Energy Optimal Flight Path Planning for Unmanned Aerial Vehicles in Urban Environments Based on a Novel Energy-Distance Map

H. Rienecker* and V. Hildebrand† and H. Pfifer‡
Chair of Flight Mechanics and Control, Technische Universität Dresden, Dresden, 01062, Germany

This paper introduces a comprehensive approach for calculating energy-efficient flight paths for unmanned aerial vehicles (UAVs) operating in urban environments. The primary objective is to minimize energy consumption by exploiting local wind phenomena, specifically targeting upwind and tailwind regions resulting from the airflow around buildings. The flight path planning algorithm uses a precalculated wind field to optimize the flight path. To achieve optimized flight trajectories, a customized A-star-Algorithm, enhanced with path smoothing techniques, is applied. A novel energy-distance map forms the base for the A-star heuristic function, which incorporates the key influential factors. The proposed approach is demonstrated using a benchmark scenario involving a delivery UAV, where energy-efficient flight paths are compared against the shortest way trajectories across 12 distinct scenarios. The results demonstrate significant energy savings potential when flying in urban areas by exploiting knowledge of the current wind conditions.

I. Introduction

The field of last-mile logistics in urban environments currently faces significant challenges. With the increase in traffic congestion and space usage, there is a growing need for novel approaches to address these issues. Specifically, the final step of the supply chain is often the least efficient. However, electrically powered unmanned aerial vehicles (UAVs) present a promising solution for improving last-mile logistics, even in remote areas, while minimizing environmental impact. The focus of this paper is motivated by the increasing adoption of delivery UAVs in urban environments for last-mile logistics. UAVs offer the potential to alleviate urban street traffic and reduce delivery times. Moreover, their automation capabilities and electric power source make them environmentally friendly. It is important to note that UAVs typically have lower payload capacities compared to ground-based vehicles [1]. Consequently, optimizing the efficiency of logistic UAVs becomes crucial. One effective approach is to optimize the flight path to achieve energy-efficient trajectories. Our previous research [2, 3] has demonstrated that by exploiting local wind conditions in urban environments, significant reductions in power consumption can be achieved.

This paper presents a novel approach for obtaining energy-efficient flight paths for typical delivery missions in an urban environment by using the knowledge of the wind field. The approach is holistic, consisting of several components. Firstly, in section II, we introduce a realistic city district representative of a typical European area, which was developed in our previous work [3]. This city district comes with a realistic wind field for the urban environment that was derived using a Parallelized Large-Eddy Simulation Model (PALM) [4]. The determined wind field was substantiated through validation conducted in a wind tunnel experiment [5]. Secondly, achieving an energy-optimal flight path across various wind conditions poses a classic route optimization problem. Numerous methods have been developed to find optimal routes, including Branch-and-Bound [6], evolutionary computing (e.g., [7]), multiple-agent systems [8], neural networks [9], and experience optimization, all of them can be applied in the embedded optimization process. Previous studies in UAV flight path planning and optimization have addressed various challenges, such as finding the shortest trajectory in scenarios with obstacles [10], avoiding hazardous weather conditions [11], navigating restricted airspaces [12], optimizing agricultural applications like fertilizer and pesticide spraying in specific crop fields [13], and addressing military objectives such as evading enemy radar sites [14]. Minimizing travel time in a hybrid routing and scheduling problem for UAV delivery systems has also been explored [15]. Reference [16] focused on the optimization of contemporary airline trajectories to minimize fuel consumption, taking into account constraints on flight paths in the

*Research Assistant, Chair of Flight Mechanics and Control, hannes.rienecker@tu-dresden.de
†Research Assistant, Chair of Flight Mechanics and Control, veit.hildebrand@tu-dresden.de
‡Professor, Chair of Flight Mechanics and Control, harald.pfifer@tu-dresden.de

horizontal plane. Many trajectory optimization methods, including [16, 17], apply the A-star algorithm or one of its variant, such as Theta-Star [11, 18]. The Branch-and-Bound method offers advantages in three-dimensional path finding [16] and computational efficiency [19].

Section III introduces a customized A-star algorithm designed to address the energy-efficient path planning challenges in urban environments. This algorithm takes into account the turning constraints of the UAV and incorporates smoothing techniques to generate a flyable trajectory directly during the optimization process. Path smoothing is achieved by applying piecewise polynomials that ensure a continuous trajectory within the limits of the UAV's flight envelope. Specifically, continuous cubic Bézier spiral segments, as derived in [20], are employed to satisfy the maximum curvature constraints of the UAV, see Section III.B.

Finally, in Section III.D, a novel diffusion-based method for generating an energy-distance map is presented, which is then used to derive a well-suited heuristic function for the A-star algorithm. While the A-star algorithm is commonly applied to minimize distance or travel time, various improvements such as Theta-Star [11, 18], ALT algorithms based on ***A**** *search,* ***L****andmarks, and the* ***T****riangle inequality* [21] have been developed. In contrast, the proposed approach incorporates both distance and upwind factors, as these factors have the most significant impact on the energy required of a lightweight UAV.

We apply a realistic city district connected with the generated wind field and apply the path optimization methodology, which includes flight trajectory modelling and faster search enabled by heuristics. The effectiveness of the proposed approach is demonstrated through 12 delivery tasks, where the energy-optimized paths are compared to the shortest routes in Section IV.

II. Problem Formulation

One common situation where a UAV is used for last-mile logistics involves delivering goods within a specific area of a city. This is done by flying the UAV from a pick-up point to a drop-off point, using a fully electric fixed wing aircraft. The scenarios aircraft is similar in size and features to the Phoenix Wings PWOne delivery UAV [22], which has a wingspan of 1.3 m and a maximum take-off weight of 3.4 kg. The UAV typically cruises at an average speed of 60 kph, and it's estimated to have a glide ratio of 20.

The generic city model and its wind field is adopted and validated from previous work [3, 5]. The model is representative of a typical European urban area and includes eight buildings of unique shapes and varying heights, which are depicted in Figure 1. Specifically, there are three residential buildings with a height of 50 m, four terraced houses with a height of 20 m, and a supermarket building with an attached office block with a height of 15 m. We designed this arrangement to obtain typical local wind effects found in those environments. By exploiting these wind fields, we can potentially decrease the energy consumption of an UAV during a delivery mission.

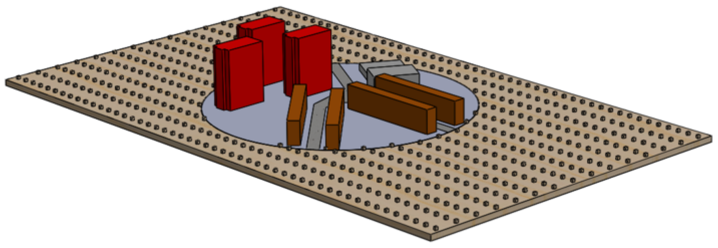


Fig. 1 Generic city model that represents a typical European urban area for simulating delivery tasks

This paper analyses 12 different test scenarios that involve delivering items under various wind speeds. Specifically, there are four delivery tasks that were performed in three different wind speeds for each task. Figure 2 shows these scenarios, with a single wind direction being used for all cases to remain within the influence zone of high-rise buildings. The delivery tasks involve flying from Point North to Point South, South to North, West to East, and East to West. The goal in each scenario is to reduce the energy required by the UAV by taking advantage of local wind effects. It is assumed that the aircraft flies at a constant true airspeed corresponding to its best-performance cruise speed. The start and end altitude is set to 20 meters. This altitude is realistic for air delivery in cities with takeoff and landing in multicopter-mode. The energy required for these procedures is not considered. The investigation is based on average wind speeds in a typical European city, with Dresden, Germany serving as an example. The wind speeds for the city, averaged over a year, are shown in Figure 3. Three characteristic wind speeds at a height of 10 meters were selected to represent the wind speed spectrum. These are the average wind speed for the windiest and calmest days and the average wind speed for the entire year. These speeds were used to create a wind profile shape, which was obtained from a wind tunnel experiment in [23]. Table 1 summarizes the freestream wind speed $u_{W\infty}$ for the wind profiles.

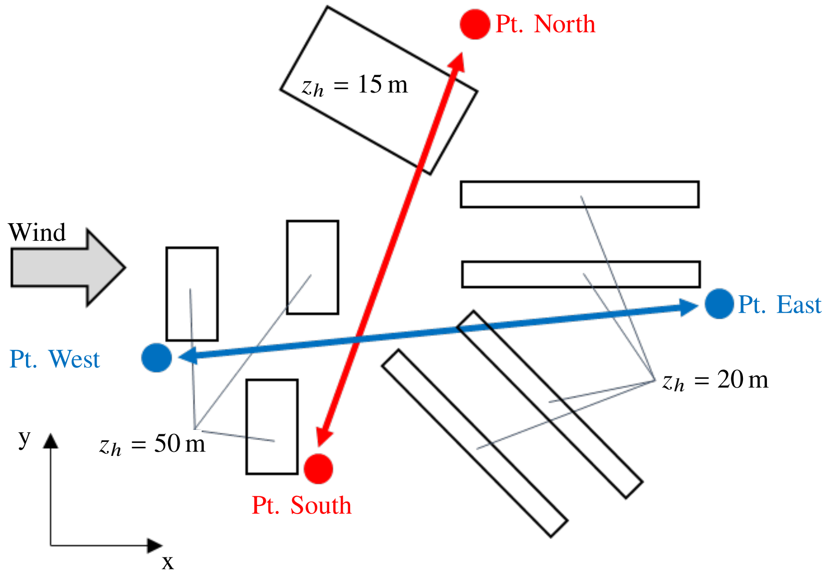


Fig. 2 Flight scenario with constant wind direction from west, as well as four tracks by flying from Point West to East, South to North and vice versa for each

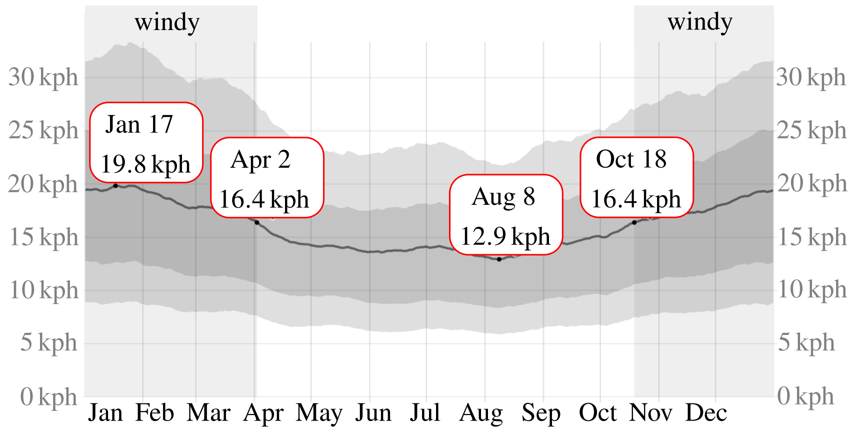


Fig. 3 Average of mean hourly wind speeds (dark gray line), with 25th to 75th and 10th to 90th percentile bands for Dresden, representing a typical continental European city [24]

Table 1 Summary of wind speeds applied in scenarios

Explanation	$u_W(z = 10\,\text{m})$ in kph	$u_W(z = 10\,\text{m})$ in m/s	$u_{W\infty}$ in m/s
Calmest day	12.9	3.6	6.5
Year-averaged day	16.4	4.6	8.3
Most windy day	19.8	5.5	9.9

The topography of the generic city model and the realistic wind field were applied to a large eddy simulation to obtain a realistic wind field within the urban environment for the following flight path optimization. A detailed description of the used software, the settings, and its validation can be found in [3].

III. Trajectory Optimization

The aim of planning the flight path is to minimize a cost function J,

$$J = \int_{s_0}^{s_f} A(X, s)\,\mathrm{ds}, \tag{1}$$

where X is the position vector of the aircraft, s_f the arrival point and $A(X, s)$ the quantity to be minimized at each step. For example, if the goal is to minimize the distance, then $A(X, s)$ is set to $A(X, s) = 1$. This is a complex optimization problem that is usually solved using a discrete grid instead of a continuous integral. There are several solvers available in the literature to minimize the cost function Eq. (1), including the Branch-and-Bound algorithm used in this paper [6] or evolutionary computing-based algorithms [7]. This chapter introduces a tailored version of the A-Star Algorithm. The basic *A-Star-Algorithm* is a popular variant of the Branch-and-Bound algorithms, which is adapted from the Dijkstra Algorithm [19]. The presented algorithm is modified to satisfy flight mechanic constraints and achieve an optimal smoothed flight trajectory. The chapter begins by presenting the applied discrete grid, followed by an explanation of the algorithm. The derivation of the cost function required to calculate the exact cost is then provided. Unlike basic A-Star Algorithms that use the distance as an estimate of the minimum cost from any vertex to the target, this modified version incorporates a novel approach to its heuristic cost function, which is elaborated at the end of the chapter.

A. Model Area Discretization

As a starting point in this paper, the vertices from large eddy simulation (LES) in [3] are used, namely 2.5 m equidistant grid points. In the path finding problem, an aircraft can fly from the current vertex to all adjacent vertices as illustrated in Fig. 4. This figure shows the 3D grid with the current vertex in black, the vertices on the same level in yellow, below them in green, and above them in red. However, an equidistant grid, as used for the wind field calculation, disregards any limits imposed by the UAV's performance. For instance, an equidistant grid would lead to a flight path angle demand of 45 degrees which well exceeds the UAVs maximum achievable climb speed. This requires a shrinking of the original grid in the vertical direction. The optimal gliding ratio G determines the vertical flight grid distance Δf_v as:

$$\Delta f_v = \frac{\Delta x}{G}\ , \tag{2}$$

where Δx can be substituted by Δy due to the equidistant grid of the LES. Eq. (2) assumes flying with the maximum glide ratio as the best flight condition for descent, even if the glide path differs because of the wind influence. Hence, the value of climb ratio is the same as the maximum glide ratio, where it is assumed that the electric propulsion is powerful enough and close to its optimum operation condition for the climbing flight. Given the specifications of the UAV $u_{UAV,\mathrm{TAS}} = 60\,\mathrm{kph}$ and $G = 20$, as well as LES grid $\Delta x = 2.5\,\mathrm{m}$, Δf_v is $\Delta f_v = \frac{\Delta x}{\Upsilon} = \frac{2.5m}{20} = 0.125\,\mathrm{m}$. Since this derived vertical grid is finer than the one, used in LES, wind components u_w, v_w, w_w are derived by linear interpolation. In addition to the flight path angle range, a minimum turn radius is considered later in the next chapter.

To achieve a finer heading spectrum in the horizontal direction, we implemented the idea of increasing connectivity [11]. This involves the evaluation of the nodes of adjacent connections to allow for greater directional possibilities. Figure 5 illustrates, how the expansion of a single node varies with the connectivity constant K, where more connections result in longer computational times. This figure simply demonstrates the connections in 2D for the purpose of simplification. For our implementation, we have used a value of $K = 3$ as a trade-off between the quality of headings covered and computational efficiency. Nevertheless, the wind field data of between it is taken into account for energy required determination.

To limit the number of nodes, we eliminate unnecessary points. Previous studies have shown that slight changes in altitude occurs in optimization as climbing requires much energy effort. Furthermore, regulations restrict the altitude of flights. Therefore, as we set the starting and ending points at the same level of $z_0 = 20\,\mathrm{m}$ to exclude take-off and landing procedures, we introduce a flight sector with allowable flight altitudes of $z = z_0 \pm 0.75\,\mathrm{m}$.

B. Extended A-Star-Algorithm

As previously mentioned, the *A-Star-Algorithm* is frequently used to optimize routes or trajectories from a starting to an ending point. To begin with, we will provide a brief explanation of the basic *A-Star-Algorithm*. Afterwards, a

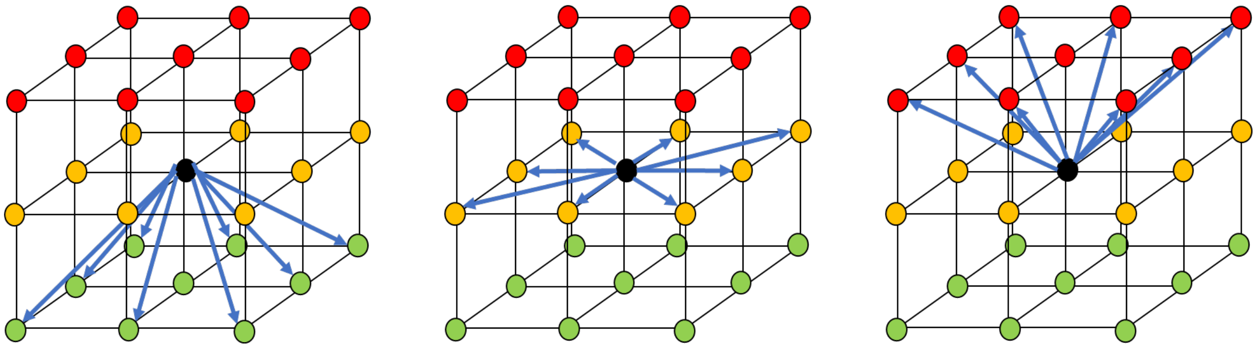


Fig. 4 The connection lines from a vertex (black) to all adjacent vertices (red, green and orange) define possible tracks (blue arrows) in 3D space

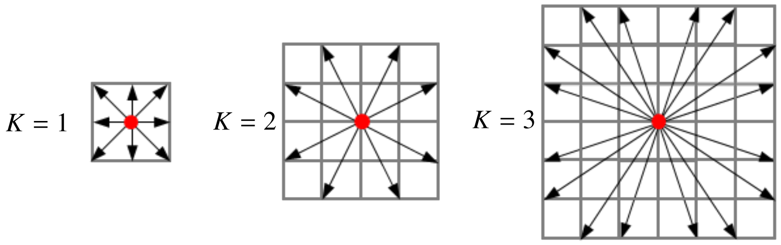


Fig. 5 The connectivity constant K determines the connections of a single node to allow for greater directional possibilities [11]

customized version is proposed.
In short, the *A-Star-Algorithm* commences at the starting point and determines the best path by selecting the optimal point at each branching. The best point is identified by the lowest total cost of the path, which is determined by adding up the exact cost $g(s)$ of the path from the starting point to the node s, and the heuristic estimated cost from node s to the ending point, represented by $h(s)$. Thus, the algorithm begins by analysing all possible paths in detail from the starting point by successively selecting the next node s with the lowest total cost $J = g(s) + h(s)$. As a result, a potential path is not examined if the branch is considered too expensive. This method saves computational time since not all possible paths have to be calculated. The exact steps of the basic *A-Star-Algorithm* are shown and explained in algorithm VI in the appendix. The main disadvantage of the basic algorithm in our application is that it only takes into account a single cost value between nodes, which limits the available paths between two nodes. This means that only one specific route can be taken, whereas there may be alternative trajectories that can be flown, such as a direct route or one with various lateral movements.
An extension of the basic algorithm enables to consider smoothed flight motions of the UAV. Hence, we examine three nodes at once to allow heading changes and check if the path respects a minimum turning radius r_{min} as a limit of the UAV. Therefore, a continuous-curvature path-smoothing algorithm based on cubic Bezier curves, including a maximum curvature constraint [20], is used. This method can be described on the basis of Fig. 6 as follows. Consider the section containing three points s_1, s_2 and s_3. Point s_2 becomes a curvature steering point and s_1, as well as s_3 get a continuous junction to the lines between the three points. The smoothing procedure is proclaimed in [20]. In short, a plane containing the three points yields to a 2D problem, and then eight control points are determined to create two cubic Bézier spiral curves between s_1 and s_3. One curve P is defined by the four control points B_0, B_1, B_2, B_3 and the function

$$P(m) = B_0(1-m)^3 + 3B_1(1-m)^2 m + 3B_2(1-m)m^2 + B_3 m^3 \ , \ m \in [0,1] \ . \tag{3}$$

The control points are then transformed back into 3D space. The minimum turning radius r_{min} of the UAV, which is

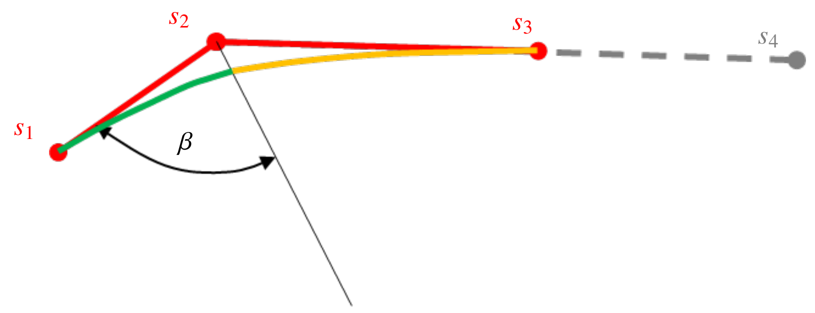


Fig. 6 Path smoothing between the three selected points s_1, s_2, and s_3: They get steering points and construct two cubic Bézier spiral curves, colored in green and orange

used during the second step for the curvature constraint calculation, is defined by

$$r_{min} = \frac{u^2_{UAV,TAS}}{g \cdot \tan(\arccos(\frac{1}{n_{max}}))}, \tag{4}$$

where $u_{UAV,TAS}$ is the true airspeed (TAS), g the gravitational acceleration, and n_{max} a maximum load factor of the UAV. Reference [25] shows that static load factors of UAVs do not exceed existing aviation regulations. Thus, a load factor of $n = 2.5$ is chosen for turning flights. The minimum turning radius of the UAV r_{min} determines the maximum curvature k_{max} of the flight path:

$$k_{max} = \frac{1}{r_{min}} \tag{5}$$

leads to a distance d_{min}

$$d_{min} = \frac{1.228 \cdot \sin\beta}{k_{max} \cdot \cos^2\beta}, \tag{6}$$

where β is the half angle between the two lines. This is done by comparing the distance d_{min} required for the UAV to turn with the lengths of the two lines $\overline{s_1s_2}$ and $\overline{s_2s_3}$. If one line is shorter than d_{min}, it is not possible to create a continuous path that fulfils the curvature requirement. In this case, the combination of the three points is excluded from further investigation. Afterwards, the path is verified to ensure it is obstacle-free. Finally, the cost of flying the path segment between the three points $J_{1,3}$ is calculated using the method outlined in the following section for exact cost, and it is added to the cost of flying to the first point s_1 to obtain the exact cost of flying to point s_3: $g(3) = J_{start,3} = J_{start,1} + J_{1,3}$. To ensure a continuous connection to the next three-point-segment, the midpoint s_4 of the segment is placed on the collinear line of $\overline{s_2s_3}$, see Fig. 6. As a consequence, each point s has different outgoing branches to leave s, depending on the three-point-segment before. Thus, we introduce an extended A-Star-Algorithm. Algorithm VII in the appendix shows and explains the differences to the basic one, where functions ensures the mentioned prespecified requirements. Furthermore, the basic variable of just one point-to-point connection become one that consist the point connections of the three points to consist the allowed path segment.

C. Cost Function for Exact Energy Identification

The objective of this paper is to minimize the energy supplied by the UAV's propulsion system during a flight with variable altitude between a fixed start and end point as mentioned in Section II. To achieve an optimal flight trajectory, a new cost function is proposed in this section. The first step is to define a generic flight path cost function (7) as an integral over the flight path s, which is then discretized in grid points sequentially flown through until the end point N.

$$J = \int_{s_0}^{s_e} A(s)\,\mathrm{ds} = \sum_{i=0}^{N} A_i\,\Delta s_i. \tag{7}$$

In the second step, a function for the energy required is derived to substitute the A_i in Eq. (7). Firstly, a 2D level flight is considered, and then the approach is extended to 3D space. All flight mechanical assumptions are based on standard literature, such as [26]. To simplify the calculations, the energy supplied by the propulsion system is assumed to be proportional to the thrust force multiplied by the distance covered by the UAV. For an UAV flying in steady cruise flight, it is assumed that the force generated by the propulsion system is equal to the aerodynamic drag force D. Based on this assumption, the energy required to fly between two points on the path is given by

$$E_i = D \cdot \Delta s_i = D \cdot u_{UAV,TAS} \cdot \Delta t_i \tag{8}$$

where $u_{UAV,TAS}$ is the true airspeed (TAS) of the UAV and Δt_i is the time required to cover the distance between two grid points Δs_i. Introducing Δt_i allows the incorporation of wind effects in the model. Ground speed at the i-th grid point is denoted as $u^*_{UAV,i}$, and the time required to travel Δs_i is expressed as

$$\Delta t_i = \frac{\Delta s_i}{u^*_{UAV,i}} \,. \tag{9}$$

With headwind $u_{W,i}$ and crosswind $v_{W,i}$ at the i-th grid point, the ground speed u^*_{UAV} of the UAV is related to its true airspeed by the following equation. u^*_{UAV} and true airspeed is given by

$$u^*_{UAV,i} = \sqrt{(u_{UAV,TAS} - u_{W,i})^2 - v^2_{W,i}} \,. \tag{10}$$

The equation for the energy required between two nodes in 2D can be obtained by using equations (8), (9), and (10):

$$E_i = D \cdot u_{UAV,TAS} \cdot \frac{\Delta s_i}{\sqrt{(u_{UAV,TAS} - u_{W,i})^2 - v^2_{W,i}}} \,. \tag{11}$$

In the next step of the analysis, the effect of up- and downwinds on level flight is taken into account. For instance, if there is an upwind component, the UAV needs to pitch down to maintain the same flight level. This means that the upwind component $w_{W,i}$ will cause a decrease in the flight path angle $\gamma_{i,wind}$:

$$\sin(\gamma_{i,wind}) = \frac{w_{W,i}}{u^*_{UAV,i}} \,. \tag{12}$$

When flying in an upwind condition, this decrease in flight path angle results in a reduction of the required thrust force to maintain steady level flight, since a component of the weight vector now supports the force in the direction of flight. As a result, at an upwind point, the perceived drag force that needs to be compensated by thrust force is reduced, and can be calculated as:

$$D_i = D_0 - \sin(\gamma_{i,wind}) \cdot mg \tag{13}$$

Here, D_0 is the drag force in normal horizontal flight condition and mg is the weight of the UAV.
In the 3D case, where changes in altitude are allowed, the flight path angle γ_i is composed of two components: the path component $\gamma_{i,path}$ and the wind component $\gamma_{i,wind}$, as given by $\gamma_i = \gamma_{i,path} + \gamma_{i,wind}$. The wind component is defined by Eq. (12). The altitude changing component is given by

$$\tan(\gamma_{i,path}) = \frac{\Delta s_{i,z}}{\Delta s_{i,xy}} \,, \tag{14}$$

where $\Delta s_{i,z}$ is the vertical distance and $\Delta s_{i,xy}$ the horizontal distance to the next point. Using the equation given in Eq.(11) to obtain

$$E_i = \frac{\Delta s_i}{\sqrt{(u_{UAV,TAS} - u_{W,i})^2 - v^2_{W,i}}} \cdot (D_0 - \sin(\gamma_i) \cdot mg) \cdot u_{UAV,TAS} \,. \tag{15}$$

The effect of vertical wind on ground speed $u^*_{UAV,i}$ is small compared to the other factors and can be negligible. The value of D_0 can be determined by the UAV's glide ratio G:

$$D_0 = \frac{L}{G} = \frac{mg}{G} \tag{16}$$

with L as lift force equal to the weight force and the glide ratio G of the fixed-wing UAV. In curve flight with roll angle μ, D_0 rises to D, defined by

$$D = \left(\frac{mg}{G} + k \cdot \frac{(mg)^2}{(\frac{\rho}{2} u_{UAV,TAS}^2 S)^2 \cdot cos^2 \mu} \right) \cdot \frac{\rho}{2} u_{UAV,TAS}^2 S \,, \tag{17}$$

where S is the wing surface and k the wing contour factor. Using Eq. (15), the cost function for the minimization of the total energy required can be written as:

$$E = \sum_{i=0}^{N} \frac{\Delta s_i}{\sqrt{(u_{UAV,TAS} - u_{W,i})^2 - v_{W,i}^2}} \cdot (D - \sin(\gamma_i) \cdot mg) \cdot u_{UAV,TAS} \,. \tag{18}$$

D. Heuristic Function for Energy Identification

The exact cost function $g(s)$ for the algorithm is defined in Eq. (18), as described in the previous section. The heuristic cost function $h(s)$ provides A-Star an estimate of the minimum cost from any vertex s to the goal. If $h(s)$ is closer to the exact cost of the path, A-Star can find the best way more quickly [27]. However, if $h(s)$ overestimates the exact cost, the algorithm may not find the best path. Therefore, the heuristic function should always be smaller than the exact cost, and both must be on the same scale.
In reference [3], $h(s)$ is defined based on the energy required to fly directly from point s to the final point, without the influence of wind. This approach requires a very small weighting factor $w = 0.001$ to account for favourable wind conditions that require less energy. Therefore, the performance of the computation is heavily influenced by the heuristic function. Hence, this paper proposes a new heuristic function that combines two ideas: Using diagonal distances as a good fit for minimizing distances, and distorting maps to represent region-specific data. Specifically, an energy-distance map is introduced. This section starts by introducing the cartogram method for distorting maps and deriving its driving variable. It then evaluates the quality of the heuristics and compares the novel method to the baseline proposed in [3].
Cartograms are maps that rescale geographic regions to visualize spatial statistics, see Fig. 8. The method behind them is derived in [28], which assigns an area in the map a density instead of a specific variable and allows the density to diffuse like in elementary physics. After density equalization, the specific area is placed at a new position. The driving process of diffusion is described by the current diffusion flux vector $\mathbf{J}$ with

$$\mathbf{J} = \mathbf{v}(\mathbf{r}, t) \cdot \rho(\mathbf{r}, t) \,, \tag{19}$$

where $\mathbf{v}(\mathbf{r}, t)$ is the velocity and $\rho(\mathbf{r}, t)$ the density at position $\mathbf{r}$ and time t. Diffusion follows the gradient of the density field from regions of high density to regions of low density:

$$\mathbf{J} = -\nabla \rho \,. \tag{20}$$

The diffusion process ends when the density becomes uniform everywhere after some time. The total displacement from start to finish determines the map distortion. The pure *cartogram* code from [28] is available as C source code [*].
The density function $\rho(\mathbf{r}, t)$ drives the diffusion process. Hence, a initial density function $\rho(\mathbf{r}, t = 0)$ is required. The density must be positive and should capture the main driving force behind the energy required to fly. Upwind conditions have a greater impact on exact costs than tailwind or crosswind using the derived equation Eq. (18). We established a linear function for density that decreases with increasing upwind at a single node. This reduces the energy-distance, which means that less energy is required to fly between nodes in upwind conditions. To ensure that the density is always positive, there is a minimum density value ρ_{min} at a specific upwind value. The specific upwind value $w_{w,max}$ is determined by the upwind threshold where the energy required would be negative. These constraints result in the following equation, which is illustrated in Fig.7.

$$\rho(\mathbf{r}, t = 0) = \frac{\Delta \rho}{\Delta w_w} \cdot \left(w_w(\mathbf{r}) - w_{w,max} \right) + \rho_{min} \,. \tag{21}$$

As mentioned earlier in Sec. III.A, only small changes in altitude occur. Therefore, in this paper, the diffusion process is limited to the altitude of $z = 20$ m. It is the level of the starting and ending points that makes the process 2D.

[*]http://websites.umich.edu/ mejn/cart/

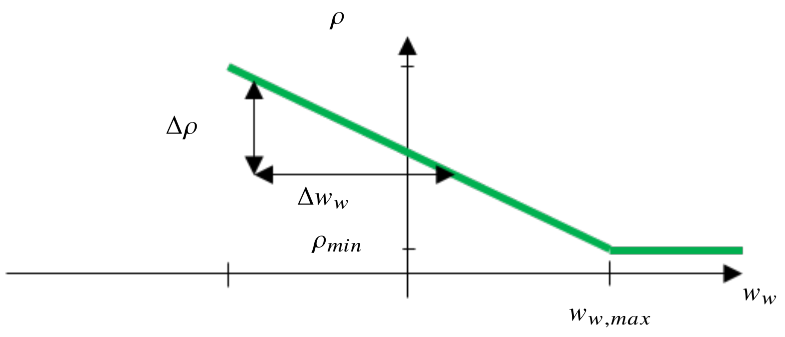


Fig. 7 Initial linear density function $\rho(\mathbf{r}, t = 0)$ at position r with minimum density value ρ_{min} if upwind $w_w(\mathbf{r})$ is greater than or equal to $w_{w,max}$

Higher and lower altitudes are represented by the position of the corresponding nodes $\mathbf{r} = (x, y, z) = (x, y, z = 20\,\text{m})$. This is valid because of the slight changes in the upwind in the z-direction.
Furthermore, the density of nodes inside buildings is set to $\rho(\mathbf{r}, t) = 0.1$, to facilitate the movement of adjacent nodes that have also low density and are forced to move. Additionally, we add a 25-node-wide grid at each map border with the same low density value to enable movement. To ensure no movement at the border, one row of nodes with the mean density value is added as a boundary condition.

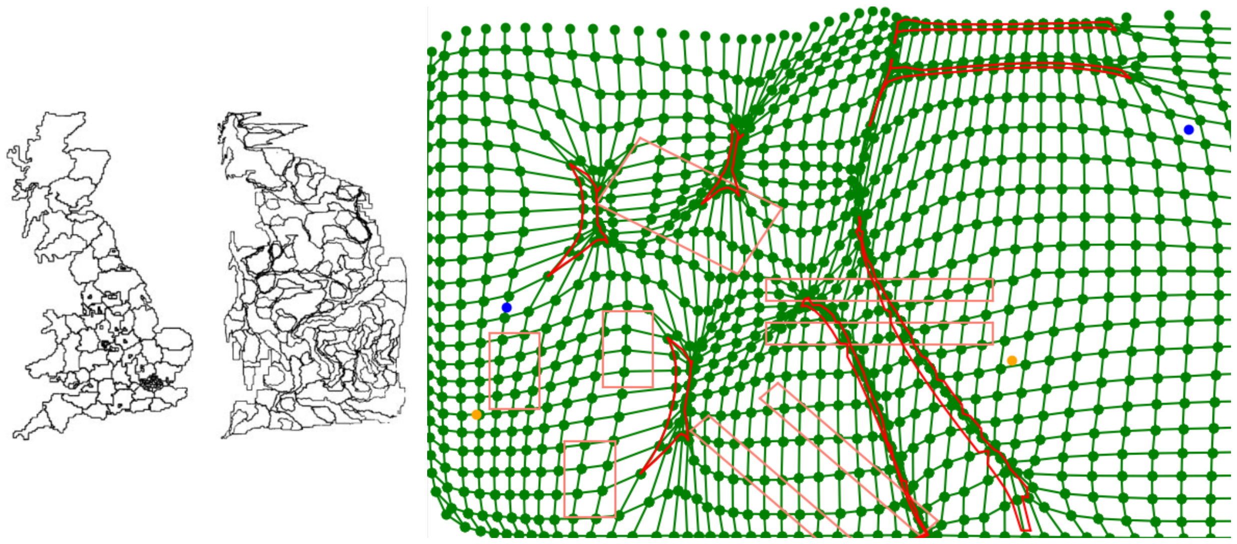


Fig. 8 Diffustion-based Cartograms with population cartogram of Britain by county on the left as an example from [28] and the obtained energy-distance map on the right with the grid (—●—), initial buildings (——), and distorted buildings (——)

After the diffusion process, each node from the basic x-y-coordinates described in Section III.A is assigned an energy-grid position $\mathbf{r} = (x, y)$, see Fig. 8. The heuristic energy required between two nodes s_i and s_j is determined by their energy-distance, given by

$$E_{h,ij} = \alpha \cdot \overline{\mathbf{r}_i \mathbf{r}_j}\ , \tag{22}$$

where α is a factor defined as follows. We examine every connection from each node to its adjacent vertex at constant flight level of start and target. By comparing the exact costs to fly Eg, ij from node s_i to s_j with the energy-distance $\overline{\mathbf{r}i\mathbf{r}j}$, we derive α through the mean value

$$\alpha = \frac{1}{N} \sum_{i=1}^{N} \frac{E_{g,ij}}{\overline{\mathbf{r}_i \mathbf{r}_j}}\ . \tag{23}$$

We obtain $\alpha \approx 1.0$ using this approach. For example, in the West-East scenario with a wind speed of $u_{W\infty} = 6.5$ m/s, the value of α is 1.3 and the values for $\overline{\mathbf{r}_i\mathbf{r}_j}$ vary between 0 and 43.9. The quality of the energy grid can be quantified by the standard deviation σ, which measures the amount of variation. The obtained standard deviation of $\sigma = 0.92$ indicates good quality, but leads to differences contrary to exact costs. Thus, it is necessary to include a weighting factor for heuristic costs in this case as well. An extensive parameter study has shown that $w = 0.7$ consistently results in the optimum path within an acceptable computation time (in this scenario).
Finally, we will now demonstrate the improvement achieved by the novel heuristic approach in detail, using a visualization of the optimized path for the scenario with wind speed $u_{W\infty} = 6.5m/s$ and an West-East track direction. Specifically, we will compare the heuristic costs along the optimized path to the exact costs from each point along the path to the target, expressed as a percentage of the total cost, depicted in Fig. 9. The black line shows the percentage exact cost, which are 100% by definition. The red line represents the approach described in [3], which is based on the heuristic function defined by

$$E_{h,ij} = \Delta s_i \cdot D_0 \ . \tag{24}$$

On the other hand, the green solid line shows the new heuristic approach based on Eq.22. It is evident from the graph that the new heuristic function leads to a significant improvement, with the heuristic costs being much closer to the exact costs compared to the previous approach. However, some nodes still have higher heuristic costs than exact costs, violating the constraint that the heuristic costs should always be smaller. Hence, it is necessary, to apply a weighting function that effectively lowers the cost beyond that threshold.

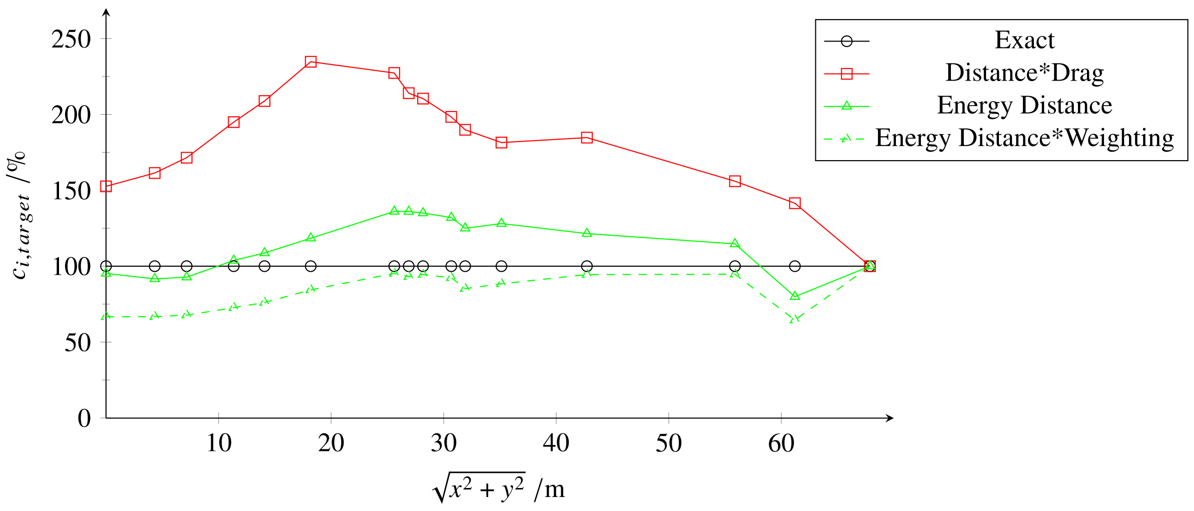


Fig. 9 Comparison between heuristic function values along optimized path for the scenario with wind speed $u_{W\infty} = 6.5m/s$ and the North-South track direction

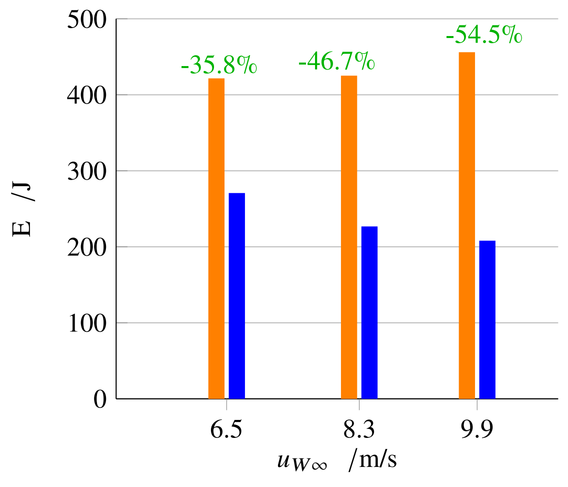
We found a weighting function that consistently results in the optimum path, described by

$$w = \begin{cases} 0.7, & \text{if } \frac{s_{Start,i}}{s_{Start,Target}} < 0.5 \\ 0.4 + 0.6 \cdot \frac{s_{Start,i}}{s_{Start,Target}}, & \text{otherwise} \end{cases} . \tag{25}$$

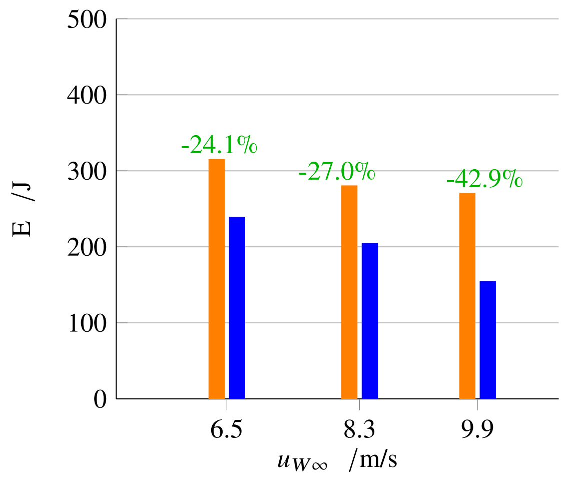
It takes into account the fact that the estimate becomes more accurate as a node gets closer to the target. The heuristic function including the weighting factor is shown in Fig. 9 as dashed green line.

IV. Results of Trajectory Optimization

In this section, we present the results of the flight path optimization using the cost function proposed in Section III.C and the heuristic in Section III.D. The optimization was performed for 12 different scenarios as specified in Section II, where each scenario involved one of the four track directions and a freestream wind speed of 6.5 m/s, 8.3 m/s, or 9.9 m/s. For each scenario, two optimal trajectories were computed, one using the energy optimal cost function of Eq.(18), and the other using a simple shortest path optimization described in Eq.(7). The A-Star-Algorithm was used for optimization in each case. The trajectory grid resolution was set to the adjusted LES grid from the wind field prediction as described in Section III.A.

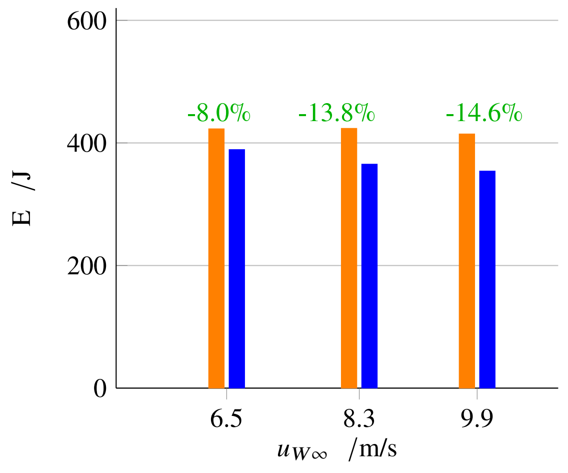


(a) ***North-South***

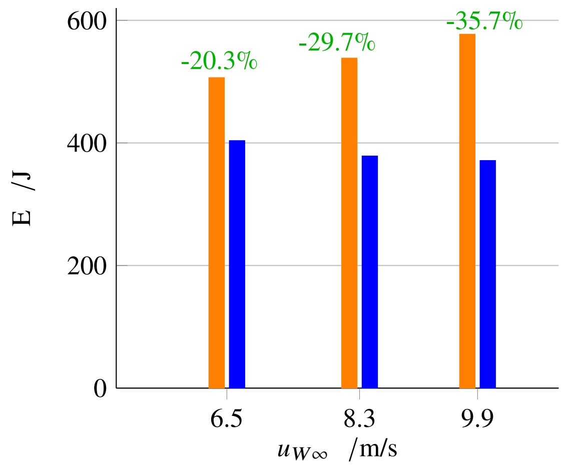


(b) ***South-North***

Fig. 10 Comparison between shortest-way-optimization (🟧) and energy-optimization (🟦) flight track in North-South-North scenarios



(a) ***West-East***



(b) ***East-West***

Fig. 11 Comparison between shortest-way-optimization (🟧) and energy-optimization (🟦) flight track in West-East-West scenarios

Figures 10 and 11 depict the results of the flight path optimization for all the scenarios considered in this study. The energy required for each scenario is quantified and presented in terms of energy savings in percentage. This percentage is calculated as

$$p = \frac{E_{\text{shortest way}} - E_{\text{energy opt.}}}{E_{\text{shortest way}}} \cdot 100\% , \tag{26}$$

where $E_{\text{shortest way}}$ and $E_{\text{energy opt.}}$ are the energy required for the shortest path and the energy-optimal path, respectively. The obtained results indicate that all the scenarios achieve significant energy reduction, with a clear trend of increased energy savings observed for higher wind speeds, as anticipated.

Figure 12 presents an example of an actual flight paths, where the orange line represents the shortest path and the blue line shows the path with the lowest energy required. Note that the energy savings achieved through optimization mainly result from exploiting regions of upwind not of tailwinds. This can be seen in Figure 13, which shows the upwind profiles along the trajectories. The energy-optimized trajectory passes through much larger upwind fields than the shortest path. According to Eq. (15), small differences in upwind have a greater impact on the required energy than small differences in tailwinds.

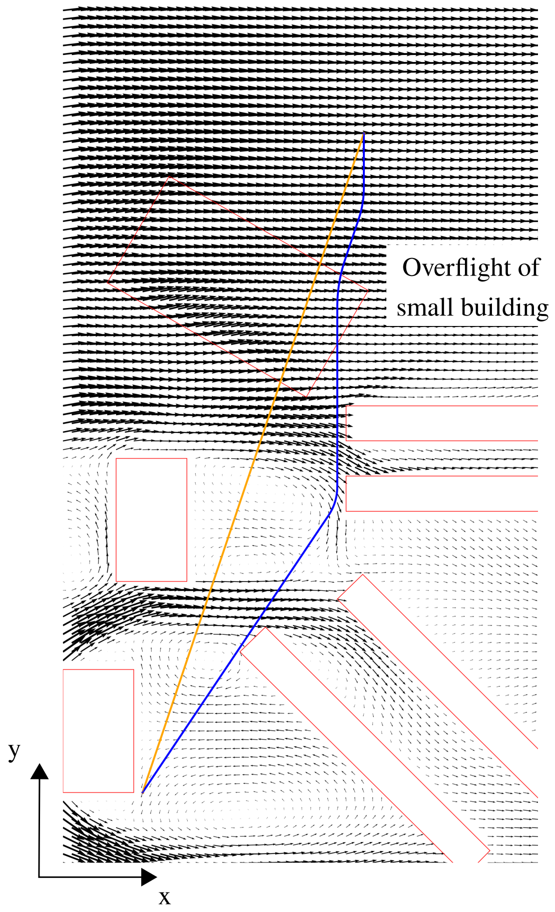


Fig. 12 Flight path for $u_{W\infty}$ = 6.5m/s with wind field in height of 20 m($\longrightarrow$), buildings (—), shortest way (—), and energy optimized path after A-Star-Algorithm (—), flying South to North

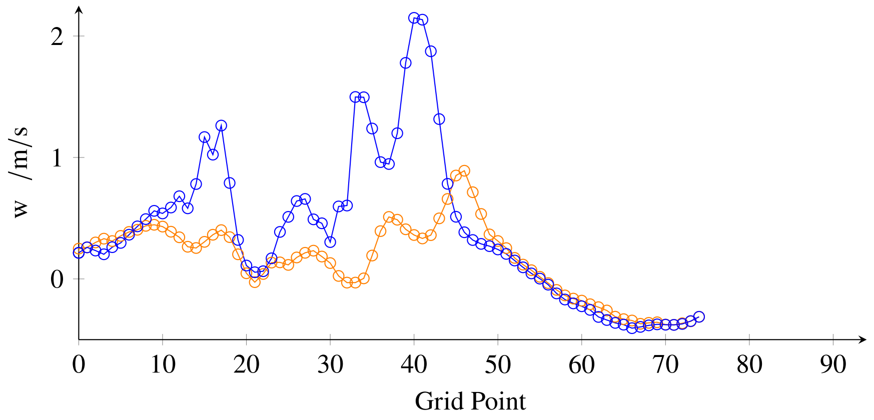


Fig. 13 Upwind for $u_{W\infty}$ = 6.5m/s, South to North, flying shortest way (—) and energy optimized path (—)

The flight paths obtained from the optimization process exhibit interesting characteristics that merit further investigation. Firstly, all energy optimized paths are observed to be situated near the rooftops of elongated buildings. This is due to the presence of strong upwinds in front of buildings, which result in lower energy consumption when flying alongside them with the wind perpendicular to the building's orientation. This phenomenon is akin to ridge lift, similar to gliding in mountainous regions. Consequently, North-South-North paths exhibit greater energy savings than West-East-West paths due to the greater opportunities to "slide" on the rooftops. Secondly, the altitude changes during the optimized flight paths are minimal. This is because the energy required to climb is significantly higher than that required to maintain a level flight. Conversely, descending does not sufficiently offset these costs. This altitude behaviour is seen in Fig. 14. Lastly, the energy optimized flight paths tend to be longer than the shortest path, as they exploit favorable upwind locations. However, the energy savings obtained compensate for the additional distance traveled. Table 2 provides a summary of the flight tracks corresponding to all the different scenarios.

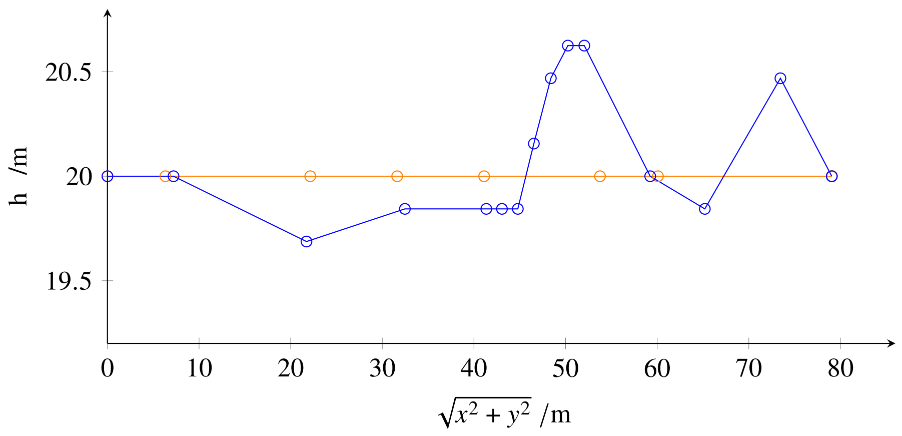


Fig. 14 Altitude for $u_{W\infty}$ = 6.5m/s, South to North, flying shortest way (—) and energy optimized path (—)

Table 2 Flight tracks

Route	$u_{W\infty}$	shortest way track in m	energy optimized track in m
North-South	6.5	197.6	2069.3
	8.3	197.6	206.5
	9.9	197.6	207.3
South-North	6.5	197.6	204.4
	8.3	197.6	204.1
	9.9	197.6	277.4
West-East	6.5	215.6	219.9
	8.3	215.6	220.1
	9.9	215.6	221.0
East-West	6.5	215.5	222.1
	8.3	215.5	221.9
	9.9	215.5	218.4

The newly proposed heuristic function has resulted in significantly shorter computation times. Table 3 presents the computation durations on a standard workstation PC for the previous method in Eq. 24 and the new heuristic function in Eq. 22. The longer computation time for the West-East track is due to the fact that the start-target node vector slope is not part of the connection slope, as mentioned in Section III.A. Conversely, for the South-North track, the start-target node vector slope is part of the connection slope.

Table 3 Computation durations

Route and type	$u_{W\infty}$ in m/s	Duration in s
West-East, former heuristic related to Eq. 24	6.5	172800
West-East, energy-distance map heuristic related to Eq. 22	6.5	85778
South-North, energy-distance map heuristic related to Eq. 22	6.5	2797

V. Conclusion

We introduced an approach for optimizing the energy required of 3D flight trajectories for delivery UAVs. The methodology applied an existing realistic city model including the wind field to facilitate the path optimization process. Our results demonstrated the effectiveness of the customized A-Star-Algorithm in optimizing flight paths. The main result is that the consistent benefit of optimizing flight paths at various wind speeds, leading to a significant reduction in energy consumption. Notably, our approach differed from previous work by implementing smoothing techniques during the optimization process and apply a heuristic A-star function that considered the primary influential factors. These enhancements contributed to the improved quality of the results obtained.
In the future, greater emphasis will be given on addressing the constraints associated with the initial energy-distance map in order to enhance its quality by reducing the standard deviation. Furthermore, it is planned to incorporate turbulent wind field data, as opposed to relying solely on time-averaged wind data.

Appendix

VI. Basic A-Star-Algorithm

The algorithm is structured in a main loop (line 1-29). At each step, it picks the node s out of the node list *openPoints* (line 9-11, at the beginning the starting point) with the smallest cost and processes that node. This cost is defined by summing up the exact cost of the path from the starting point to the node s, $g(s)$, and the heuristic estimated cost from node s to the ending point described by $h(s)$. After that, all connected neighbour nodes of node s are inserted in the node list *openPoints*, if they were not attended before (line 15-26). This loop of the algorithm ends the checking for a path if the ending point is reached (line 12).

Algorithm 1 Basic A-Star-Algorithm

```
Input: Neighbour list of points $neighbourlist = [p_0 : (p_{xxx}, p_{xxy}, ...), ..., p_n : (...)]$, cost list from point to neighbour
       cost($p_i, p_j$), heuristic cost list from point to destination $h(p_i)$
Output: A-Star-Path $path = [s_0, ..., s_n]$
 1: function AStar()
 2:     $g(s_{start}) := 0$
 3:     $parent(s_{start}) := s_{start}$
 4:     $openPoints := \emptyset$
 5:     $openPoints.add(s_{start})$
 6:     $closedPoints := \emptyset$
 7:     while $open \neq \emptyset$ do
 8:         $s := None$
 9:         for $v \in openPoints$ do
10:             if $s = None$ or $g(v) + h(v) < g(s) + h(s)$ then
11:                 $s := v$
12:         if $s = s_{goal}$ then
13:             $path$ =ReconstPath()
14:             return "Path found:", $path$
15:         for all $s' \in$ neighbours($s$) do
16:             if $s' \notin openPoints$ and $s' \notin closedPoints$ then
```

```
17:                 $openPoints.add(s')$
18:                 $parent(s') := s$
19:                 $g(s') := g(s)$+**cost**$(s, s')$
20:             **else**
21:                 **if** $g(s') > g(s)$+**cost**$(s, s')$ **then**
22:                     $g(s') := g(s)$+**cost**$(s, s')$
23:                     $parent(s') := s$
24:                     **if** $s' \in closedPoints$ **then**
25:                         $closedPoints.remove(s')$
26:                         $openPoints.add(s')$
27:         $openPoints.remove(s)$
28:         $closedPoints.add(s)$
29:     **return** "Path does not exist!"
```

VII. Extended A-Star-Algorithm

At each three-point-segment $s_1s_2s_3$, the information about all three points have to be available to compute the angle between the two connecting lines. This angle is equivalent to the heading change of the UAV. As a consequence, each point list in the algorithm has to contain these points for three reasons. Firstly, this information is necessary for checking the heading change constraint in allocation of the neighbour points (Alg. VI, line 15). Line s_1s_2 has to be collinear with line $s_2's_3'$ from the next three-point-segment $s_1's_2's_3'$. Secondly, it is required during the reconstruction of the path at the end (Alg. VI, line 13). The *parent*-list consists the analysed connection between two three-point-segments and the reconstruction algorithm collects these connections and put them to one path together. It is possible that this procedure contains two paths. These can be one desired path and one path with a connection that violates the constraint. Thirdly, the algorithm suspends points that were analysed (Alg. VI, line 28). However, this point has to be analysed again, if the point is reached from another direction. Thus, the extended A-Star-Algorithm contains the function angleReq() that ensures the prespecified requirements of the connecting lines between three-point-segment (line 17). Furthermore, it contains function bezier() to construct the cubic Bézier spiral curves between these points, crashcheck() for obstacle avoidance on this curves, and cost() to calculate the exact cost, the energy required, for this flight trajectory. Moreover, the point lists in the basic algorithm become lists that consist the point connections ($s_1s_2s_3$) to prevent the mentioned shortcomings. The function neighbours() includes the mentioned connectivity constant and ensures the collinearity between the two segments. The modifications made to the basic A-Star algorithm are highlighted in red text color, indicating the differences of the extended A-Star algorithm.

Algorithm 2 Extended A-Star-Algorithm

```
Input: Neighbour list of points $neighbourlist = [p_0 : (p_{xxx}, p_{xxy}, ...), ..., p_n : (...)]$, cost list from point to neighbour
       **cost**$(p_i, p_j)$, heuristic cost list from point to destination $h(p_i)$
Output: A-Star-Path $path = [s_0, ..., s_n]$
 1: **function** ASTAR()
 2:     $g(s_{start}\&s_{start}\&s_{start}) := 0$
 3:     $parent(s_{start}\&s_{start}\&s_{start}) := s_{start}\&s_{start}\&s_{start}$
 4:     $openPoints := \emptyset$
 5:     $openPoints.add(s_{start}\&s_{start}\&s_{start})$
 6:     $closedPoints := \emptyset$
 7:     **while** $open \neq \emptyset$ **do**
 8:         $n := None$
 9:         **for** $v \in openPoints$ **do**
10:             **if** $s = None$ **or** $g(v) + h(v[2]) < g(n) + h(n[2])$ **then**
11:                 $n := v$
12:         $s_1\&s_2\&s_3 := n$
13:         **if** $s_3 = s_{goal}$ **then**
14:             $path$ =**ReconstPath**()
15:             **return** "Path found:", $path$
```

```
16:        for all $s'_1\&s'_2\&s'_3 \in$ neighbours$(s_1\&s_2\&s_3)$ do
17:            $n' := s'_1\&s'_2\&s'_3$
18:            $rightDirectionBoolean$ = angleReq$(n')$
19:            $bezierBoolean$ = bezier$(n')$
20:            $crashcheckBoolean$ = crashcheck$(n')$
21:            $Boolean = rightDirectionBoolean \wedge bezierBoolean \wedge crashcheckBoolean$
22:            if $n' \notin openPoints$ and $n' \notin closedPoints$ and $Boolean$ then
23:                $openPoints.add(n')$
24:                $parent(n') := n$                                ▷ Note: $n = s_1\&s_2\&s_3$
25:                $g(n') := g(n)$+cost$(s'_1, s'_2, s'_3)$
26:            else
27:                if $g(n') > g(n)$+cost$(s'_1, s'_2, s'_3)$ and $Boolean$ then
28:                    $g(n') := g(n)$+cost$(s'_1, s'_2, s'_3)$
29:                    $parent(n') := n$
30:                    if $n' \in closedPoints$ then
31:                        $closedPoints.remove(n')$
32:                        $openPoints.add(n')$
33:        $openPoints.remove(n)$
34:        $closedPoints.add(n)$
35:    return "Path does not exist!"
```

Acknowledgments

This research was funded by the German Federal Ministry for Economic Affairs and Climate Action under grant number 20D2106C. The responsibility for the content of this paper is with its authors. The financial support is gratefully acknowledged.

References

[1] Kirschstein, T., "Comparison of energy demands of drone-based and ground-based parcel delivery services," *Transportation Research Part D: Transport and Environment* ***78**, Article 102209*, 2020. doi:10.1016/j.trd.2019.102209.

[2] Rienecker, H., Hildebrand, V., and Pfifer, H., "Energy optimal flight path planing for unmanned aerial vehicle in urban environments," *Proceedings of the 2022 CEAS EuroGNC conference*, CEAS, Berlin, Germany, 2022. CEAS-GNC-2022-031.

[3] Rienecker, H., Hildebrand, V., and Pfifer, H., "Energy optimal 3D flight path planning for unmanned aerial vehicle in urban environments," *CEAS Aeronautical Journal*, 2023. Accepted.

[4] Maronga, B., Gryschka, M., Heinze, R., Hoffmann, F., Kanani-Sühring, F., Keck, M., Ketelsen, K., Letzel, M. O., Sühring, M., and Raasch, S., "The Parallelized Large-Eddy Simulation Model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, recent developments, and future perspectives," *Geoscientific Model Development*, Vol. 8, No. 8, 2015, pp. 2515–2551. doi:10.5194/gmd-8-2515-2015.

[5] Frey, J., Rienecker, H., Schubert, S., Hildebrand, V., and Pfifer, H., "Wind Tunnel Measurement of the Urban Wind Field for Flight Path Planning of Unmanned Aerial Vehicles," *Proceedings of the 2024 AIAA SciTech conference*, AIAA, Orlando, USA, 2024.

[6] Land, A. H., and Doig, A. G., "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, Vol. 28, No. 3, 1960, pp. 497–520. doi:10.2307/1910129.

[7] de Camargo, J. T. F., de Camargo, E. A. F., Veraszto, E. V., Barreto, G., Cândido, J., and Aceti, P. A. Z., "Route planning by evolutionary computing: an approach based on genetic algorithms," *Procedia Computer Science*, Vol. 149, 2019, pp. 71–79. doi:10.1016/j.procs.2019.01.109.

[8] Biswas, S., Anavatti, S. G., and Garratt, M. A., "Particle swarm optimization based co-operative task assignment and path planning for multi-agent system," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, New York City, United States, 2017, pp. 1–6. doi:10.1109/ssci.2017.8280872.

[9] Yu, J., Su, Y., and Liao, Y., “The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning,” *Frontiers in Neurorobotics*, Vol. 14, 2020. doi:10.3389/fnbot.2020.00063.

[10] Tang, L., Wang, H., Li, P., and Wang, Y., “Real-time Trajectory Generation for Quadrotors using B-spline based Non-uniform Kinodynamic Search,” *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, New York City, United States, 2019, pp. 1133–1138. doi:10.1109/robio49542.2019.8961485.

[11] Garcia, M., Viguria, A., and Ollero, A., “Dynamic Graph-Search Algorithm for Global Path Planning in Presence of Hazardous Weather,” *Journal of Intelligent & Robotic Systems*, Vol. 69, 2012, p. 285–295. doi:10.1007/s10846-012-9704-7.

[12] Babel, L., “Flight path planning for unmanned aerial vehicles with landmark-based visual navigation,” *Robotics and Autonomous Systems*, Vol. 62, No. 2, 2014, pp. 142–150. doi:10.1016/j.robot.2013.11.004.

[13] Srivastava, K., Pandey, P. C., and Sharma, J. K., “An Approach for Route Optimization in Applications of Precision Agriculture Using UAVs,” *Drones **4**(3), Article 58*, 2020. doi:10.3390/drones4030058.

[14] Bortoff, S., “Path planning for UAVs,” *Proceedings of the 2000 American Control Conference*, IEEE, New York City, United States, 2000, pp. 364 – 368 vol.1. doi:10.1109/acc.2000.878915.

[15] Sajid, M., Mittal, H., Pare, S., and Prasad, M., “Routing and scheduling optimization for UAV assisted delivery system: A hybrid approach,” *Applied Soft Computing **126**, Article 109225*, 2022. doi:10.1016/j.asoc.2022.109225.

[16] Rosenow, J., Lindner, M., and Fricke, H., “Assessment of air traffic networks considering multi-criteria targets in network and trajectory optimization,” *Deutscher Luft- und Raumfahrtkongress 2015*, DGLR, Rostock, Germany, 2015.

[17] Junwei, Z., and Jianjun, Z., “Path Planning of Multi-UAVs Concealment Attack Based on New A Method,” *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, IEEE, New York City, United States, 2014, pp. 401–404. doi:10.1109/ihmsc.2014.198.

[18] Mandloi, D., Arya, R., and Verma, A. K., “Unmanned aerial vehicle path planning based on A$*$ algorithm and its variants in 3d environment,” *International Journal of System Assurance Engineering and Management*, Vol. 12, No. 5, 2021, pp. 990–1000. doi:10.1007/s13198-021-01186-9.

[19] Hart, P., Nilsson, N., and Raphael, B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107. doi:10.1109/tssc.1968.300136.

[20] Yang, K., and Sukkarieh, S., “3D smooth path planning for a UAV in cluttered natural environments,” *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, New York City, United States, 2008, pp. 794–800. doi:10.1109/iros.2008.4650637.

[21] Goldberg, A., and Harrelson, C., “Computing the shortest path: A* search meets graph theory,” *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003. doi:10.1145/1070432.1070455.

[22] “PWOne,” Phoenix-Wings GmbH, 2022. URL `https://phoenix-wings.de/pwone/`, accessed 2021-10-18.

[23] “VDI 3783 Part 12, Environmental meteorology: Physical modelling of flow and dispersion processes in the atmospheric boundary layer, Application of wind tunnels,” , Jul. 2022. URL `http://www.vdi.de/3783-12`.

[24] “Climate and Average Weather Year Round in Dresden Germany,” Cedar Lake Ventures, Inc., 2022. URL `https://weatherspark.com/y/75895/Average-Weather-in-Dresden-Germany-Year-Round`, accessed 2022-07-25.

[25] Majka, A., “Flight Loads of Mini UAV,” *Solid State Phenomena*, Vol. 198, 2013, pp. 194–199. doi:10.4028/www.scientific.net/ssp.198.194.

[26] McClamroch, N. H., *Steady Aircraft Flight and Performance*, Princeton University Press, Princeton, 2011. ISBN: 9781680159097.

[27] “Pathfinding with A$*$,” Python Pool, 1997-2023. URL `theory.stanford.edu/~amitp/GameProgramming/`, accessed 2022-06-28.

[28] Gastner, M. T., and Newman, M. E. J., “Diffusion-based method for producing density-equalizing maps,” *Proceedings of the National Academy of Sciences*, Vol. 101, No. 20, 2004, pp. 7499–7504. doi:10.1073/pnas.0400280101.