



Energy optimal 3D flight path planning for unmanned aerial vehicle in urban environments

Hannes Rienecker¹ · Veit Hildebrand¹ · Harald Pfifer¹

Received: 23 December 2022 / Revised: 21 April 2023 / Accepted: 22 May 2023
© The Author(s) 2023

Abstract

This paper presents a general approach to compute energy optimal flight paths for unmanned aerial vehicle (UAV) in urban environments. To minimize the energy required, the flight path is optimized by exploiting local wind phenomena, i.e., upwind and tailwind areas from the airflow around buildings. A realistic wind field of a model urban environment typical for continental Europe is generated using PALM, a Large Eddy Simulation tool. The calculated wind field feeds into the flight path planning algorithm to minimize the energy required. A specifically tailored A-Star-Algorithm is used to optimize flight trajectories. The approach is demonstrated on a delivery UAV benchmark scenario. Energy optimal flight paths are compared to shortest way trajectories for 12 different scenarios. It is shown that energy can be saved significantly while flying in a city using knowledge of the current wind field.

Keywords Flight path optimization · Unmanned aerial vehicle · Unmanned aerial system

1 Introduction

There are currently big challenges in last-mile logistics in urban environments. Novel approaches are becoming increasingly essential due to rising traffic and space use. The last step of a supply chain is the least efficient. However, electrically powered unmanned aerial vehicles (UAV) are an environmentally and time-efficient alternative in the last step of a supply chain, even in remote locations.

The adoption of last-mile logistics in urban environment by delivery UAVs motivates this work. UAVs can reduce urban street traffic and shorten delivery times. Furthermore, they also have the possibility of automation and are environmentally friendly due to electric powering. In contrast to

ground-based vehicles, UAVs have lower payload [1]. Therefore, the efficiency of logistic UAVs has to be increased. An easy approach is optimizing the flight path for an energy optimal flight trajectory. Our previous work in [2] shows that exploiting local wind conditions in an urban environment can significantly reduce the power consumption.

This paper contributes an approach to obtain optimal energy efficient flight paths of typical delivery missions using the knowledge of the wind field in an urban environment. This novel holistic approach consists of different aspects. First, a qualitative realistic wind field for an urban environment is necessary. Various methods ranging from simple statistical methods to complex approximative solutions of the atmospheric boundary layer equations exist in literature. For a good overview, the reader is referred to [3]. In this paper, a Parallelized Large-Eddy Simulation Model (PALM) [4] is used to prognose the wind field, see Sect. 3. It is a Computational Fluid Dynamics (CFD) simulation, or to be more precise a Large Eddy Simulation (LES).

Second, ensuring an energy-optimal flight through different wind conditions is a classical route optimization problem. There are various methods to find an optimal route: Branch-and-Bound [5], evolutionary computing, e.g., [6], multiple-agent-systems [7] neural network [8], and experience optimization can all be used in the embedded optimization process. Common problems in UAV flight path

Veit Hildebrand and Harald Pfifer have contributed equally to this work.

✉ Hannes Rienecker
hannes.rienecker@tu-dresden.de
Veit Hildebrand
veit.hildebrand@tu-dresden.de
Harald Pfifer
harald.pfifer@tu-dresden.de

¹ Chair of Flight Mechanics and Control, Technische Universität Dresden, 01062 Dresden, Saxony, Germany

planning and optimization in literature deal with finding the shortest trajectory in scenarios with obstacles [9], hazardous weather avoidance [10], restricted air space [11], agricultural applications, such as fertilizers and pesticide spray, in crop fields just on specific regions [12], and military objectives like avoiding enemy radar sites [13]. A hybrid routing and scheduling problem of an UAV delivery system leads to minimizing the travel time [14]. In [15], modern airline trajectories have been optimized with respect to minimum fuel flow, considering flight paths constraint in the horizontal plane. Most of the trajectory optimizations, e.g., [15–17], uses the A-star-algorithm or its variant Theta-Star, e.g., [10, 18]. This type of Branch-and-Bound method has advantages in three dimensional path finding [15] and computational efficiency [19]. A custom-tailored A-star algorithm is proposed in Sect. 4 to deal with the energy efficient path planning in urban environments. It specifically considers turning constraints of the UAV directly in the optimization.

Finally, pruning and smoothing techniques are required to achieve a flyable trajectory. The optimization itself is based on discrete grid points leading to a non-smooth, fractioned track. Pruning is a basic pre-smoothing technique to find shortcuts in grids for unrestricted air space. Path smoothing is defined using piecewise polynomials to achieve a continuous trajectory respecting the limits of the UAV flight envelope. We use continuous cubic Bézier spiral segments, derived in [20] to satisfy maximum curvature constraints of the UAV, see Sect. 4.3.

We generate a realistic city district to showcase the proposed approach applied to a typical European area. Moreover, the applied holistic approach connects the generated wind field, using the Large Eddy Simulation tool PALM, path optimization, and flight trajectory modeling. The methodology is verified by 12 delivery tasks, where their energy optimized paths are compared to the shortest way in Sect. 5.

2 Problem formulation

A typical last-mile logistic scenario for a UAV is the delivery of goods in a city district flying from a pick-up point to a drop-off point. The route is operated by a fully electric fixed wing aircraft, similar in size and characteristics to a Phoenix Wings PWOne delivery UAV [21] with a wing span of 1.3 m and a maximum take-off mass of 3.4 kg. Average cruise speed is 60 kph, where an estimated best glide ratio of 20 is assumed.

We develop a generic city model, which represents a typical European urban area. It is consisting of an arrangement of eight uniquely shaped buildings with different building heights, as illustrated in Fig. 1. In detail, there are three residence buildings with a height of 50 m, four-terraced houses with a height of 20 m and a supermarket

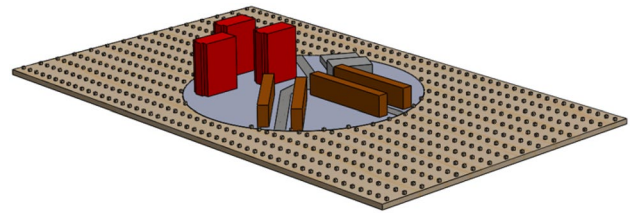


Fig. 1 Generic city model, which represent a typical European urban area for simulating delivery tasks

building with attached office block with a height of 15 m. This arrangement is developed to obtain a scenario with typical local wind effects. Exploiting these wind fields presents an opportunity to reduce the energy consumption of an UAV during a delivery mission.

This paper examines four delivery tasks with three different wind speeds each. This results in a total of 12 test scenarios. Figure 2 depicts these scenarios. Only one wind direction is used for all cases to stay in the influence zone of the high-rising buildings. These four delivery tasks are defined by flying from Point North to Point South and South to North, as well as Point West to Point East and East to West. In each scenario, the goal is to minimize the UAV's energy required using local flow effects to its advantage. It is assumed that the aircraft always flies at a constant true airspeed corresponding to its best-performance cruise speed. For each scenario, the same start and end altitude of 20 m are assumed. This represents a realistic initial altitude for take-off and landing in multicopter-mode for air delivery in cities. Take-off and landing procedures are neglected for energy consideration. Average wind speeds in a typical continental European city are the basis for this investigation. Dresden, Germany, represents such a city, and its wind speeds averaged over a whole year are depicted in Fig. 3. At the height of 10 m, three characteristic wind speeds can be extracted to represent the wind speed specter. The average wind speed of the windiest and calmest days and the average wind speed over one year are summed up in Table 1. They were transformed into a wind profile shape, obtained by a wind tunnel experiment, taken from [22]. The upper end of such a profile is characterized by the undisturbed flow above with its speed. Table 1 summarizes this freestream wind speed $u_{w\infty}$ for the profiles.

3 Wind field prediction using LES

The PALM system is used in the present paper to solve the large eddy equations and to obtain a realistic wind field within an urban environment for the following flight path

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

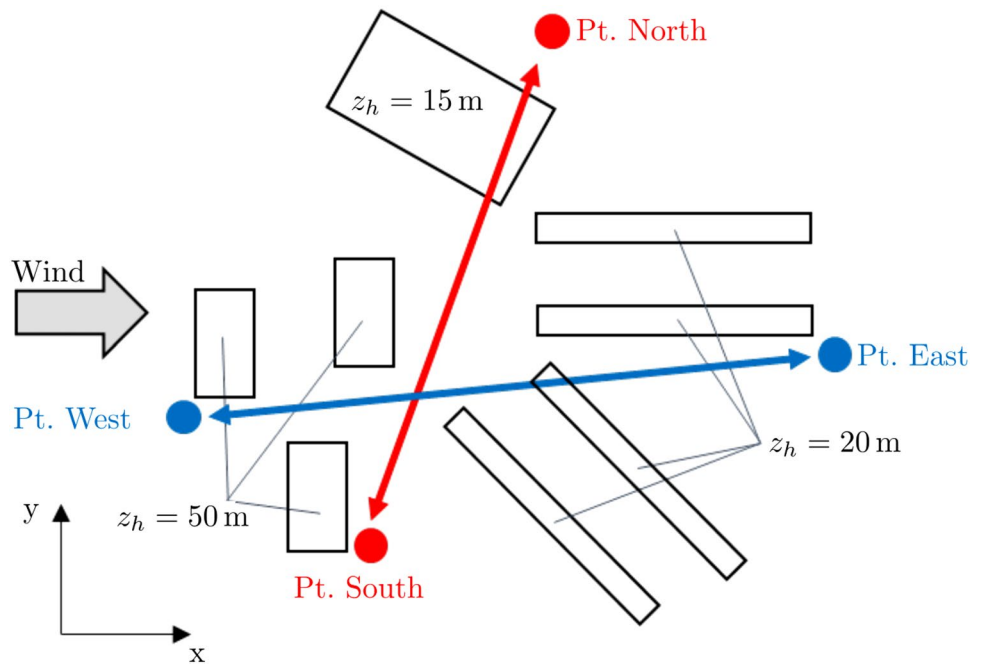


Fig. 3 Average of mean hourly wind speeds (dark gray line), with 25th to 75th and 10th to 90th percentile bands for Dresden, representing a typical continental European city [23]

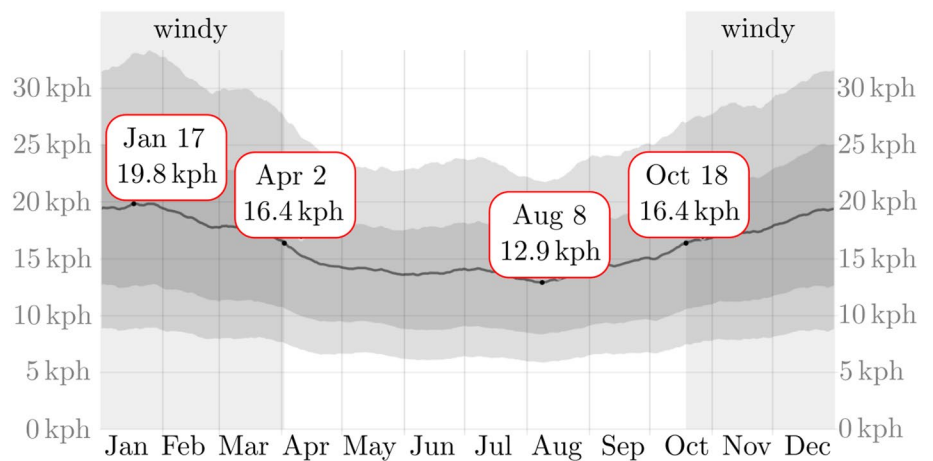


Table 1 Summary of wind speeds applied in scenarios

Explanation	$u_w(z = 10\text{ m})$ in kph	$u_w(z = 10\text{ m})$ in m/s	$u_{w\infty}$ in m/s
Calmeast 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

optimization. PALM¹ was developed by the University of Hannover as a tool to simulate urban climate and its boundary layer [4]. It has been validated for flow around

solid obstacles [24, 25] and used in various real urban environments [24–27]. A detailed description of PALM and its capabilities can be found in [4]. In short, PALM calculates the non-hydrostatic, filtered, incompressible Navier–Stokes equations in Boussinesq-approximated form. Furthermore, the subgrid-scale turbulent kinetic energy (SGS-TKE) is solved.

For the presented study, the topography of the generic city model was created in PALM. PALM uses equidistant horizontal grid spacings. Each grid volume can be either air or solid, i.e., representing a building. Due to the rectangular grid, oblique building walls require additional care in modeling. If more than half of a grid volume is geometrically belonging to a building, the whole volume will be set as a solid. An obstacle-free extended computational domain

¹ <https://palm.muk.uni-hannover.de/>.

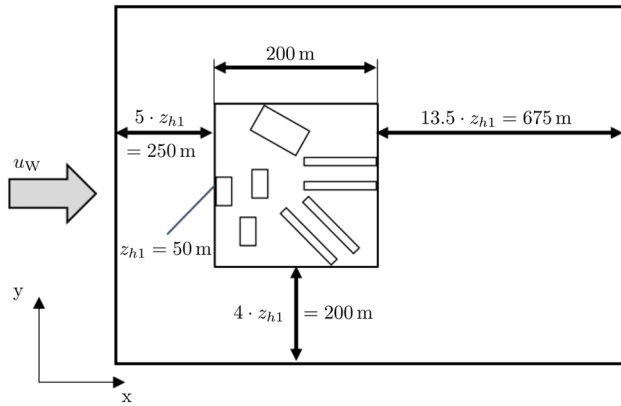


Fig. 4 Scenario for LES simulation with up- and downstream area

in the vertical, lateral, and flow directions is necessary to catch all flow effects. A guideline for CFD simulation of flows in the urban environment in [28] includes recommendations for the distances from built-up area to boundary. The simulation inlet and the top boundary should be at least five times the height of highest building away from the closest building to comply with this guideline. For urban areas with multiple buildings, the lateral boundaries can be closer and are placed four times the height of highest building away. The region behind the considered area is important

to allow redevelopment of the flow behind the wake region. It depends on the blocked cross-sectional area of the buildings in flow direction [28]. Sensitivity studies with different computation domain sizes were conducted to eliminate artificial accelerations to fulfill these requirements. These yield a distance of 13.5 times the height of highest building from last building to outflow. The grid size for all experiments was set to $\Delta_{x,y,z} = 2.5$ m. This size offered the best trade-off between computational time and accurate resolution of flow effects. The same setup was used for the different wind speeds described in Sect. 2. The setup of the topography for the scenarios is depicted in Fig. 4.

For the simulation, a realistic wind profile is very important to achieve meaningful results. PALM is using a cycling boundary condition that uses the outflow as inflow, i.e., the outflow is turbulent after flowing through the urban area and is reused as realistic turbulent inflow until the whole simulation converges. Hence, it is necessary to set an initial wind profile to initialize the simulation. Therefore, a realistic baseline from a scaled wind tunnel experiment for urban environment applications was used. Since the simulation converges to some quasi-steady solution, it has to be checked finally, if the actual wind profile from PALM is still matching with the wind tunnel experiment. For the simulation, the settings shown in Table 2 are used for PALM. The wind tunnel experiment was conducted at Technische Universität

Table 2 Used settings in the LES model PALM

Setting	Set in computation
Turbulence closure	1.5-order according to Deardorff
Perturbation pressure solver	FFT with temperton-algorithm routines
Momentum advection	Wicker-Skamarock-scheme 5th order
Scalar advection	Wicker-Skamarock-scheme 5th order
Boundary conditions	Horizontal: cyclic boundaryconditions Volume flow conservation at right and north boundary Bottom and building walls: constant flux layer following Monin-Obukhov similarity theory Top: free-slip conditions
Instream	bulk velocity u_b with 1D wind profile for initialization
Discretization (space)	Finite differences (Unchangeable)
Grid size nx	450 (corresponds to 1125 m)
ny	240 (corresponds to 600 m)
nz	100 (corresponds to 250 m)
Discretization (time)	Runge–Kutta-3
End time	14400 s
Averaging output time	3600 s
Coriolis force	Switched off
External forcing and nesting	Switched off
Ocean option and coupling to atmosphere	Switched off
Embedded models	Switched off

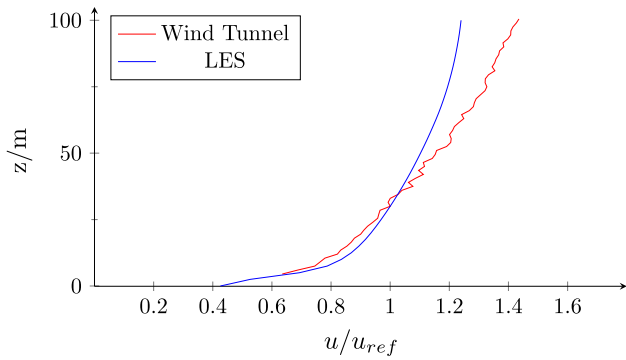


Fig. 5 Comparison between wind profile of wind tunnel experiment and LES simulation

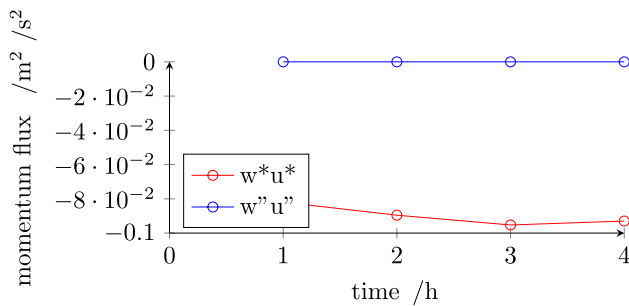


Fig. 6 Subgrid-scale momentum flux $w''u''$ and resolved-scale counterpart w^*u^* , for each mean value over the height

Dresden and its data is taken from [22]. Hence, the inflow averaged wind profiles were compared. Figure 5 shows both, rescaled wind tunnel experiment and LES simulation, normalized to u_{ref} , the velocity u in the height of $z = 30$ m. They behave in a good consistency in the range of the building height. The variations above 50 m can be neglected as these are outside the considered altitude range.

Before the data is used for trajectory optimization, it has to be ensured that the LES is fully developed. Therefore, it has to be checked, if a quasi-stationary state is reached and the grid spacing is small enough to resolve the turbulent transport with it. Both criteria are satisfied. The time series of kinetic energy E , turbulent kinetic energy E^* and the maximum velocity components show no trends anymore after 15100 s (≈ 4 h) of simulation time, so the simulation was converging. Furthermore, the subgrid-scale momentum flux $w''u''$ is one order of magnitude smaller than the resolved-scale counterpart w^*u^* after two hours of calculation time, see Fig. 6. From this, it can be concluded that the grid spacing is sufficiently small according to [29].

The path planning algorithm needs the velocity components of the wind at each grid point as input, but PALM provides the perpendicular component at each side face of the volume mesh [30], e.g., component u in Fig. 7. Hence,

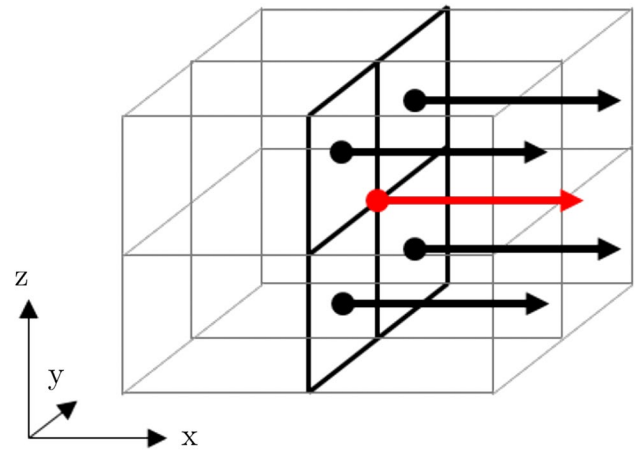


Fig. 7 Converting wind component u from grid data u_{i^*,j^*} to point data $u_{i,j}$

they are converted. In the converting process, the mean value of one velocity component is generated by respective component of the four adjacent side faces that belong to the point and are perpendicular to the component.

4 Trajectory optimization

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

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

where s is the position vector of the aircraft, s_0 the starting point, s_f the arrival point and $A(s)$ the quantity to be minimized at each step. To minimize the distance, for instance, $A(s)$ is set to $A(s) = 1$. In general, this is a complex optimization problem that is usually solved over a discrete grid instead of the continuous integral. Multiple solvers exist in literature to minimize Eq. (1), such as the Branch-and-Bound algorithm [5] in the present paper or evolutionary computing-based ones [6].

4.1 Cost function

The optimization problem in the present paper is to minimize the energy supplied by the propulsion system for the UAV for a flight with variable altitude between a fixed starting and ending point (see Sect. 2 for the detailed problem). In order to achieve an optimal flight trajectory, a new cost function detailed in this section is proposed. First, the generic flight path cost function (2) is defined as an integral over the flight paths and then discretized in sequentially flown through grid points until ending point N .

$$J = \int_{s_0}^{s_e} A(s) ds = \sum_{i=0}^N A_i \Delta s_i. \tag{2}$$

The second step is to derive a function for energy required to substitute the A_i in Eq. (2). First, we consider straightforward a 2D level flight and extend the approach to 3D space afterward. All flight mechanical assumptions are derived from corresponding standard literature, e.g., [31]. For simplification, the energy supplied by the propulsion system is assumed to be proportional to the thrust force multiplied by the path covered by the UAV. For the UAV in steady cruise flight, it is assumed that the thrust force equals the aerodynamic drag force D . Using this assumption, the energy required between two points of the discretized flight path E_i is given by

$$E_i = D \cdot \Delta s_i = D \cdot u_{UAV}, TAS \cdot \Delta t_i \tag{3}$$

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 . By introducing Δt_i , it is possible to implement the wind influence as follows. Defining u_{UAV}, i^* as the ground speed at the i -th grid point, the time it takes to cover Δs_i is given by

$$\Delta t_i = \frac{\Delta s_i}{u_{UAV}, i^*}. \tag{4}$$

Considering now headwind $u_{W,i}$ and crosswind $v_{W,i}$ at the i -th grid point, the relationship between ground speed u_{UAV}, i^* and true airspeed is given by

$$u_{UAV}, i^* = \sqrt{(u_{UAV}, TAS - u_{W,i})^2 - v_{W,i}^2}. \tag{5}$$

Using Eqs. (3), (4) and (5) yields the following equation for the 2D energy required between two nodes:

$$E_i = D \cdot u_{UAV}, TAS \cdot \frac{\Delta s_i}{\sqrt{(u_{UAV}, TAS - u_{W,i})^2 - v_{W,i}^2}}. \tag{6}$$

In the next step, up- and downwinds are implemented for level flight. E.g., assuming no change in flight height due to upwind components requires the UAV to pitch down to hold the flight level. Hence, the upwind $w_{W,i}$ leads to a decrease of the flight path angle $\gamma_{i,wind}$:

$$\sin(\gamma_{i,wind}) = \frac{w_{W,i}}{u_{UAV}, i^*}. \tag{7}$$

A decrease of flight path angle leads to less required thrust due to the balances of flight mechanic forces, because a component of the weight supports the force in flight direction.

Hence, at an upwind point, the perceived drag that needs to be compensated by thrust reduces to:

$$D_i = D_0 - \sin(\gamma_{i,wind}) \cdot mg \tag{8}$$

with D_0 as drag force in normal horizontal flight condition and mg the weight.

Considering now the 3D case allowing changes in altitude, an additional summand is added to the flight path angle γ_i , i.e., $\gamma_i = \gamma_{i,path} + \gamma_{i,wind}$. The wind component is as defined in Eq. (7). The altitude changing component is given by

$$\tan(\gamma_{i,path}) = \frac{\Delta s_{i,z}}{\Delta s_{i,xy}}, \tag{9}$$

where $\Delta s_{i,z}$ is the vertical distance and $\Delta s_{i,xy}$ the horizontal distance to the next point. Implementing this into Eq. (6) and using the relationship in Eq. (8) yields:

$$E_i = \frac{\Delta s_i}{\sqrt{(u_{UAV}, TAS - u_{W,i})^2 - v_{W,i}^2}} \cdot (D_0 - \sin(\gamma_i) \cdot mg) \cdot u_{UAV}, TAS. \tag{10}$$

The influence of vertical wind on ground speed u_{UAV}, i^* can be neglected because it is significantly smaller than the other components. The definition of D_0 is set by the glide ratio G :

$$D_0 = \frac{L}{G} = \frac{mg}{G} \tag{11}$$

with L as lift force equal to the weight force and the glide ratio G of the fixed-wing UAV. Using Eq. (10), the cost function for the minimization of the total energy required can be written as:

$$E = \sum_{i=0}^N \frac{\Delta s_i}{\sqrt{(u_{UAV}, TAS - u_{W,i})^2 - v_{W,i}^2}} \cdot (D_0 - \sin(\gamma_i) \cdot mg) \cdot u_{UAV}, TAS. \tag{12}$$

4.2 Extended A-star-algorithm

As mentioned before, the *A-Star-Algorithm* is often used to optimize routes or trajectories. First, we want to give a brief explanation of the basic *A-Star-Algorithm*. Afterward, a custom tailored version is proposed. The basic *A-Star-Algorithm* is an adaption of the Dijkstra-algorithm [19] and finds cost optimal paths from a starting to an ending point. Alg. A in the appendix shows and explains the exact steps of basic *A-Star-Algorithm*. In brief, the

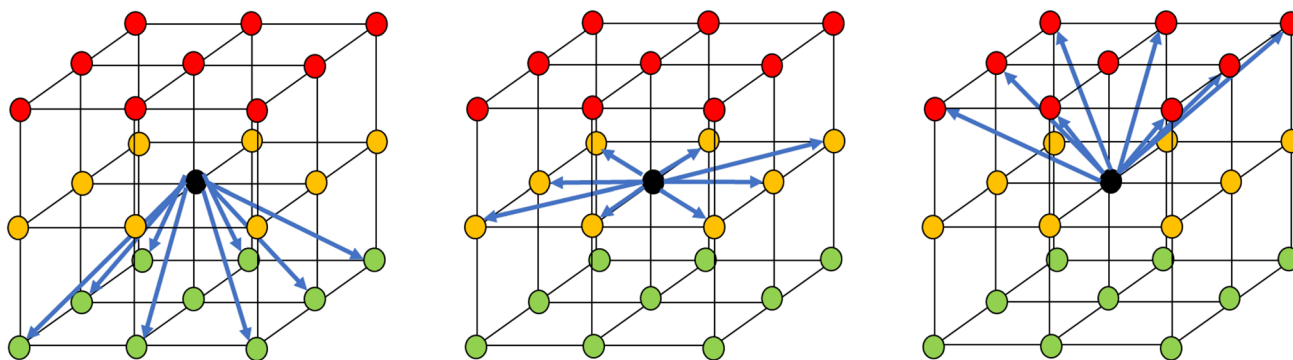


Fig. 8 The connection lines from a vertex (black) to all adjacent vertices (red, green and orange) define possible tracks (blue arrows) in 3D space

algorithm starts at the starting point and creates the optimal path by following the best point at a branching. The best point is characterized by the lowest total cost of the path. 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 $h(s)$. Hence, the algorithm starts to check all possible paths from the starting point in detail, but stops moving forward with a potential path if the considered branch is considered too expensive. This approach leads to computational savings time because not all possible paths need to be computed.

In this paper, the exact cost function $g(s)$ of the algorithm is set to Eq. (12). The heuristic cost function $h(s)$ provides A-Star an estimation of the minimum cost from any vertex s to the goal. On the one hand, the closer $h(s)$ is to the exact cost of this path, the faster A-Star finds the best way [32]. On the other hand, if $h(s)$ is greater than the exact cost, it is not guaranteed to find the best way. Hence, the heuristic function must be smaller than the exact cost. Heuristic costs must be the same scale as the exact cost. Therefore, in this paper, $h(s)$ is defined on the basis of Eq. (3) as

$$h(s) = D_0 \cdot \Delta s_{s,s_{goal}} \cdot w, \quad (13)$$

where D is the aerodynamic drag force, $\Delta s_{s,s_{goal}}$ the distance from point s to the ending point and w a weighting factor. The weighting factor is set to 0.001 to ensure staying below the exact cost. Equation (13) represents the energy required from point s to the final point if the UAV flies directly without the influence of wind. However, it is possible that this energy required is approximately zero if favorable wind conditions are present. An extensive parameter study showed that $w = 0.001$ always leads to the optimum path under acceptable computation time (in this scenario).

An extension of the basic algorithm enables to consider a maximum curvature constraint of the UAV (specified in Sect. 4.3). At constant speed, an UAV would not be able to turn at smaller radii, e.g., due to a maximum allowed load factor. That imposes the demand for the curvature constraint. This constraint leads to a maximum allowed change of heading at a specific distance. Therefore, at each point s , the information about the point before has 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 s has different outgoing branches to leave s , depending on the direction of reaching this point. Thus, we introduce an extended A-Star-Algorithm. Alg. B in the appendix shows and explains the differences to the basic one, where a function ensures the prespecified requirements of the connecting lines between the points. Furthermore, the basic variable of just one point-to-point connection becomes one that consists of the point connections of three points to consist of the allowed path segment.

4.3 Path modeling

4.3.1 Path discretization

As a starting point in this paper, the vertices from PALM in Sect. 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. 8. 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. In addition to the flight path angle range, a minimum turn radius is considered in this paper. Both lead to a shrinking of the original grid in the vertical direction and an expansion in the horizontal one.

Specifically, the grid is adjusted in the following ways. The minimum turning radius r_{min} of the UAV, which is used for the horizontal grid distance, is defined by

$$r_{min} = \frac{u_{UAV, TAS}^2}{g \cdot \tan(\arccos(\frac{1}{n_{max}}))}, \tag{14}$$

where $u_{UAV, TAS}$ is the true airspeed (TAS), g the gravitational acceleration, and n_{max} a maximum load factor of the UAV. Reference [33] 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 horizontal flight grid distance Δf_h is set to

$$\Delta f_h = \frac{r_{min}}{\Delta x}, \tag{15}$$

where Δx can be substituted by Δy due to the equidistant grid of the LES.

The optimal gliding ratio G determines the vertical flight grid distance Δf_v as:

$$\Delta f_v = \frac{\Delta x}{G}. \tag{16}$$

Eq. (16) 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 $u_{UAV, TAS} = 60$ kph and $G = 20$, as well as LES grid $\Delta x = 2.5$ m, Δf_h is $\Delta f_h = \frac{r_{min}}{\Delta x} = \frac{12.4m}{2.5m} \approx 5$ and Δf_v is $\Delta f_v = \frac{\Delta x}{G} = \frac{2.5m}{20} = 0.125$ m. Since this derived vertical grid is finer than the one, used in PALM, wind components u_w, v_w, w_w are derived by linear interpolation. The horizontal grid is coarser than the LES grid. Nevertheless, the wind field data of the points between it is taken into account for energy required determination.

4.3.2 Path pruning

The resulting path of an A-Star-Optimization run is characterized by successive waypoints. The adjusted grid size as explained in the previous paragraph constrains each step size. However, there can be paths with lower cost by



Fig. 9 The basic pruning algorithm is not able to find a direct line if there is a doubled-convex path

ensuring a direct connection between non-adjacent points. Pruning or sometimes misleadingly called *post-smoothing*² is a method to find these shortcuts after the first optimization constrained to the grid. A basic pruning technique is looping through the path obtained by A-Star-Algorithm and check if there is a straight connection between points without crashing an obstacle (see Alg. C in the appendix). This method has a shortcoming, e.g., if searching for the shortest path. On the one hand, these line-of-sights lead to shorter tracks. On the other hand this algorithm is not able to find the shortest way, if there is a path with more than two convex courses. There will always be a stop at the protruding point, which is illustrated in Fig. 9.

The approach in this study is using the A-Star-Algorithm once again for the pruning step. Only the points s^* obtained by the first run are considered as vertices for the second run. Each vertex has a connection to every other vertex if there is a line of sight between them. The other A-Star-Inputs are handled as follows. The exact cost $g(s^*)$ of going unbounded to the is defined by calculate this direct way. The heuristic cost $h(s^*)$ is set by straight-line cost without respecting obstacles like in the first run. The algorithm finds a path that is the true low-cost one. Due to the very low number of considered vertices, this step is computationally fast.

4.3.3 Path smoothing

Finally, the optimal waypoint-based paths have to be converted to a flyable smooth trajectory. Therefore, a continuous-curvature path-smoothing algorithm based on cubic Bezier curves, including a maximum curvature constraint [20], is used. This method is applied to the path in sections. Every section takes three path points into account. Every line between two path points is split into two halves

² In this paper, smoothing is defined as creating a flyable trajectory consisting of curves without sharp changes in heading.

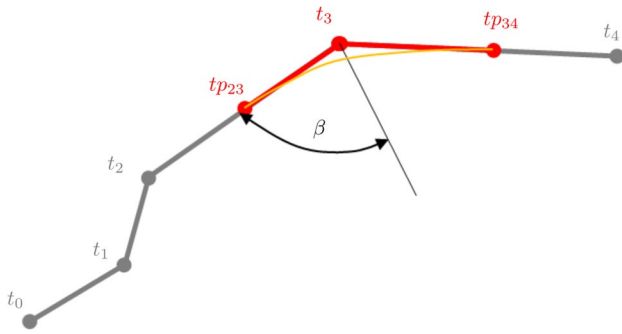


Fig. 10 Path smoothing of gray path between point t_2 and t_4 : Define turning points $tp_{2,3}$, $tp_{3,4}$ and construct two cubic Bézier spiral curves

to retrieve turning points of one section with a continuous junction to the next section. On the basis of path point t_3 , the procedure is illustrated in Fig. 10 and can be described as follows. Consider the section containing the points t_2 , t_3 and t_4 . The proposed smoothing adds turning points tp_{23} and tp_{34} halfway between t_2 and t_3 and t_3 and t_4 , respectively. The original point t_3 becomes a curvature steering point and hence, is no longer a part of the flight trajectory. The smoothing procedure is detailed in [20]. In short, the points are transformed into one plane to become a 2D problem, first. Second, eight control points are determined that construct two cubic Bézier spiral curves between tp_{23} and tp_{34} .

One curve P is defined by the four control points B_0, B_1, B_2, B_3 via the function

$$P(m) = B_0(1 - m)^3 + 3B_1(1 - m)^2m + 3B_2(1 - m)m^2 + B_3m^3, m \in [0, 1]. \tag{17}$$

Third, the control points are getting transformed back into 3D space. Step 2 contains the consideration of the minimum turning radius constraint. The minimum turning radius r_{min}

of the UAV in Eq. (14) provides a maximum curvature k_{max} of the flight path:

$$k_{max} = \frac{1}{r_{min}} \tag{18}$$

and is considered by

$$d_{min} = \frac{1.228 \cdot \sin \beta}{k_{max} \cdot \cos^2 \beta}, \tag{19}$$

where β is the half angle between the two lines. The distance d_{min} is compared with the length of the two lines $tp_{2,3}t_3$ and $t_3tp_{3,4}$. Note that, it is possible that one line is shorter than d_{min} . To fulfill the constraint, we introduce several approaches in this order: First, it is checked if skipping the path point, the point before or the following point, will improve the smoothing procedure. Second, it is checked if slightly shifting proper path point improves the smoothing procedure. Third, continuous-curvature segments can be defined, which cross the initial path line at the position of intended turning point. Hence, there is a difference between initial path and adjusted trajectory, if these techniques were applied. Equations (18) and (19) show that the difference is increasing if the amount of heading changes and the trajectory differs from the optimized path. Thus, there is just a small difference due to the implemented flight grid discretization. All approaches include obstacle avoidance. Finally, the resulting trajectory sections are merged to obtain the continuous trajectory.

5 Results of trajectory optimization

This section presents the results of the flight path optimization using the cost function proposed in Sect. 4.1. Overall, 12 different scenarios are tested according to the problem formulation in Sect. 2. Specifically, these are

Fig. 11 Comparison between shortest-way-optimization (Orange Square) and smoothed energy-optimization (Blue Square) flight track in North-South scenarios

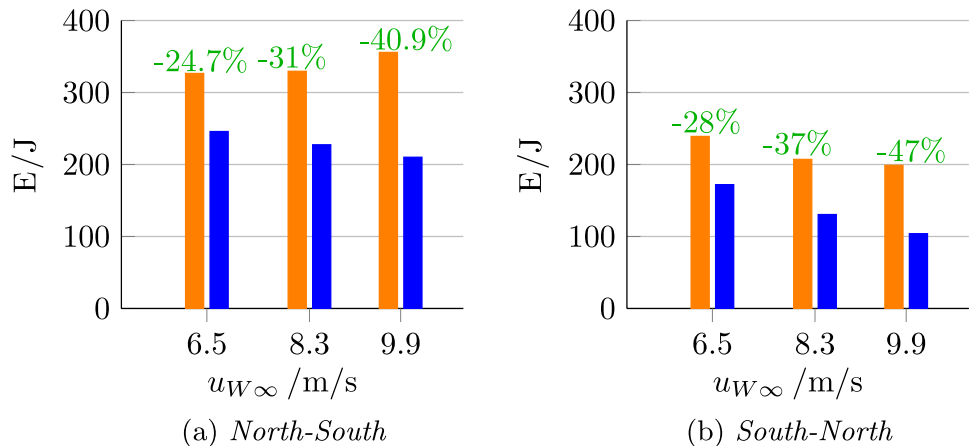


Fig. 12 Comparison between shortest-way-optimization (Orange Square) and smoothed energy-optimization (Blue Square) flight track in West-East–West scenarios

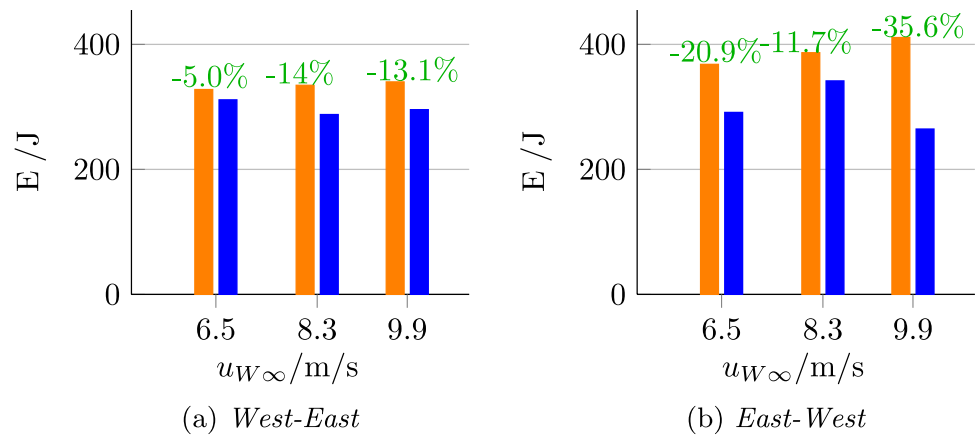
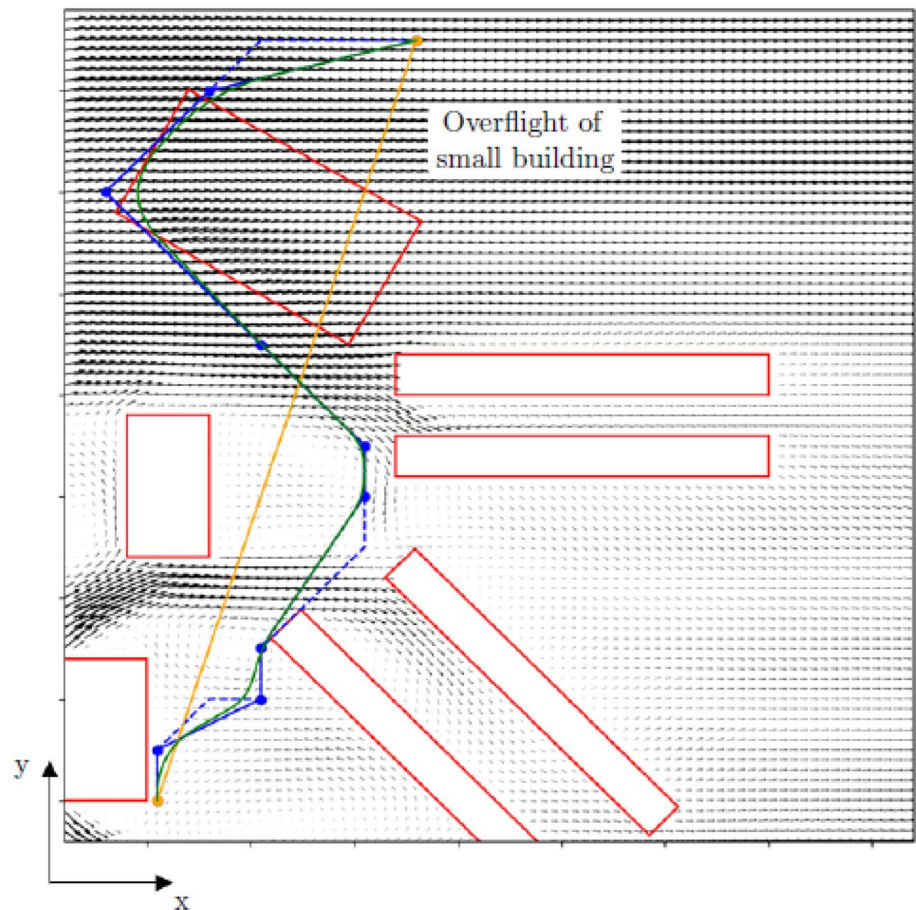


Fig. 13 Flight path for $u_{W\infty} = 6.5m/s$ with wind field in height of 20 m(\rightarrow), buildings (red line), shortest way (orange line), energy optimized path after A-Star-Algorithm (dotted line), Pruning (line merged with circle) and Smoothing (green line), flying South to North



three scenarios with a freestream wind speed of 6.5 m/s, 8.3 m/s and 9.9 m/s per one of the four track directions. For each scenario, two different optimal trajectories are computed. The first one is based on the energy optimal cost function of Eq. (12). The second one is a simple shortest

path optimization as described in Eq. (2). The optimization itself is in each case performed by the A-Star-Algorithm. The grid resolution of the trajectory optimization is the adjusted LES grid from the wind field prediction as described in Sect. 4.3.

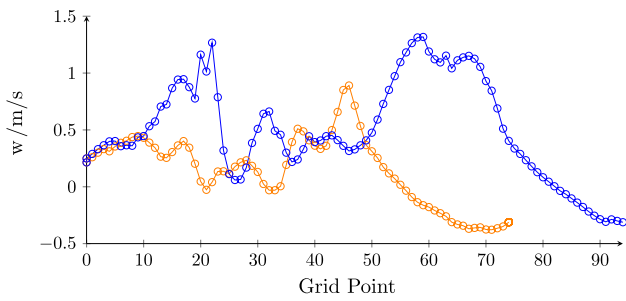


Fig. 14 Upwind for $u_{W\infty} = 6.5\text{m/s}$, South to North, flying shortest way (Orange line) and energy optimized path (Blue line)

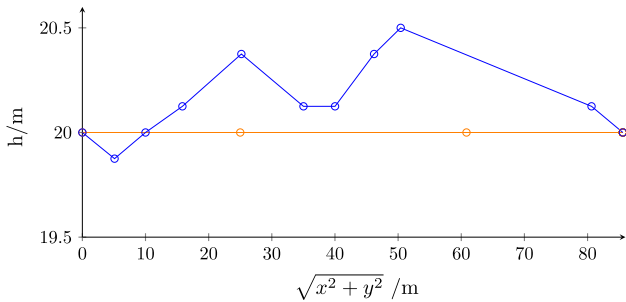


Fig. 15 Altitude for $u_{W\infty} = 6.5\text{m/s}$, West to East, flying shortest way (Orange line) and energy optimized path (Blue line)

Figures 11 and 12 show the results for the energy required of all scenarios. The energy savings in percentage are calculated as follows:

$$p = \frac{E_{\text{shortest way}} - E_{\text{energy opt.}}}{E_{\text{shortest way}}} \cdot 100\% \tag{20}$$

All scenarios are capable of achieving significant energy reduction. Furthermore, a clear trend of increased energy savings with higher wind speeds is present as expected. The only exception marks the East-West-8.3 m/s-scenario. The shortest path optimization has no knowledge about the current wind field. Thus, the same trend is more washed out and depends on head- or tailwind.

An example of the actual flight paths is given in Fig. 13, where orange is the shortest path and green the smoothed lowest energy required. Note that most of the energy savings actually come from exploiting regions of upwind and not tailwinds. This becomes clear when looking at the upwind along the trajectories, see Fig. 14. The energy optimized trajectory is leading through much larger upwind fields than the shortest way. If one observes Eq. (6), it can be seen that

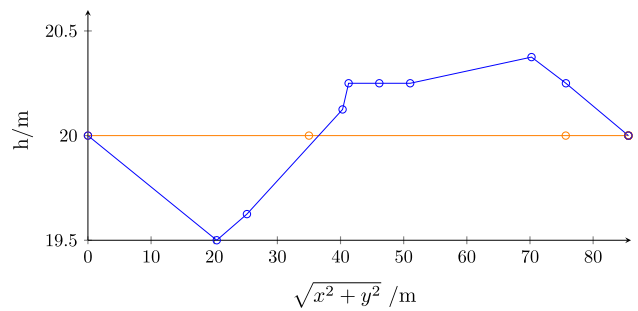


Fig. 16 Altitude for $u_{W\infty} = 6.5\text{m/s}$, East to West, flying shortest way (Orange line) and energy optimized path (Blue line)

Table 3 Flight tracks

Route	$u_{W\infty}$	Shortest way track in m	Energy optimized track in m
North-South	6.5	197.6	223.0
	8.3	197.6	306.3
	9.9	197.6	323.8
South-North	6.5	197.6	285.8
	8.3	197.6	288.0
	9.9	197.6	308.3
West-East	6.5	216.0	242.3
	8.3	216.0	242.3
	9.9	216.0	242.3
East-West	6.5	215.6	233.7
	8.3	215.6	256.2
	9.9	215.6	258.1

small differences of upwind have a greater impact on energy required than small ones of tailwinds.

The obtained trajectories have some characteristics worth noting for further investigations. First, all energy optimized paths are close to the rooftops of the elongated buildings. The optimizer detects the strong upwinds in front of buildings, which explains why flying alongside a building with wind perpendicular to it yields lower energy required. This effect can be called ridge lift, similar to gliding in mountains. Hence, the North-South-North paths have more savings than West-East-West paths due to more opportunities to “slide” on the roofs. Second, flying with tailwinds yields to flying higher and flying with head winds yields to flying lower. This is expected considering the wind profile in the ground’s boundary layer. This altitude behavior is seen with tailwind in Fig. 15 and headwind in Fig. 16. Still, note that the altitude changes are low. The energy required of climbing is

much higher than staying on level. In contrast, descent is not able to retrieve these costs. Third, energy optimized flight paths have a remarkably longer track than the shortest way to exploit the favorable upwind locations. However, the energy savings compensate the additional way. Table 3 shows the flight tracks corresponding to all scenarios.

Finally, some caveats of the used methodology shall be detailed. The heuristic approach of the A-Star-Algorithm can result in inconsistencies. A-Star is not guaranteed to find an optimal solution and the heuristic function must be chosen carefully, as it (notably) influences the results. However, these drawbacks are deemed acceptable to achieve decent computation time. Further, the heuristic cost function leads to a dependence on direction. The algorithm is more likely to lead the path to a favorable upwind location if this area is close to the starting point. Each point close to the starting point has a similar high heuristic cost and very low exact costs. The costs for the detour to the upwind location are low. At the end of the path planning, close to the destination, it is better to fly directly to the destination, because the costs flying to the upwind zone first, are too expensive without knowledge of upwinds. This effect can be seen when observing that flying with headwind (East to West) requires less energy than flying with tailwind (West to East), see Fig. 12. The same can be noticed in the North–South–North track in Fig. 11 with very low head- and tailwinds. In this case, a similar energy required would be expected. Moreover, the smoothing technique used is also direction sensitive. A 90-degree change of the pre-smoothed path close to the corner of a building yields to a suggested smoothed path through the obstacle. The smoothing technique redirects the UAV in different ways depending on the starting point that influences the path modification techniques. Thus, the delivery scenarios are designed without such case, e.g., a starting point between two terraced houses and a destination perpendicular to them.

The smoothing approach can cause unfavorable paths if it misses strong upwind zones. Especially the results of the aforementioned East–West-8.3 m/s-scenario, differ highly

from the trend of reduced energy required with increased wind speed. Hence, implementing smoothing techniques directly in the optimization process for further improvements may be ideal.

6 Conclusion

An approach to optimize 3D flight trajectories of a delivery UAV with respect to energy required was presented. The applied wind field was obtained by an LES simulation. It was shown that the tailored A-Star-Algorithm was distinguished for optimizing flight paths. The main result is that it is always beneficial to optimize the flight path at each wind speed to significantly reduce energy consumption. The test scenarios highlighted that upwinds provide the most saving potential for optimizing the flight path. In the future, more attention will be given to implementing smoothing techniques into the optimizing process to improve the presented method. Furthermore, there has some higher energy required to be respected due to short curve sections, where higher lift produces more drag. The LES wind field is planned to be validated in a series of wind tunnel experiments at TU Dresden with a scaled city model.

Appendix A: 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 neighbor 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 $neighbourlist = [p_0 : (p_{xxx}, p_{xxy}, \dots), \dots, p_n : (\dots)]$, 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 $path = [s_0, \dots, s_n]$

```

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!"

```

Appendix B: Extended A-Star-Algorithm

At each point s , the information about the point before has 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 the point s itself as well as the attended point before s^{-1} for three reasons. First, this information is necessary for checking the heading change constraint in allocation of the neighbor points (Alg. A, line 15). Second, it is required during the reconstruction of the path at the end (Alg. A, line 13). The $parent$ -list consists of the analyzed connection between two points 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. Third, the algorithm suspends points that were analyzed (Alg. A, line 28). However, this point has to be analyzed 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 the point before s^{-1} , current point s and following point s' (line 17). Furthermore, the point lists in the basic algorithm become lists that consist of the point connections (s & s^{-1} as well as s' & s) to prevent the mentioned shortcomings.

Algorithm 2 Extended A-Star-Algorithm

Input: Neighbour list of points $neighbourlist = [p_0 : (p_{xxx}, p_{xy}, \dots), \dots, p_n : (\dots)]$, 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 $path = [s_0, \dots, s_n]$

```

1: function ASTAR()
2:    $g(s_{start} \& s_{start}) := 0$ 
3:    $parent(s_{start} \& s_{start}) := s_{start} \& s_{start}$ 
4:    $openPoints := \emptyset$ 
5:    $openPoints.add(s_{start} \& s_{start})$ 
6:    $closedPoints := \emptyset$ 
7:   while  $open \neq \emptyset$  do
8:      $n := None$ 
9:     for  $v \in openPoints$  do
10:      if  $s = None$  or  $g(v) + h(v) < g(n) + h(n)$  then
11:         $n := v$ 
12:       $s := n[0]$ 
13:      if  $s = s_{goal}$  then
14:         $path = ReconstPath()$ 
15:        return "Path found:",  $path$ 
16:      for all  $s' \in neighbours(s)$  do
17:         $rightDirectionBoolean = angleReq(s^{-1}, s, s')$ 
18:        if  $s' \& s \notin openPoints$  and  $s' \& s \notin closedPoints$  and
19:           $rightDirectionBoolean$  then
20:           $openPoints.add(s' \& s)$ 
21:           $parent(s' \& s) := n$   $\triangleright$  Note:  $n = s \& s^{-1}$ 
22:           $g(s' \& s) := g(n) + cost(s, s')$ 
23:        else
24:          if  $g(s' \& s) > g(n) + cost(s, s')$  and  $rightDirectionBoolean$ 
25:            then
26:               $g(s' \& s) := g(n) + cost(s, s')$ 
27:               $parent(s' \& s) := n$ 
28:              if  $s' \& s \in closedPoints$  then
29:                 $closedPoints.remove(s' \& s)$ 
30:                 $openPoints.add(s' \& s)$ 
31:           $openPoints.remove(n)$ 
32:           $closedPoints.add(n)$ 
33:      return "Path does not exist!"

```

Appendix C: Basic pruning technique algorithm

Algorithm 3 Basic pruning technique algorithm

Input: A-Star-Path $path = [s_0, \dots, s_n]$
Output: Pruned Path $path = [t_0, \dots, t_k]$

```

1: function PRUNING()
2:    $k := 0$ 
3:    $t_k := s_0$ 
4:   for all  $i := 1..n - 1$  do
5:     if not LineOfSight( $t_k, s_{i+1}$ ) then
6:        $k := k + 1$ 
7:        $t_k := s_i$ 
8:      $k := k + 1$ 
9:      $t_k := s_n$ 
10:  return  $[t_0, \dots, t_k]$ 

```

Acknowledgements 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.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Kirschstein, T.: Comparison of energy demands of drone-based and ground-based parcel delivery services. *Transp. Res. D Transp. Environ.* (2020). <https://doi.org/10.1016/j.trd.2019.102209>
- Rienecker, H., Hildebrand, V., Pfifer, H.: Energy optimal flight path planning for unmanned aerial vehicle in urban environments. In: Proceedings of the 2022 CEAS EuroGNC Conference. CEAS, Berlin, Germany CEAS-GNC-2022-031 (2022)
- Oettershagen, P., Muller, B., Achermann, F., Siegwart, R.: Real-time 3d wind field prediction onboard UAVs for safe flight in complex terrain. In: 2019 IEEE Aerospace Conference, pp. 1–10. IEEE, New York City, United States (2019). <https://doi.org/10.1109/aero.2019.8742160>
- Maronga, B., Gryschka, M., Heinze, R., Hoffmann, F., Kanani-Sühring, F., Keck, M., Ketelsen, K., Letzel, M.O., Sühring, M., Raasch, S.: The parallelized large-eddy simulation model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, recent developments, and future perspectives. *Geosci. Model Dev.* **8**(8), 2515–2551 (2015). <https://doi.org/10.5194/gmd-8-2515-2015>
- Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. *Econometrica* **28**(3), 497–520 (1960). <https://doi.org/10.2307/1910129>
- de Camargo, J.T.F., de Camargo, E.A.F., Veraszto, E.V., Barreto, G., Cândido, J., Aceti, P.A.Z.: Route planning by evolutionary computing: an approach based on genetic algorithms. *Procedia Comput. Sci.* **149**, 71–79 (2019). <https://doi.org/10.1016/j.procs.2019.01.109>
- Biswas, S., Anavatti, S.G., Garratt, M.A.: Particle swarm optimization based co-operative task assignment and path planning for multi-agent system. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–6. IEEE, New York City, United States (2017). <https://doi.org/10.1109/ssci.2017.8280872>
- Yu, J., Su, Y., Liao, Y.: The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Front. Neurobot.* (2020). <https://doi.org/10.3389/fnbot.2020.00063>
- Tang, L., Wang, H., Li, P., Wang, Y.: Real-time trajectory generation for quadrotors using b-spline based non-uniform kinodynamic search. In: 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1133–1138. IEEE, New York City, United States (2019). <https://doi.org/10.1109/robio49542.2019.8961485>
- Garcia, M., Viguria, A., Ollero, A.: Dynamic graph-search algorithm for global path planning in presence of hazardous weather. *J. Intell. Robot. Syst.* **69**, 285–295 (2012). <https://doi.org/10.1007/s10846-012-9704-7>

11. Babel, L.: Flight path planning for unmanned aerial vehicles with landmark-based visual navigation. *Rob. Auton. Syst.* **62**(2), 142–150 (2014). <https://doi.org/10.1016/j.robot.2013.11.004>
12. Srivastava, K., Pandey, P.C., Sharma, J.K.: An approach for route optimization in applications of precision agriculture using UAVs. *Drones* (2020). <https://doi.org/10.3390/drones4030058>
13. Bortoff, S.A.: Path planning for UAVs. In: Proceedings of the 2000 American Control Conference, pp. 364–3681. IEEE, New York City, United States (2000). <https://doi.org/10.1109/acc.2000.878915>
14. Sajid, M., Mittal, H., Pare, S., Prasad, M.: Routing and scheduling optimization for UAV assisted delivery system: a hybrid approach. *Appl. Soft Comput.* (2022). <https://doi.org/10.1016/j.asoc.2022.109225>
15. Rosenow, J., Lindner, M., Fricke, H.: Assessment of Air Traffic Networks Considering Multi-criteria Targets in Network and Trajectory Optimization. In: Deutscher Luft- und Raumfahrtkongress 2015. DGLR, Rostock, Germany (2015)
16. Junwei, Z., Jianjun, Z.: Path planning of multi-UAVs concealment attack based on new a method. In: 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, pp. 401–404. IEEE, New York City, United States (2014). <https://doi.org/10.1109/ihmsc.2014.198>
17. Li, J., Deng, G., Luo, C., Lin, Q., Yan, Q., Ming, Z.: A hybrid path planning method in unmanned air/ground vehicle (UAV/UGV) cooperative systems. *IEEE Trans. Veh. Technol.* **65**(12), 9585–9596 (2016). <https://doi.org/10.1109/tvt.2016.2623666>
18. Mandloi, D., Arya, R., Verma, A.K.: Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3d environment. *Int. J. Syst. Assur. Eng. Manag.* **12**(5), 990–1000 (2021). <https://doi.org/10.1007/s13198-021-01186-9>
19. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **4**(2), 100–107 (1968). <https://doi.org/10.1109/tssc.1968.300136>
20. Yang, K., Sukkarieh, S.: 3d smooth path planning for a UAV in cluttered natural environments. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 794–800. IEEE, New York City, United States (2008). <https://doi.org/10.1109/iros.2008.4650637>
21. Pwone. Phoenix-Wings GmbH. <https://phoenix-wings.de/pwone/>. Accessed 18 Oct 2021
22. VDI-Richtlinie 3783 Blatt 12, Umweltmeteorologie, Physikalische Modellierung von Strömungs- und Ausbreitungsvorgängen in der atmosphärischen Grenzschicht, Windkanalanwendungen. (Gründruck) (2022)
23. Climate and average weather year round in dresden germany. Cedar Lake Ventures, Inc.. <https://weatherspark.com/y/75895/Average-Weather-in-Dresden-Germany-Year-Round>. Accessed 25 Jul 2022
24. Letzel, M.O., Krane, M., Raasch, S.: High resolution urban large-eddy simulation studies from street canyon to neighbourhood scale. *Atmos. Environ.* **42**(38), 8770–8784 (2008). <https://doi.org/10.1016/j.atmosenv.2008.08.001>
25. Kanda, M., Inagaki, A., Miyamoto, T., Gryschka, M., Raasch, S.: A new aerodynamic parametrization for real urban surfaces. *Bound.-Layer Meteorol.* **148**(2), 357–377 (2013). <https://doi.org/10.1007/s10546-013-9818-x>
26. Letzel, M., Gaus, G., Raasch, S., Jensen, N., Kanda, M.: Turbulent Flow Around High-rise Office Buildings in Downtown Tokyo. *Dynamic Visualization in Science*, No. 13100, 2008
27. Tack, A., Koskinen, J., Hellsten, A., Sievinen, P., Esau, I., Praks, J., Kukkonen, J., Hallikainen, M.: Morphological database of Paris for atmospheric modeling purposes. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **5**(6), 1803–1810 (2012). <https://doi.org/10.1109/jstars.2012.2201134>
28. Franke, J., Baklanov, A.: Best Practice Guideline for the CFD Simulation of Flows in the Urban Environment: COST Action 732 Quality Assurance and Improvement of Microscale Meteorological Models, (2007)
29. Palmgroup: E3: Flow Around a Cubical Building. Institute of Meteorology and Climatology, Leibniz Universität Hannover, (2020). Institute of Meteorology and Climatology, Leibniz Universität Hannover
30. Arakawa, A., Lamb, V.R.: Computational design of the basic dynamical processes of the UCLA general circulation model. In: *Methods in Computational Physics: Advances in Research and Applications*, Vol. 17, pp. 173–265. Elsevier, Amsterdam, Netherlands (1977). <https://doi.org/10.1016/b978-0-12-460817-7.50009-4>
31. McClamroch, N.H.: Steady aircraft flight and performance. Princeton University Press, Princeton (2011). (ISBN: 9781680159097)
32. Pathfinding with A*. Python Pool. <http://theory.stanford.edu/ai/GameProgramming/> Accessed 28 Jun 2022
33. Majka, A.: Flight loads of mini UAV. *Solid State Phenom.* **198**, 194–199 (2013). <https://doi.org/10.4028/www.scientific.net/ssp.198.194>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.