS, represented and partitioned by cube grid, where the Centre Points of the cubes are calculated and the occupied cubes are coloured blue. SW support is ensured by the *Python* package, with the different tools activated and suited for the requirements. Moreover, the Centre Points represent the nodes, while the connecting lines denote the edges of the graph. When the neighbouring centre points are connected, this results in 22 lines, 6 of which are the lateral adjacencies (shorter lines) and the rest are diagonal adjacencies (longer lines). The mathematical formula for length calculations is simply based on Pythagorean axiom: $length\ e_j = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$, here the indices <*i, j*> are the number of nodes resp. edges.

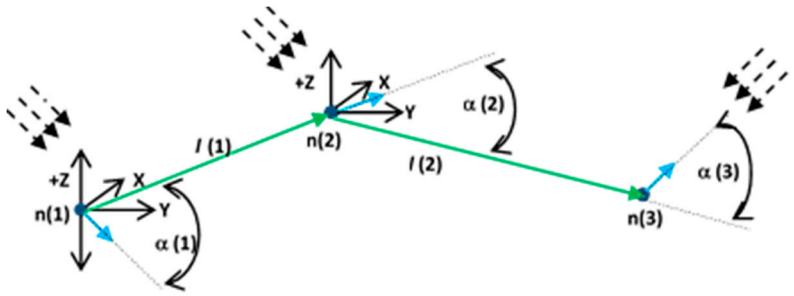
4.1. The Weighting Mechanism of the Graph, Calculations of the Weights of Nodes and Edges

Let the graph "*G*" be defined as follows: *G=(n,e)*, where *n(i)* – are the nodes of the graph, and *e(j)* - are the edges of the graph. The weighting of the graph is divided into two phases. In phase 1, the edges are weighted related to the length (*l(j)*) of the edge *e(j)*, then, in phase 2, the energy demand of the node (*E(i)*) is calculated using the previously defined mathematical expressions.

4.1.1. The Energy Demand Calculation of the Node

To evaluate the energy demand of the node, the following conditions are taken into account: the **orientation** of the drone (see: yaw/pitch/roll –equations of motion (6), (7)) [22]; **altitude** changing (see: rise/fall –equations of motion (5)); **air resistance** and **wind directions** (see: E_{move} – calculation, eq.: (11)). But basically in E_{motors} eq.: (10), all these calculations can be found. Consequently, the final energy demand of graph-node *(i)* can be calculated:

$$E_{(i)} = E_{motors} + f(yaw, pitch, roll) + f(rise, fall, hover) \quad (13)$$



 doi:10.20944/preprints202407.1465.v2

Figure 8. Illustration of weight calculations along the path in 3 different situations, see nodes n(i). The drone is represented as a point and orientation, see the blue full arrow line. The wind direction is represented with dashed arrow line.

Description of n(1): This is the node where the drone starts from, accelerates to the desired velocity, reaches the desired altitude (rise), and changes its orientation by an angle of $\alpha(1)$ (rotate around x-axis). After this, the drone will cover the distance $l(1)$ in the given orientation at the given altitude. The tailwind is considered in E_{motors} equation.

Description of n(2): In this node the drone maintains the altitude, it only has to change its orientation around z-axis by an angle of $\alpha(2)$. In E_{motors} equation a crosswind has to be considered. After this, the drone will cover the distance $l(2)$.

Description of n(3): In this node the drone has to change its altitude (drop to the given altitude), change the orientation by the angle $\alpha(3)$ around the z-axis, and the headwind (contra wind) is considered in E_{motors} equation.

4.1.2. The Final Weighting Calculation of the Path

There are many routes between the START and GOAL positions. Let the index of these paths be $k=\{1,\ldots,p\}$. Then the final weighting of the selected (k^{th}) path can be calculated as:

$$sumE_{(k)} = \sum_{i=1}^{n} E_{(i)} + \sum_{j=1}^{m} l_j \tag{14}$$

The final execution path can be chosen based on the minimum cost function ($C_{f(min)}$), where:

$$C_{fmin} = min(sumE_k) \tag{15}$$

5. Possible Graph-Search Procedures, Surveying and Comparison the Effectiveness of Different Optimal Path Planning Methods.

In mobile robot platforms there are several methods and procedures to achieve the desired optimal path between the given points in space. Before determining the most suitable algorithm, or algorithms, for this particular case, the methods should be classified to determine which, or what combinations, might be the best solution. It is essential to differentiate between path search algorithms and the trajectory optimization methods. In the case of path search, the routes are found between the START and GOAL points of the space (if such routes can be found). In most cases, this search yields the best possibilities, which means the shortest or the fastest path. In comparison, path search, optimization methods are usually performed by choosing some optimization goal (parameter) and the optimization algorithm is run in order to approximate this goal function. The graph below displays a possible classification (Figure 9). A detailed description of each method is provided below.

 doi:10.20944/preprints202407.1465.v2

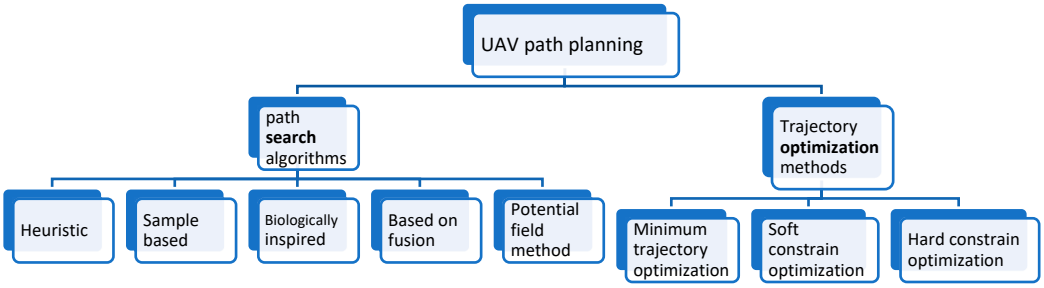


Figure 9. UAVs Path Planning Classification.

5.1. Path Search Algorithms

5.1.1. Heuristic-Based Algorithms

Heuristic-based path search algorithms are mostly based on graph search methods, such as: Dijsktra, A*, LifeTime Plan A* (LPA), dynamic A* (D*). These algorithms are fast and suitable for identifying the shortest path between the selected positions. As of 2019, these algorithms are extended for 3D environments, and improvements (mainly of A* algorithm) can be used for variable step size and variable weights. Summary: large amount of node calculation, poor RT performance, cannot be used in dynamic environment.

5.1.2. Sample-Based Algorithms

Two famous algorithms should be mentioned here, namely the Probabilistic RoadMap (PRM) and Rapidly-exploring Random Trees (RRT). RRT is fast, but unfortunately kinematic constrains avoidance and re-planning is not available. The path identification is fast, but it not necessarily optimal. This method has improvements such as RRT*, DDRRT, Anytime-RRT. PRM works with randomly scattered graph-nodes, which are connected to each other through some rules, and then processes a graph search in space. Summary: environmental pre-information is required, the optimal path is found randomly.

5.1.3. Potential Field (PF) Algorithms

These are typical algorithms for dynamic, or for multi-agent environments. This method usually does not find the optimal path, but it avoids moving obstacles and other agents in space. Summary: good for dynamic PPL, but can be trapped in local minima.

5.1.4. Biologically Inspired Algorithms

This is an imitation of biological behavior. During the process, the building of the complex environment model is saved and based on that, these algorithms propose a powerful search method. They usually find the optimal path, but the process is relatively slow. These algorithms can be divided into 2 groups: evolutionary algorithms (Ant colony, SWARM technology); and NNs. The disadvantage of these algorithms is that sometimes they suffer from the problem of premature convergence. To avoid this, the input parameters must be stated carefully, e.g. by pre-classification. Summary: dealing with unstructured constrains. Long application time, wide search range, slow.

 doi:10.20944/preprints202407.1465.v2

5.1.5. Algorithms Based on Fusion

These algorithms can be divided into 2 classes. The first class includes those that combine several PPL algorithms, integrated together, to work together to find the best path. The second class includes those that consist of several PPL algorithms, which are sequentially executed. When one algorithm completes its part, the second one starts to work immediately. As a sequential executing in a 3D environment the following sequence can be solved: 3D grid representation of the environment → using the 3D PRM (by this the obstacle-free roadmap is created) → A* algorithm to find the best path. Summary: in the case of well-chosen fusion of algorithms the best PPL can be created.

5.2. Trajectory Optimization Methods

Regarding optimization methods, the short characteristics of the techniques are:

5.2.1. Minimum Trajectory Optimization

Minimum trajectory optimization only ensures that the generated trajectory is smooth and dynamic, but does not constrain the trajectory itself and is suitable for unobstructed flight between two points. Summary: obstacle avoidance is problematic, but smooth trajectory dynamics are feasible.

5.2.2. Constrain Optimization

This increases the safety constraints on flight safety. This method is split into two main categories: hard constraints that increase the boundary value and soft constraints that increase the binding force. The hard constraint method considers all safe areas to be equivalent. Its disadvantage is that some parts of the trajectory come too close to the obstacle, and when the controller cannot follow the generated path, it may cause a collision and further, the flight speed is also low. The soft constraint method applies a "push force" (using the gradient method to calculate the proximity to the obstacle) to push the trajectory away from the obstacle. Summary: hard constrain – ensuring global optimality with the convex formulation. Soft constrain – pushing the trajectory far from obstacle (safety), if there is a local minimum, the success rate and the kinematic feasibility are low.

It must be emphasized that the list is far from complete, but this study will lead to further examination and discussions regarding which of these methods, or what combination of them proves most suitable for creating the energy-optimal path.

6. Calculation the Resulting Velocity and Energy Demand of the Drone

Figure 5 clearly shows that most of the energy is required by the driving motors of the drone's propellers, and the other consumption is almost negligible. The power of the driving motors is closely related (directly proportional) to drone speed.

$$\vec{P} = \sum \overrightarrow{F_{(i)}} * \vec{v}; \tag{16}$$

where, $\vec{P} = \frac{\sum \overrightarrow{E_l}}{t}$;

Since the forces and energies were calculated previously, see equations (1)-(12), now the velocities should be determined.

After obtaining the “wind map” of the environment, the resulting velocity vector can be calculated. As seen above (see eq. (16)), the velocity directly influences the drone’s energy consumption, which means determining the correct resulting velocity is vital. As indicated in eq. (12), $E_{kinetic}$ considered the wind conditions based on the following principles, seen Figure 10.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

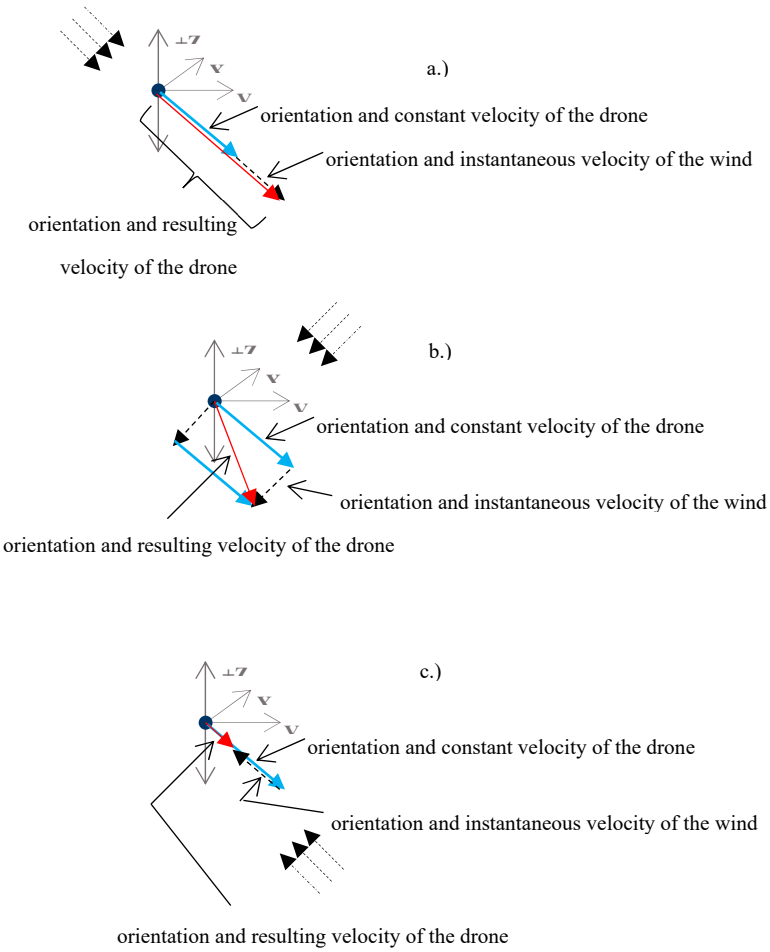


Figure 10. The resulting velocity calculation of the drone. (**a**) tailwind; (**b**) crosswind; (**c**) headwind. The wind direction is represented by the dashed-arrow lines; the resulting (calculated) velocity vector of the drone is represented by red full-arrow line; the initial velocity and orientation of the drone is represented by blue full-arrow line. .

Figure 10 a-c display that the resulting drone velocity is calculated from the signed *vector sum* of the wind speed and direction, as well as the average drone speed and direction (*note: in the case of scalar calculations, the resulting velocity, magnitude and direction, must be performed based on well-known trigonometric (cosine theorem) equations*).

$$\vec{v}_{resulting} = \vec{v}_{drone} + \vec{v}_{wind}$$

in scalar:

$$\left| v_{\substack{resulting \\ (amplitude)}} \right| = (v_{drone}^2 + v_{wind}^2 - 2v_{drone}.w_{wind}.cos\alpha\ )^{\frac{1}{2}}; \quad (17)$$

$$v_{\substack{resulting \\ (orientation)}}(\beta^0) = arccos\left(\frac{v_{wind}^2 - v_{drone}^2 + v_{result}^2}{2.v_{result}v_{wind}}\right);$$

Theoretically (if the drone velocity remains constant), in certain cases the drone may hover in one place. The algorithm avoids this situation by making the drone change its orientation in a direction with lower resistance, and then, after travelling some distance, it turns again towards the target position. This situation is illustrated in the figures bellow, see Figures 11–14.

By combining eq. (16) and the power – energy relation, the following equation can be obtained:

$$\frac{\vec{E}}{t} = \vec{F} * \vec{v} \quad \rightarrow \quad \vec{E} = \vec{F} * \vec{v} * t \quad (18)$$

This calculated energy is $\vec{E}_{kinetic}$, and must be considered, and added to the final calculation of the node's energy demand.

 doi:10.20944/preprints202407.1465.v2

7. Completing the Algorithm and Flowchart of the Process

The ultimate aim underlying this study, as indicated in the abstract, was to use a drone for small package delivery to locations not easy accessible (e.g., mountain rescues) and to disabled people living in cities. To achieve this goal, a number of initial conditions must be met.

For the sake of this project, a 10x10km^2 territory will be assumed somewhere in a mountainous area, with an altitude of 250m from the highest terrain point. This gives the WS origin, which has to be modelled. There are excellent SW tools capable of creating a 3D model of this environment based on the "Google map" (where the territory ought to be selected). The wind direction and wind strength sensors should be placed somewhere near (or directly within) this territory. These sensors provide data via the IoT to the "MeteoCloud" (a proxy server developed for Data Collection with Database and Data Management functions) which serves to create a so-called "Wind Map" of the environment.

The first step is to perform the **configuration of the WS**, which means that the obstacles need to be **increased by the radius** of the sphere around the drone (**modelling** the environment). The next step is WS **partitioning,** where the environment is divided into the cubes; then the occupied spaces (where the obstacles are) and free spaces are designated. In conjunction to this process, the centre points of the free-spaces (cubes) must be designated, and based on these, the distance calculations between the centre points (edges of the graph - $e_{(j)}$ -) can begin. By designating the ***START-GOAL*** **pair**, the **graph orientation** can be specified. The **resulting velocity** of the drone can be calculated by the drone represented as a point (where the orientation and the average speed of the drone are given) and the wind conditions (the resulting velocity can speed up, or slow down the average velocity of the drone, based on the wind direction and speed). The power can be calculated (eq. 16) from the resulting velocity, which is directly related to the battery energy (eq. 18), through the drone's energy consumption (eq. 9).

As a result, the flight time over the segments (edges) of the graph is acquired, which is directly connected to the resulting velocity of the drone. If the wind increases the average speed, the flight time decreases and the energy consumption is less. Eventually, the final execution path can be selected based on cost functions (eq. 15) of the calculated paths. The process described above, is repeated following each update of data from IoT "MeteoCloud".

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

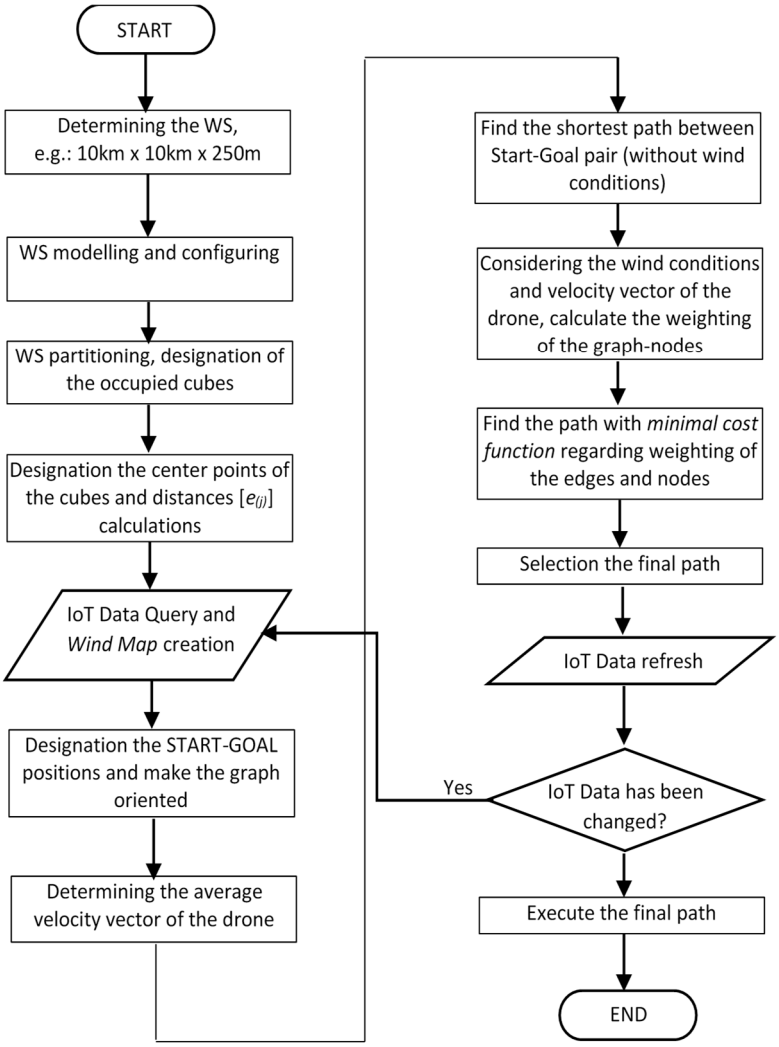


Figure 11. The flow-chart of the entire process.

8. Results and Analysis

In order to verify the theoretical results, the authors prepared the above-mentioned (outlined) work area and examined their related assumptions.

The fastest way to find the path with minimum energy demand of the drone is by the heuristic A* search method. In this particular case, the basic A* method is extended by the wind conditions, and the weighting of the edges and the nodes is introduced based on the energy demands of the path-segments. In the first step, the modelled work space map is embedded in the system, and then the graph search executed on the given environment.

In this project, the developed algorithm was tested for 3 different scenarios. In each situation the drone and the wind velocities are constant and the wind velocity $v_{wind}=0,75*v_{drone}$.

1. In Scenario 1, the tailwind is acting in the direction of the velocity vector of the drone. The energy consumption of the drone is minimal, the path faces the target position almost directly, see Figure 12. The starting position $S=[1,1,0]$ and the goal position coordinates $G=[11,13,10]$, $v_{wind}=0,75*v_{drone}$
2. In Scenario 2, the drone's velocity vector and the wind velocity vector were at 90 degrees. It is clear from the Figure 13, while the drone is flying between the houses, near to ground in leeward, it does not have much effect on the trajectory planning, but in an open field it does. The gliding effect can be partially observed. The wind vs. drone velocity ratio remains the same.
3. In Scenario 3 the total headwind affects the drone, the two velocity vectors are opposite to each other. The ratio is: $v_{vind}=0,75*v_{drone}$. In this instance two different starting points were considered. $S_L=[6,3,0]$ starting between the buildings (Figure 14, left), and $S_R=[6,0,0]$. The goal positions are

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

identical in both instances. The significant difference between the routes becomes apparent only in the open area, namely, the gliding effect is much more prominent, compared to the previous case.

The model's program is featured by object orientation, where first, the *class Map* is created and embedded into the search algorithm. It has a function called *free()* that checks if a point in 3D space is occupied or free. It also has a function *heuristic()* that calculates the absolute value of the distance between current and final points (x,y,z). It further calculates the weighting of the wind from given parameters. The A_*star_search()* function itself moves from point to point to check if the current point is the goal, until it actually reaches the final point. It uses the function *heursitic()* to find the optimal neighbour points. The visited nodes are saved in an array and can be represented visually. The previously described algorithm (Figure 11) considers three different paths under three different wind conditions, as illustrated in Figure 10. The final results of the above described path planning processes can be seen in the figures bellow, see Figures 12–15. The pseudo codes of the program are given in **Appendices A** and **B**.

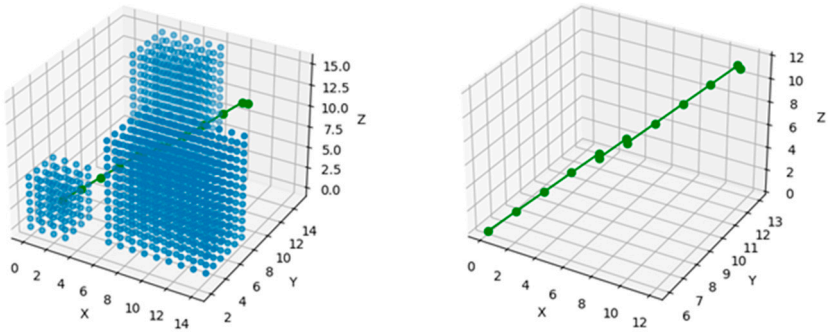


Figure 12. The calculated optimal trajectory from the START (left bottom corner) to the GOAL positions in a *tailwind* situation.

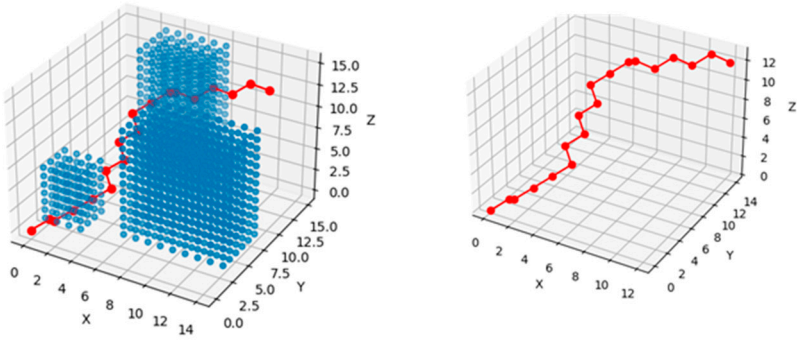


Figure 13. The calculated optimal trajectory from the START (left bottom corner) to the GOAL positions in a *crosswind* situation.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.1465.v2

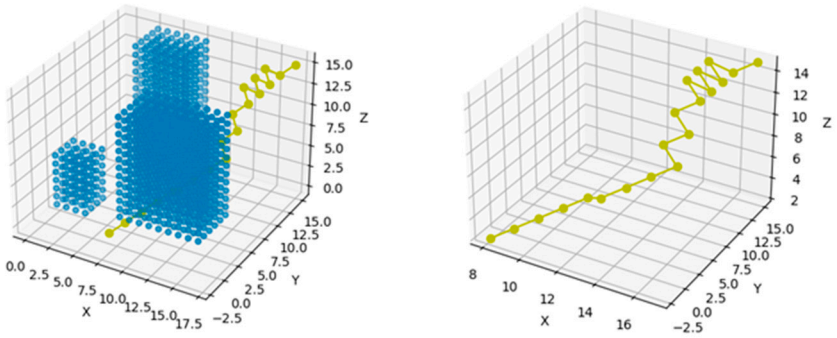


Figure 14. The calculated optimal trajectory from the START (left bottom corner) to the GOAL positions in a *headwind* situation.

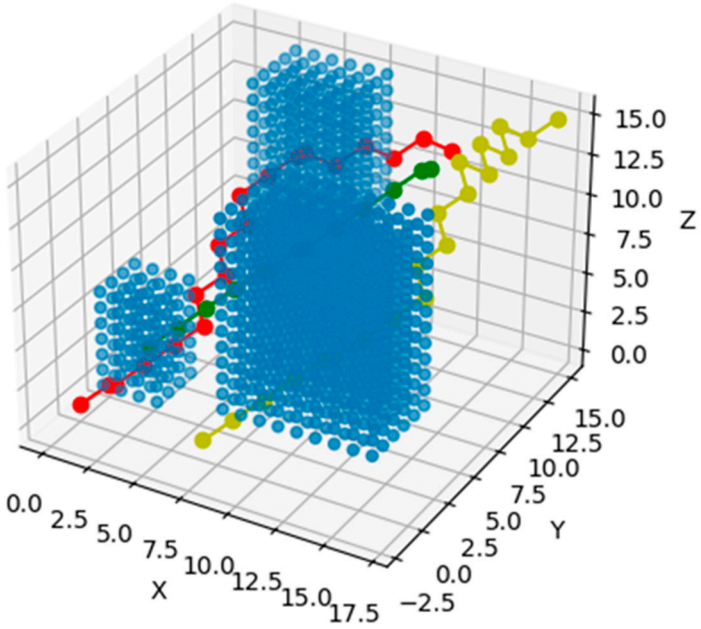


Figure 15. The sum of calculated optimal trajectories from the START (left bottom corner) to the GOAL positions.

9. Conclusion

In this work the authors developed an energy-optimal trajectory planning algorithm for an emergency drone. This energy-optimal path can also save lives in case of emergency calls or assistance needed, such as in mountain tourism, when a smaller medical package must be delivered to those in trouble. To achieve this, the authors analyzed several path planning algorithms, from which the following conclusions could be drawn.

Originally, this study focused on the PPL processes using evolutionary algorithms, such as NN, and GA-based PPL processes. Unfortunately, these methods were considerably limited in the case of dynamic weighting, because they were highly dependent on the training (or new population generation) time and the update frequency of the weights. Simply put, during training, the weights can change several times, and the path selected would not be the optimal and current one. Based on these studies, the dynamic and Real-Time PPL methods seemed to be most suitable, thus the heuristic A* algorithm was opted for.

 doi:10.20944/preprints202407.1465.v2

Essentially, a *modified A** algorithm was used, because in the classic algorithm the edges were weighted and the optimal path was identified based on a cost function. In this modification, not only the edges but also the nodes were weighted, and the algorithm stepped from one node to the next, selecting the branch to the next node based on the actual weight (which is calculated and evaluated) at each node. A good example can be seen in Figure 14 (see the yellow path), where the drone flew in *headwind* and "sailed" between the nodes to achieve minimum energy use. Basically, this is a highly complex system, therefore it was divided into sub-programs (procedures) and their operations were interconnected and/or interlinked. The system relied on two Databases (DBs): 1. "MeteoCloud" – where the IoT data of the meteo stations was uploaded. The wind map for the selected Working Space was created from this DB; 2. is the "PointCloud" of geographical data of the selected WS. After partitioning the WS, the points were represented by the centre-points of the cubes, and these points were also the nodes of the graph (of course, only in free-spaces). These points (nodes) were in fact vectors (arrays) containing coordinates and calculated weights, $\vec{n}_{(i)} = [x, y, z, w_{(i)}]$.

As the Figs. 12-14 reveal, the developed algorithm fulfilled the expectations of authors. In the case of tailwind (Figure 12) the trajectory was an almost straight line to the goal, while in the case of cross- and headwinds, the drone drew a trajectory resembling that of a glider aircraft.

10. Future Works

The outlined examples confirmed that the algorithms were working. However, the authors plan further developments, to make the algorithms work more efficiently. One of the issues was that a global path planning procedure was applied, which planned the entire path between the Start-Goal positions for the specified conditions. That means, if the wind direction/speed changed during the flight, optimal energy consumption would no longer be achievable. This calls for the application of some type of dynamic path planning model, or the possibly combination of this model with some kind of deep learning method. This line of reasoning leads to the path planning based on Markov decision-making mechanism, where the next step always depends only on the previous one. Regarding this problem, the authors studied reference [23], where the authors modelled a fixed wing UAV and worked in plane by the 3 DoF system. More inspiring theoretical ideas originated from [24], which offered no concrete, real solutions, but provided good theoretical proposals.

Summarizing the future work, firstly the authors will validate further existing models by specifying the existing models, secondly they aim to develop a new strategy based on a dynamic model by combining the Markov method and the parameterizable B-spline methods.

Author Contributions: Conceptualization, I.N. and E.L.; methodology, I.N.; software, I.N. and E.L.; formal analysis, I.N.; investigation, I.N and E.L.; resources, I.N..; writing—original draft preparation, I.N. and E.L.; writing—review and editing, I.N. and E.L.; supervision I.N. and E.L. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the Óbuda University Open Access Publication Support Foundation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to this is an ongoing research.

Acknowledgments: This work was supported by the Fuzzy Systems Scientific Group at the Bánki Donát Faculty of Mechanical and Safety Engineering of Óbuda University.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The pseudo code of the modified heuristic A* algorithm

Function is_free():

For obstacles:

 doi:10.20944/preprints202407.1465.v2


```
    if x >= obstacle[0] and x <= obstacle[1] and y >= obstacle[2] and y <= obstacle[3] and z >= obstacle[4] and
z <= obstacle[5]:
            return False
     return True
    Function heuristic(x1, y1, z1, x2, y2, z2, wind_speed, wind_direction):
            # Calculate the heuristic cost between two points
            distance = abs(x1 - x2) + abs(y1 - y2) + abs(z1 - z2)
            # Calculate the angle between the direction of the wind and the line connecting the two points
            wind_angle = arctan(y2 - y1, x2 - x1) - wind_direction
            # Calculate the component of the wind speed in the direction of the line connecting the two points
            wind_component = wind_speed * cos(wind_angle)
            # Return the total heuristic cost, taking into account the wind speed and direction
            return distance + wind_component
    Function a_star_search(map, start, goal, wind_speed, wind_direction):
        # Create a list to store the visited nodes
        visited = []
        # Create a list of the remaining nodes to be explored
        remaining = [start]
        # Keep looping until there are no more nodes to explore
        while remaining:
            # Sort the remaining nodes by their heuristic cost
            remaining.sort(key=lambda x: map.heuristic(x[0], x[1], x[2], goal[0], goal[1], goal[2],
wind_speed, wind_direction))
            # Get the node with the lowest heuristic cost
            current = remaining.pop(0)
            print(current)
            # Check if we have reached the goal
            if current == goal:
            # Return the list of visited nodes if the goal is reached
                return visited
        # Check the neighboring nodes of the current node
          for x in [current[0] - 1, current[0] + 1]:
              for y in [current[1] - 1, current[1] + 1]:
                  for z in [current[2] - 1, current[2] + 1]:
    # Check if the neighboring node is free of obstacles and has not been visited
                      if map.is_free(x, y, z) and (x, y, z) not in visited:
         # Add the node to the list of remaining nodes to be explored
                              remaining.append((x, y, z))
            # Add the current node to the list of visited nodes
            visited.append(current)
        # Return an empty list if the goal could not be reached
        return []
    # Define the start and goal coordinates
    start = (0, 0, 0)
    start2 = (0, 6, 0)
    start3 = (8, -2, 2)
    goal = (13, 15, 13)
    goal2 = (13, 13, 13)
    goal3 = (15, 15, 15)
    # Define the wind speed and direction
    wind_speed1 = 0
    wind_direction1 = math.pi / 2
```

 doi:10.20944/preprints202407.1465.v2


```
wind_speed2 = 2
wind_direction2 = math.pi / 3
# Perform the A* search, taking into account the wind speed and direction
visited = a_star_search(map, start, goal, wind_speed1, wind_direction1)
visited2 = a_star_search(map, start2, goal2, wind_speed1, wind_direction2)
visited3 = a_star_search(map, start3, goal3, wind_speed2, wind_direction2)
# Plot the visited nodes
ax.plot([x[0] for x in visited], [x[1] for x in visited], [x[2] for x in visited], "ro-")
ax.plot([x[0] for x in visited2], [x[1] for x in visited2], [x[2] for x in visited2], "go-")
ax.plot([x[0] for x in visited3], [x[1] for x in visited3], [x[2] for x in visited3], "yo-")
```

Appendix B

The generalized pseudo code of the whole process

 # Creating the configured area

read (envMAP);
nmax ← number of obstacles;
n=1;
while n ≠ nmax {
- *Dimension = Calculate the dimension of obstacle*
- *ConfigObstacle = Dimension + R*
- *n = n+1*

}
make the configured map of WS (confMAP);
fig (confMAP);

 # Partitioning the WS

read (confMAP);
select resolution (D)
For *x/y/z=0/0/0* To *xmax/ymax/zmax* Step *"D"*
- *make scaling*
- *print x/y/z*

next *x/y/z*
fig (partMAP);

 # Separating the free-spaces, creating the pointCloud (voxelMAP)

var
(voxelMAP) – array of n(i) vectors;
n(i)-vector (x,y,z,w(i));

read (partMAP);
- *Calculate the center points of cubes*
- *Select the center points in obstacles, sign "1"*
- *Select the center points on free-spaces, sign "0"*
- *Numbering the center points of free spaces n(i)*
- *Store the n(i) coordinates in cloud (voxelMAP) ← n(i)*

fig (voxelMAP);

 # Calculate the distances between n(i) nodes, make the weighting of edges

var
n(i) - vector (x,y,z,w(i));

read (voxelMAP)
Repeat

 doi:10.20944/preprints202407.1465.v2

$d_{(j)} = \sqrt{n^2_{(i+1)} - n^2_{(i)}}$; *{calculate the length of edges}*

w(j) ← d(j);
i=i+1;
Until i≠i$_{max}$;

 # Making the graph oriented

read (VoxelMap);
select START (x,y,z); j←1
select GOAL (x,y,z); j← j$_{goal}$

For *START(x,y,z)=1* To *GOAL(x,y,z)=j$_{goal}$* Step *"j"*
if j < j$_{goal}$ assign "→"
next *"j"*

 # Making the weighting of the nodes

var
WindData ← vector (x,y,z,wind$_{direction}$, wind$_{strength}$);

read (voxelMAP);

Repeat
read (MeteoCloud);
 # create WindData
WinData ← MeteoCloud
voxelMAP ← WindData
$\vec{v}_{resulting\,(i)} = \vec{v}_{drone(i)} + \vec{v}_{wind\,(i)}$
i=i+1;
Until i≠i$_{max}$;

 # Making the heuristic A* algorithm by the improved weighting

var
Open ← vector of nodes between START and GOAL positions
Close ← vector of nodes fixed by the lowest weighting

open ← all the nodes between START and GOAL positions

while open ≠ 0 {

- *calculation the cost function between the actual and next possible nodes*
  $f(n_{(i)}) = \underbrace{d_{(j)} + v_{(i)}}_{g(n(i))} + h(n)_{(i)}$;
- *comparison of the calculated cost functions* $f(n_{(i)}) \leftrightarrow f(n_{(i+1)})$;
- *selecting the smallest*
- *Close ← n$_{(i)}$*

plot d(j):[f(n$_{(i)}$)$_{small}$];
path← plot d(j)
i=i+1;
};
fig(path);

References

1. B. A. Gušavac, M. Martić, M. Popović, G. Savić. Agricultural Route Efficiencies, based on Data Envelopment Analysis (DEA). *Acta Polytechnica Hungarica* **2024** *Vol. 20, No. 10*, pp. 73-88.
2. Anatomy of A Drone - What's inside a DJI Phantom Drone, https://www.dronefly.com/the-anatomy-of-a-drone (accessed on 04. November 2022)
3. Drone design – Calculations and Assumptions, https://tytorobotics.com, (accessed on 22 October 2022)
4. Working Principle and Components of Drone, https://cfdflowengineering.com/working-principle-and-components-of-drone/ , (accessed on 22 October 2022)
5. G. Pandya. *Basics of Unmanned Aerial Vehicles: Time to start working on Drone Technology*, Publisher: Notion Press, CIT Colony, Mylapore, India, **2021**, ISBN 978—1-63745-387-2,
6. Abel J.P. Gomez; I. Voiculescu; J. Jorge; B. Wyvill; C. Galbraiht. Spatial Partitioning Methods. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*, Publisher: Springer, London, **2009**; pp. 187-225. https://doi.org/10.1007/978-1-84882-406-5_7.
7. F. Sabry. Exploring Binary Space Partitioning: Foundations and Applications in Computer Vision. *Binary Space Partitioning,* Publisher: One Billion Knowledgeable, **2024**.
8. Fabrice Jaillet, Claudio Lobos. Fast Quadtree/Octree adaptive meshing and re-meshing with linear mixed elements. *Engineering with Computers*, **2021**. https://doi.org/10.1007/s00366-021-01330-w.
9. MathWorks, octree - partitioning 3D points into spatial subvolumes, https://www.mathworks.com/matlabcentral/fileexchange/40732-octree-partitioning-3d-points-into-spatial-subvolumes, (accessed on 04. November 2022)
10. MathWorks, exportOccupancyMap3D, https://www.mathworks.com/help/nav/ref/exportoccupancymap3d.html, (accessed on 04. November 2022)
11. Sanches-Lopez, J. L.; Wang, M.; Olivares-Mendez, M.A.; Molina, M.; Voos, H. A Real-Time 3D Path Planning Solution for Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments, *Journal of Intelligent & Robotic Systems* https://doi.org/10.1007/s10846-018-0809-5, **2018**, *Vol. 93*, pp. 33-53.
12. Lifen, L.; Ruoxin, B. S.; Shuandao, C. L.; Jiang, D.W. Path planning for UAVS Based on Improved Artificial Potential Field Method through Changing the Repulsive Potential Function; *In Proceedings of the 2016 IEEE Chinese Guidance, Navigation and Control Conference*, **2016**, pp. 2011-2015, https:/doi.org/10.1109/CGNCC.2016.7829099.
13. Chen, X.; Zhang, J. The Three-dimension Path Planning of UAV Based on Improved Artificial Potential Field in Dynamic Environment; In *Proceedings of the 2013 Fifth International Conference on Intelligent Human-Machine Systems and Cybernetics*, **2013**, pp. 144-147, https://doi.org/10.1007/s10846-018-0809-5.
14. Boyu Zhou, Fei Gao, Jie Pan, and Shaojie S. Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths; In *Proceeding of the 2020 IEEE International Conference on Robotics and Automation (ICRA),* Paris, France, **2020**, pp. 1208-1214. https://doi.org/10.1109/ICRA40945.2020.9196996.
15. F. Samaniego, J. Sanchis, S. Garcia-Nieto, R. Simarro. Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs. MDPI, *Electronics* **2020**, *Vol. No. 1*, 51; https://doi.org/10.3390/electronics9010051
16. Sanches-Lopez J.L., Wang M., Olivares-Mendez, Miguel A., Molina, Martin, Voos, Holger. A Real-Time 3D Path Planning Solution for Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments *Journal of Intelligent & Robotic Systems*, **2019**, pp. 33-53. https://doi.org/10.1007/s10846-018-0809-5
17. Francis Colas, Srivatsa Mahesh, François Pomerleau, Ming Liu, Roland Siegwart. 3D path planning and execution for search and rescue ground robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Tokyo, Japan, 3-7. November 2013. pp. 722–727, 10.1109/IROS.2013.6696431 . hal-01143103, 2015.
18. Ahmed G., Sheltami T., Mahmoud A., Yasar A. Energy-Efficient UAVs Coverage Path Planning Approach. *CMES*, **2023**, *Vol.136, No.32022*, pp. 3239-3263. https://doi.org/10.32604/cmes.2023.022860
19. **EASA.** Drone Regulatory System, *Understanding European Drone Regulations and the Aviation Regulatory System, EU and MS regulatory governance*. https://www.easa.europa.eu/en/domains/drones-air-mobility/drones-air-mobility-landscape/Understanding-European-Drone-Regulations-and-the-Aviation-Regulatory-System (accessed on 16. jun 2024.)
20. Thomas Kirschstein. Comparison of energy demands of drone-based and ground-based parcel delivery services. *Elsevier, Transportation Research Part D,* ***2020****, Vol 78, 102209*. https://doi.org/10.1016/j.trd.2019.102209
21. U. C. Çabuk, M. Tosun, R. H. Jacobsen and O. Dagdeviren. A Holistic Energy Model for Drones. In *2020 28th Signal Processing and Communications Applications Conference (SIU)*, Gaziantep, Turkey, 5-7. October 2020, pp. 1-4. https://doi.org/10.1109/SIU49456.2020.9302218.
22. H. N. Al-sudany, B. Lantos. Extended Linear Regression and Interior Point Optimization for Identification of Model Parameters of Fixed Wing UAVs. *Acta Polytechnica Hungarica*, **2024**, Vol. 21, No. 6, pp. 69-88

23. W. H. Al-Sabban, L. F. Gonzales, R. N. Smith, Wind-energy Based Path Planning for Unmanned Aerial Vehicles Using Markov Decision Processes, In *2013 International Conference in Robotics and Automation, Karlsruhe,* Germany, 2013, pp. 784-789, doi.: 10.1109/ICRA.2013.6630662
24. Gugan, G., Haque, A., Path Planning for Autonomous Drones: Challenges and Future Directions. *Drones* **2023**, *7*, 169. https://doi.org/10.3390/drones7030169