A Framework for Model Based Contour Flight Planning

Rafael Paintner
M. Sc.
German Aerospace Center
Braunschweig, Deutschland

Sven Schmerwitz
Dr.-Ing.
German Aerospace Center
Braunschweig, Deutschland

ABSTRACT

This paper proposes a framework for helicopter contour flight planning. To automate low altitude flight, a method for reliable path planning is essential. This work introduces a framework that yields trajectories while aiming for continuous control input. Several metrics based on terrain and land-use data can be taken into consideration, based on which a hierarchical planner is employed. A coarse euclidean planner assures a globally optimized waypoint list, followed by a highly modified Vector Field Histogram algorithm which generates a flyable trajectory. The framework has been proven to produce satisfactory results using real-world data, showing significant path cost improvement when compared to a purely local planning approach. Furthermore, it has been validated, that the generated trajectories can be followed by the path following controllers deployed in the DLR's research simulator.

NOTATION

VFH	Vector Field Histogram
ACT/FHS	Active Control Technology / Flying Helicopter Simulator
AVES	Air Vehicle Simulator
BIT*	Batch Informed Trees
CM	Command Model
DTM	Digital Terrain Model
DSM	Digital Surface Model
OMPL	Open Motion Planning Library
OSM	Open Street Map
PID	proportional-integral-derivative
PRM	Probabilistic Roadmaps
ROC	Rate of Climb
SRTM1	Shuttle Radar Topography Mission with 1 Arc-Second Resolution
TVFH	Terrain Vector Field Histogram
UTM	Universal Transverse Mercator

c	Cost factor
C	Spring stiffness
d	Distance
D	Damping factor
h	Height
I	Inertia
k	Index of angular sector
K	Number of angular sectors
M	Torque
v	Velocity
w	Radius of the active window
r_c	Curve radius
α	Angle of angular sector
β	Angle of map tile in the global coordinate frame
γ	Climb angle
ϕ	Bank angle
χ	Heading

Presented at the 50th European Rotorcraft Forum & Technology Display, Marseille, France, September 10–12, 2024.

INTRODUCTION

During military helicopter missions, flight close to terrain might become necessary when hostile detection must be avoided, e.g. during medical evacuation, or to deploy troops. In literature, differing and even contradictory definitions of low altitude flight can be found (Refs. 1, 2). This work follows the classification of (Ref. 2). They follow the US-army training circular TC-1-15 and categorize low altitude flight into three classes: Low level flight, defined by constant altitude and constant speed, is recommended to travel long distances with reduced risk of detection. Contour flight is characterized by following the rough outlines of the terrain, while maintaining constant velocity. Finally, during nap-of-the-earth flight altitude and velocity are altered to allow flight as close to the ground as possible.
One step towards automation of low altitude flight missions is a path planning framework that yields a flyable trajectory from the start to the goal of the mission, optimized concerning multiple criteria, e.g. with regard to visibility.

Research concerning path planning for low-altitude flight for rotorcraft dates back to the 1980s. In (Ref. 3) Menon et al. propose to use optimal control theory to create nap-of-the-earth trajectories. Using the helicopter's heading and speed as control variables and optimizing with regard to terrain masking and flight time, optimal trajectories are calculated based on arbitrary initial conditions.

Several researchers have investigated low altitude flight in the context of local planning. Mentionable research comes from Johnson and his colleagues. They have compared different methods for online guidance for nap-of-the-earth flight: a vertical method to create a suitable height profile, artificial potential fields and utilization of motion primitives. The methods have been examined in simulation and tested in flight (Ref. 4). Dikarew has proposed model predictive collision avoidance, projecting the trajectories from discretely sampled control inputs, choosing the one with the least cost (Refs. 5, 6).

Goerzen et al. contributed in several publications to the field of path planning for low altitude helicopter flight. They use an A*-like search on a sensor based three-dimensional risk map to generate a heading and velocity command. Continuously, the risk map is updated using LADAR sensor data and the path is recalculated (Ref. 7). They were able to demonstrate the system during flight tests on an EH-60L as well as the RASCAL JUH-60A. In recent work they integrated a hierarchical planning approach by adding an additional layer of long-term planning to guide the existing one (Ref. 8).

This work also presents a framework for low altitude path planning, more precisely for contour flight planning. After preprocessing the planning basis, it employs a hierarchical planner, consisting of a euclidean global planner and a modified Vector Field Histogram (VFH) algorithm. VFH is a local, reflexive path planning algorithm, that maps the nearby environment onto a polar histogram and chooses the obstacle-free direction closest to the current goal. The vehicle moves in the determined direction and the process repeats (Ref. 9). VFH+ has been enhanced to further consider approximate vehicle dynamics, by masking directions based on the vehicle's assumed trajectories (Ref. 10). Finally, VFH* combines the method with an A* search optimizing the planner output for several consecutive time steps (Ref. 11). A three-dimensional variant of VFH has been introduced by (Ref. 12). Here, the possible directions are mapped to a sphere. For better performance a quad tree is used.
Previous work has implemented VFH for rotorcraft: (Ref. 13) has implemented it for UAV obstacle avoidance and Jin et al. propose a framework for autonomous low altitude helicopter flight, using 3D-VFH for obstacle avoidance (Ref. 14). In this work however, assuming that we do not want to pass below obstacles, a 2.5-D variant suffices. Thus, Terrain Vector Field Histogram (TVFH) is introduced. Furthermore, in order to guide the purely local VFH algorithm, a global planner is used. Ma et al. have already demonstrated a hybrid navigation scheme, using VFH as a local obstacle avoidance, while using A* for global planning (Ref. 15). Expanding on this idea, in this work VFH is already used while planning offline, allowing for more relaxed safety margins during global planning.

The following paper is structured as follows: The next sections provide insight in the different planning steps: These are the data preprocessing, global path planning and local path planning. For local path planning, the method presented in this paper is explicitly compared to the original VFH algorithm found in literature. Paths generated by the framework at hand are discussed and compared to purely globally planned paths. Finally, a short conclusion and suggestions for further work are given.

DATA PREPROCESSING

The path planner requires information about the objective, i.e. the destination or specific waypoints it should navigate to, but also about the constraints, i.e. the locations to avoid. The most obvious constraints are, of course, terrain and obstacles. Those can be referred to as "hard constraints". In simple words, hard constraints are non-negotiable, because a violation of those would lead to a collision. These must be respected under any circumstance. But, purely terrain-based planning might not be sufficient for low-level operations. Therefore, another class of constraints needs to be defined, the "soft constraints"; Rules or strategical knowledge to be considered in the planning. For example, the helicopter should stay away from populated areas, cross power lines at the poles, climb hillsides transversely, et cetera.

The path planning is supposed to deliver an optimal mission trajectory derived from a priori knowledge. It can also be used to re-plan a mission during flight when the tactical situation changes. However, for sudden changes in the nearby environment, i.e. new obstacles detected by onboard sensors or other threats like enemy fire, the aforementioned model predictive collision avoidance will guide the helicopter (Refs. 5, 6).

As basis for the planner a Digital Surface Model (DSM) is needed. In our case this data was generated as pseudo DSM by a combination of a Digital Terrain Model (DTM) and Open Street Map (OSM) data. For the DTM the Shuttle Radar Topography Mission with 1 Arc-Second Resolution (SRTM1) with approximately 30 m lateral resolution and one meter height resolution was used. Forests, trees, obstacles, and urban areas were added either with the OSM height where available or with a constant offset. To provide an euclidean map the data was projected to Universal Transverse Mercator (UTM) raster. This data serves as the hard constraints for the planning.

As additional costs some soft constraints were generated. The results presented in this paper used urban areas from OSM as areas to avoid and a modified viewshed analysis that delivers the ratio of raster points that are in the line-of-sight of the target raster plus 50 ft, the helicopter's nominal height above ground. It is called terrain protection map. This map is not optimized yet, but already provides a very good risk map for strategical planning. It is envisaged to also use the type of terrain, i.e. agricultural, urban or forest area, as weighted cost of visibility. Due to the computational cost the range of the modified viewshed was limited to 5 km radius (see figure 1). The effect of using DSM instead of DTM for the terrain protection can also be seen in figure 1. This effect enables the planner to find a routing alongside of forests to add natural protection to the path. The constraints described above are sufficient for testing but not complete. Other risks will be added to the map iteratively over time.

The OSM data of urban areas were modified as well. The geographical region of the test area is rurally populated with

many very small settlements or standalone buildings. Additionally, the resulting polygons need to be rasterized to the corresponding UTM grid. In a first step the polygons were grown by some extent to eliminate the fact that some OSM data of urban areas exclude streets providing numerous very small polygons for one village. Thereafter, the polygons were combined and shrunken again. The data was than split into two groups – small villages and standalone buildings on the one hand and larger settlements on the other hand. Both groups receive additional safety areas – adequate and desired distance (see figure 2). For villages these distances are 30 m and 60 m, for the others 200 m and 400 m. This was necessary because of the mentioned rural population. Otherwise, appointing the same safety area to the small settlements would flood the risk map. During planning, the encoded semantics, i.e. certain combinations thereof, are mapped to a single cost value.

Figure 1. Processed data - protection from terrain
Left: Natural protection generated by a viewshed-algorithm from DTM (normalized: red=1, full protection, blue=0, fully visible)
Center: Satellite view by Google Maps of the same region
Right: Natural protection generated by a viewshed-algorithm from pseudo DSM (normalized: red=1, full protection, blue=0, fully visible)

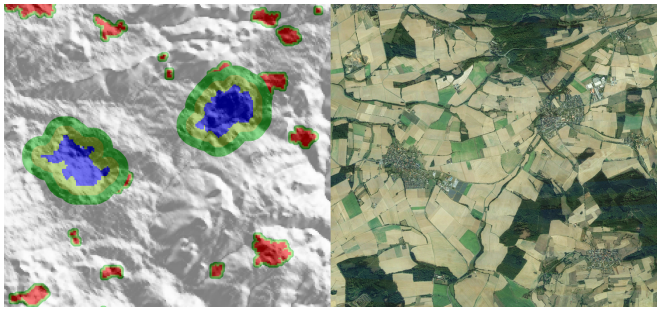

Figure 2. Processed data - avoiding settlements
Left: Semantic map of urban areas with safety distance (green) for small villages (red) and larger settlements (blue)
Right: Satellite view by Google Maps of the same region

PATH PLANNING

In path planning, challenges arise when paths for vehicles with complex dynamics need to be planned. To estimate the vehicle's path, its dynamics, i.e. as an approximation, its kinematics can be directly included into the global path planner e.g. by using Dubin's paths (Ref. 16). Alternatively, the globally planned path can be smoothed in a second step, for example by using splines (Ref. 17). Since the smoothed path differs from the original one, sufficient safety margins need to be considered during global planning, to guarantee feasibility after smoothing.

The path planning algorithm used in this work separates global and local path planning. Similar to (Ref. 8) a hierarchical planning approach is employed. Precisely, it combines a reflexive VFH-based local planner with a global path planning algorithm, in this case a Batch Informed Trees (BIT*)-algorithm. The global planner yields a coarse, however globally optimized, polygon course. The path is then smoothed by the local planning algorithm by letting a helicopter model move along said waypoints, thus creating a trajectory with mostly steady control input. Since the VFH-algorithm also avoids obstacles, the safety margin of the global planner can be reduced. The global and local algorithms will be touched upon in the subsequent paragraphs.

Helicopter Model

The framework proposed by this work requires a stabilized helicopter with decoupled axis as foundation. Here, the Command Model (CM) of the DLR's research helicopter Active Control Technology / Flying Helicopter Simulator (ACT/FHS), a highly modified EC135, is used (Ref. 5). In recent work, it has been compared to a non-linear simulation of the same helicopter, proving its validity (Ref. 6). This model reflects the helicopter's closed-loop dynamics, with attitude command and attitude hold for the cyclic inputs and rate command and height hold for the collective. This allows for three separate controllers for local planning: the longitudinal cyclic input controls the velocity, the lateral cyclic input controls the heading and the collective controls the Rate of Climb (ROC), derived from the climb angle. Pedals are implicitly utilized during turn coordination. The CM already features attitude control, attitude hold controllers for the roll and pitch axis. Thus, for the longitudinal cyclic input, a simple proportional controller suffices. Since for a coordinated turn the yaw rate increases with the bank angle, for the lateral cyclic input a proportional controller with rate damping is used. For the collective input, theCM displays the behavior of a heave rate controller with height hold. To account for stationary control error, a full proportional-integral-derivative (PID)-controller with additional anti-wind up is employed.

Since the ROC is solely controlled via the collective, the ROC is assumed to be the limiting path parameter, resulting from a maximum acceptable rotor torque and a maximum available engine power. While in further iterations of this work the maximum ROC shall be approximated by a model e.g. based on the helicopters power curve, in this paper a fixed value is specified as flight path angle constraint.

Global Path Planning

Globally, a path planning algorithm is employed that plans a feasible path from the start position to a single goal, while

optimizing with respect to the given cost function. A three-dimensional, euclidean state space is used, while only enforcing minimal constraints. This allows for a computationally cheap distance function and edge interpolation.

In this work a BIT*-algorithm (Ref. 18) is used. Simplified, this algorithm works by batch-wise sampling in the given state space and building a search tree based on the resulting implicit graph. Once the goal vertex is included in the search tree, i.e. an initial solution has been found, the algorithm keeps improving the path by refining the graph. This is done by informed sampling, either directly or by rejection sampling, thus reducing the search space to only consider samples, that can possibly improve the solution. The algorithm stops once the termination condition (e.g. a given planning time) is reached, yielding the best solution found so far.

As a path planning framework and for the basic BIT* algorithm implementation, the Open Motion Planning Library (OMPL) (Ref. 19) is utilized. Samplers, optimization objectives and validation functions, as well as modifications to the algorithm itself are implemented to adapt to the given problem and discussed in the subsequent paragraphs.

Since our objective is to plan for contour flight, a validation function has been implemented to ensure that all states and edges are located inside a specified height envelope above the terrain. For the lower bound of this envelope, a safety margin can be specified, thus also guaranteeing that each state is collision free. It has shown that overall better planning results can be achieved with looser constraints regarding the upper bound of the envelope, since this allows to traverse steeper edges in the terrain and to travel over narrow valleys. This of course leads to locally increased visibility but may result in an overall better solution. To reduce non-viable samples and edges, the sampling along the z-axis is restricted to only throw new samples inside the viable envelope. The informed sampling of the BIT*-algorithm is adapted accordingly. It is restricted to the XY-plane, respectively using a two-dimensional hyperspheroid, i.e. an ellipsis, to determine the area where new samples can be generated. For better performance, collision-checking is done by checking discrete states interpolated along a given edge with configurable resolution. Furthermore, to account for the helicopter's power, edges with climb angles corresponding to a non-feasible ROC are discarded.

The path is optimized concerning multiple objectives. Besides its length, the cost function also considers the path's overall visibility, the costs resulting from the semantic map, i.e. the costs from traversing different terrain types, and its height over ground compared to a reference, to encourage flight at the specified height band. Thus, the cost of an edge can be expressed by equation 1.

$$\begin{aligned} \mathrm{cost}_{\mathrm{edge}} &= \int_S ds \\ &+ \tilde{c}_{\mathrm{vis}} \int_S \left((1 + \tilde{c}_{\mathrm{vis},h} \max(h(s) - h_{\mathrm{ref}}, 0)) \cdot \mathrm{cost}_{\mathrm{vis}}(s) \right)^2 ds \\ &+ \tilde{c}_{\mathrm{sem}} \int_S \mathrm{cost}_{\mathrm{sem}}(s) ds \\ &+ \tilde{c}_{\mathrm{height}} \int_S |h(s) - h_{\mathrm{ref}}| ds \end{aligned} \tag{1}$$

The edge's length S is denoted by the first term of equation 1. The additional costs are weighted by $\tilde{c}_{\mathrm{vis}}$, $\tilde{c}_{\mathrm{sem}}$, $\tilde{c}_{\mathrm{height}}$, for the costs introduced by the visibility, the semantics, and the height over ground respectively. The $\tilde{*}$ is implying, that these factors are used for global planning. Those factors can be thought of as the additional distance one is willing to travel in order to avoid one unit of the associated costs. The cost of visibility and the semantic features is derived by querying the corresponding tile of the prior calculated maps. As described above, the viewshed map is calculated assuming a constant height over ground of z_{ref}. To account for higher visibility in heights exceeding z_{ref}, a linear increase in cost is modeled, scaled by $\tilde{c}_{\mathrm{vis},h}$. To especially discourage traversing highly visible areas, the corresponding term is squared. Similar to collision checking, the function is approximated by summing up the cost of discrete, interpolated points.

Since the risk maps are discrete and unstructured, only the path length is used as heuristics to make sure that the true cost of the path is never overestimated.

During testing, it showed that the algorithm tends to plan paths with acute angles between edges to pass elevations in a serpentine like fashion to adhere to the maximum climb angle. The local planner was not able to follow these waypoints due to the helicopter's minimum turn radius, leading to large detours or even non-viable paths during the second planning step. To overcome this behavior, a tree insertion condition has been implemented (see figure 3): If a vertex is to be inserted from the implicit graph into the search tree, it is first verified that the resulting angle between the new edge and its parent edge α_p does not exceed a specified maximum value. Otherwise, the edge is not inserted. If the vertex is already contained in the search tree, i.e. if a rewiring is happening, additionally it is checked whether the relative angle, between the edge formed by itself and its parent and all the edges defined by its child vertices and itself α_c are permitted.

Once the planner has terminated and a solution to the planning problem has been found, the waypoint-list is scanned for shortcuts. Here, any feasible connection between two waypoints, skipping at least one other waypoint in between and reducing the path's cost is considered a shortcut. This is realized by systematically evaluating for every waypoint, if it can form a shortcut with any succeeding waypoint. Intermediate waypoints are discarded.

Finally, the path is interpolated, resulting in a waypoint list with constant distance between the points.

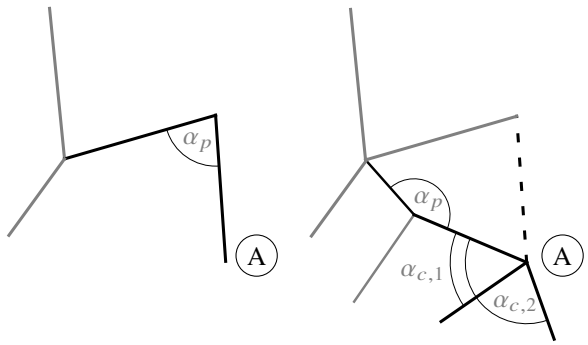

Figure 3. Tree insertion condition
Vertex *A* shall be inserted into the search tree; on the left for the first time, on the right again, as rewiring. The vertex is only inserted if the angle to its parent edge α_p, as well as the angles to all its children α_c are suitable.

Local Path Planning

Based on the output of the global planner, a flyable trajectory is generated and locally optimized. The algorithm follows the terrain by mapping the climb angles needed to pass over nearby terrain to a polar representation. The following subsection describes and discusses the used algorithm.

Terrain Vector Field Histogram During local planning, the trajectory is generated by employment of a highly modified VFH-algorithm (Refs. 9, 10). The subsequent paragraphs describe the process of the original VFH+ algorithm and highlight the differences implemented to adapt to the problem of contour flight planning. For better differentiation, the modified algorithm shall be called Terrain Vector Field Histogram (TVFH)+. The different steps are visualized in figure 5.

In the first step in the original VFH, a circular area around the vehicle, the active window with radius w is considered. The active window is partitioned into K segments of angle $\alpha = 2\pi/K$. The algorithm works on a two-dimensional obstacle map. Depending on its distance to the vehicle and a certainty measure (e.g. accounting for the number of sensor points in a given tile), each tile of the obstacle map is assigned a value. The sum over all tile-values contained in an angular sector represents this sector's obstacle density. To account for the vehicle's width, the obstacles are circularly inflated. The process results in the *primary polar histogram*, mapping the obstacle distribution in the active window to a one dimensional polar representation around the vehicle.
Since this work accounts for helicopter contour flight, it shall be assumed that all points of the height map can be passed over. Thus, an "obstacle" is defined as an elevation the helicopter can not traverse without exceeding the maximum climb angle, i.e. the maximum flight path angle at the given speed. Instead of calculating the obstacle density, the algorithm introduced by this works maps the minimum climb angle needed to traverse a given direction to the corresponding angular segment, creating a *polar climb angle histogram*. The flight path angle γ_k of one sector k is defined as the maximum climb angle of all cells contained:

$$\gamma_k = \max\left(\arctan\frac{h_{(i,j),k} + h_{\text{safety}}}{d_{(i,j),k} - r_{\text{safety}}}\right) \tag{2}$$

The index $(i,j),k$ denotes the position of a tile in the elevation map, contained in the angular sector k. As seen in figure 4, each tile is inflated by a safety radius r_{safety}. $d_{(i,j),k}$ is the distance from the tile to the helicopter and $h_{(i,j),k}$ is the elevation of the tile. As implicated in figure 4, a tile is considered to be contained in the angular sector k, if the following condition is true:

$$\begin{aligned} k\cdot\alpha \in \left[\beta_{i,j} - \beta_{\text{safety},(i,j)},\ \beta_{i,j} + \beta_{\text{safety},(i,j)}\right] \vee \\ \beta_{i,j} - \beta_{\text{safety},(i,j)} \in \left[k\cdot\alpha - \alpha/2,\ k\cdot\alpha + \alpha/2\right] \vee \\ \beta_{i,j} + \beta_{\text{safety},(i,j)} \in \left[k\cdot\alpha - \alpha/2,\ k\cdot\alpha + \alpha/2\right] \end{aligned}$$

With $\beta_{i,j}$ being the azimuth of the grid cell in the global coordinate frame and $\beta_{\text{safety},(i,j)} = \arcsin(\frac{d_{i,j}}{r_{\text{safety}}})$ being the angle enlargement due to the safety radius r_{safety}. A vertical safety margin is realized, by adding a safety distance h_{safety}. Assuming coordinated turns with at least the minimum turn radius, the sections being immediately to the left or right to the helicopter can be ignored. To take this into account, the height map is truncated at a given radius for all cells exceeding a specified angular distance relative to the current heading, as seen in figure 4.

In the second and third step, VFH+ generates the *binary polar histogram*, and the *masked binary polar histogram*. The *binary polar histogram* indicates whether an angular sector is traversable during the next time step. It is derived by checking if the obstacle density from the *primary polar histogram* exceeds a specified value. To account for the vehicle's dynamics, the *binary polar histogram* is masked subsequently: The vehicle's path is approximated by arc segments. If the vehicle needs to pass through an obstacle in order to move towards a valid direction, this direction is blocked as well. This process results in masking all otherwise feasible directions of the *binary polar histogram*, if they cannot be reached without passing through an obstacle, yielding the *masked binary polar histogram*.
For TVFH+ these two steps are reversed: First the climb angles are masked, creating the *masked polar climb angles*, then the *binary polar histogram* is determined, by comparing the climb angles to the maximum feasible climb angle. The original VFH+ assumes a constant turning radius and declares obstacles that lie within the circles described by the vehicle's approximated trajectory as blocking. This approach works well for mobile vehicles with a fixed turn radius. However, since the turn radius of the helicopter cannot be assumed to be constant, the approach is modified. The helicopter is assumed to use coordinated turns, thus the trajectory is assumed to follow a circular arc. A lower bound is given by the minimum turn radius $r_c = \frac{v^2}{\tan\phi\, g}$, with v denoting the velocity, ϕ denoting the maximum bank angle and g being the gravitational constant. However, considering the dynamics of the helicopter

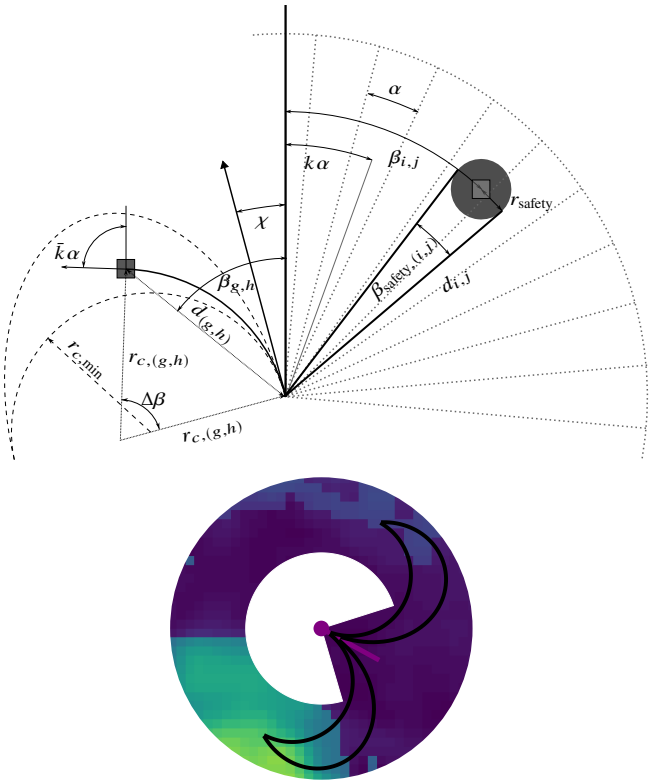

Figure 4. Active window, obstacle and approximated dynamics

Top: To the right the *active window* segmented in angular sectors of α is displayed. The tile with index i, j of the elevation map is shown as an example. Inflating it with r_{safety} results in $\beta_{\text{safety},(i,j)}$.
To the left the tile with index g, h is shown. It is located between the arcs described by the maximum and the minimum turn radius. The corresponding turn radius $r_{c,(g,h)}$ can be derived, by applying the law of cosines. Note that for better readability, the left and right are separated, while actually the active window describes a full circle and the turn radius also considers right turns.
Bottom: Elevation map masked by the active window. The helicopter's heading is implied by the purple line. The area close to the helicopter's sides and rear are omitted, since they cannot be reached following the approximated dynamics.

model, e.g. roll-ins and roll-outs, as well as the behavior of the employed controller, the radius of the resulting arc might exceed the just described lower bound. To compensate for the helicopter's dynamics, also an upper bound is defined. All radii in between the lower and the upper bound are deemed realistic and thus cells intersecting them might potentially mask the path. Recalling that the heading command is translated into a proportional controller error and an attitude controller is used, greater changes in the heading result in smaller arc radii. In other words, due to the controller employed, great heading changes result in a small turning radius and vice versa. To approximate this behavior, the upper bound of the assumed turn radius is modeled to decrease linearly with increasing arc angle: $r_{c,\max} = (r_{c,\max,0} - r_{c,\min})(1 - \frac{\Delta\beta}{\pi/2}) + r_{c,\min}$.
Using the law of cosines, for a given cell i, j the corresponding arc radius can be expressed by equation 3.

$$r_{c,(i,j)} = \frac{d_{i,j}}{\sqrt{2(1-\cos\Delta\beta_{i,j})}} \tag{3}$$

Here, $\Delta\beta_{i,j} = 2 \cdot |\beta - \chi|$ is the angle traveled along the arc and χ is the current heading. Figure 4 shows the approximated helicopter dynamics.

As seen in equation 4, the masking climb angle is calculated, considering the distance travelled along the corresponding arc, with $\Delta\beta_{i,j} \cdot r_{c,(i,j)}$ denoting the distance from a tile contained in sector $\bar{k}$ along the minimum turn radius and $\bar{k}$ denoting the angular sector corresponding to the resulting angle $\chi + \Delta\beta_{i,j}$.

$$\gamma_{\text{mask},\bar{k}} = \max\left(\arctan\frac{h_{(i,j),\bar{k}} + h_{\text{safety}}}{\Delta\beta_{i,j} \cdot r_{c,(i,j)} - r_{\text{safety}}}\right) \tag{4}$$

Starting from the sector containing the current ground track k_χ, and simultaneously marching to the left and the right (see algorithm 1, lines 3-4): If the climb angle γ_k of the current sector k is less than the corresponding climb angle considering the arc $\gamma_{\text{mask},k}$, γ_k is replaced by $\gamma_{\text{mask},k}$ and all subsequent climb angles are set to be at least $\gamma_{\text{mask},k}$ (see algorithm 1, lines 5-8).

Algorithm 1 Masking the polar climb angles

Require: $\chi, \gamma_{\text{mask}}, \gamma$ ▷ χ being the current heading

1: $i \leftarrow 1$
2: **while** $i < \frac{K}{2}$ **do**
3: $\quad k_l \leftarrow k_\chi + i$
4: $\quad k_r \leftarrow k_\chi - i$
5: $\quad \gamma_{\text{mask},k_l} \leftarrow \max(\gamma_{\text{mask},k_l}, \gamma_{\text{mask},k_l-1})$
6: $\quad \gamma_{\text{mask},k_r} \leftarrow \max(\gamma_{\text{mask},k_r}, \gamma_{\text{mask},k_r+1})$
7: $\quad \gamma_{k_l} \leftarrow \max(\gamma_{k_l}, \gamma_{\text{mask},k_l-1})$
8: $\quad \gamma_{k_r} \leftarrow \max(\gamma_{k_r}, \gamma_{\text{mask},k_r+1})$
9: $\quad i \leftarrow i + 1$
10: **end while**

In short, it is assured that the minimum climb angle corresponding to a direction also considers the minimum climb angle needed to fly an arc towards this direction. Consequently, also making the resulting *binary polar histogram* a *masked binary polar histogram.*

Finally, VFH+ determines the next direction to move towards: First feasible candidate directions are determined, based on the openings in the *polar binary histogram*. If the direction towards the goal is feasible, the vehicle is moved directly towards the goal, otherwise a feasible candidate direction is chosen based on a cost function. The lateral and vertical target directions are then fed to the helicopter-model as target values for the employed controllers.
In this work, in the lateral plane, all feasible directions are considered as candidate directions, allowing for more complex cost functions. The simplest cost function to use is the angular distance between the goal direction and the current heading $\text{cost}_{\text{goal}} = |\alpha - \chi_{\text{goal}}|$. The target direction, χ_{target} for the current step is the direction corresponding to the angular sector with the lowest cost. Additionally, the goal direction

itself is considered explicitly. As soon as a certain distance to the goal is undercut, the cost function is set to only consider the direction towards the goal. This is done to assure that the goal is reached with satisfactory deviation. In the vertical plane, the climb angle is chosen by $\gamma_{\text{target}} = \max(\gamma_{k_{\text{goal}}}, \gamma_{\text{goal}})$: If feasible, the helicopter is steered directly towards the goal, otherwise it follows the contour. The local planner terminates, as soon as the goal has been reached with satisfactory deviation.

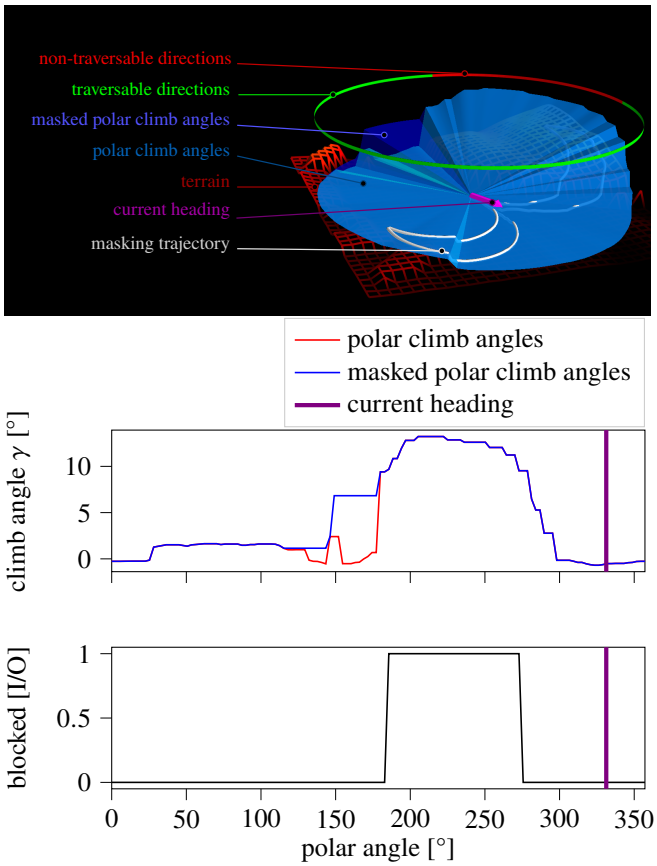

Figure 5. TVFH+: Determination of traversable directions
Based on the surrounding terrain(red, heatmap), the climb angle necessary to pass over it is calculated in every direction (turquoise surface) and masked (blue surface) based on simplified vehicle dynamics (white paths). A direction is marked as non-traversable ('halo', red) if the calculated climb angle exceeds the maximum legal climb angle, otherwise it is marked as traversable('halo', green).
The upper plot displays the *polar climb angles* in red and the *masked polar climb angles* in blue respectively. The lower plot indicates wether a direction is blocked (1) or traversable (0). Note, that the plots are relative to a global coordinate system and not to the current heading.

Note, due to the purely local property of the algorithm, the vehicle might navigate into concave obstacle arrangements and not find a feasible direction to steer towards. To circumvent these situations, the helicopter is not directed towards the goal directly but rather guided along the previously computed, globally planned interpolated waypoint list. The helicopter's position is projected onto the waypoint list by determining the closest waypoint. The waypoint closest to a specified distance ahead of the helicopter is set as the target of the current time step. Figure 6 shows a trajectory generated by the algorithm described so far, using only the angular distance towards the goal as cost function. In comparison, the trajectory generated by the same local planner, but guided by a globally planned waypoint list, is shown to produce a more efficient path while solving the same planning problem.

The system as described so far solves the problem of path smoothing: On the one hand, the local planner avoids obstacles, and thus the constraints on the global planner can be relaxed, on the other hand, by simulating the CM, the resulting path and its are inherently steady.

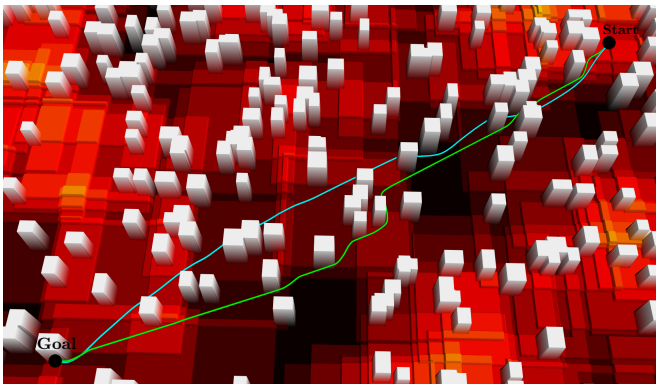

Figure 6. Locally and globally planned trajectories
Trajectories in a randomly generated environment. The purely locally planned trajectory (green) steers directly towards the goal. In contrast, the combination with a global planner results in a more efficient trajectory (blue).

Cost Function In the previous section, only the angular distance to a single target direction, is considered to guide the helicopter, thus each angular sector is assigned a cost proportional to the angular difference to the current heading: $\chi_{\text{target}} - \chi_{\text{current}}$.
For local optimization however, more factors can be considered. To include risk, derived from the semantic map and the viewshed map, for each angular segment the risk values of all cells contained are taken into account. (see equation 5).

$$\text{cost}_{\text{risk},k} = \frac{\sum_k^K \left(n_{(i,j),k}\left(c_{\text{sem}} \cdot \text{cost}_{\text{sem},(i,j),k} + c_{\text{vis}} \cdot \text{cost}_{\text{vis},(i,j),k}\right)\right)}{\sum_k^K \left(n_{(i,j),k}\right)} \quad (5)$$

With $n_{(i,j),k} = 1 - d^2_{(i,j),k}/w^2$ being a weight factor, inspired by the calculation of the obstacle density in the original VFH+ algorithm. Normalization is necessary, to account for differing numbers of cells contained in each sector.
Climb angles exceeding the one proposed by the global planner are discouraged: $\text{cost}_{\text{climb, rel},k} = c_{\text{climb, rel}} \max(\gamma_k - \gamma_{\text{target}}, 0)$. Furthermore, climbing in general can be discouraged: $\text{cost}_{\text{risk},k} = c_{\text{climb}}\gamma_k$, however this may lead to strong deviations from the globally planned path.

The area lying to the helicopter's rear has already been passed and the area lying to the helicopters sides is about to be. To take

this into account, the cost factors are weighted depending on the angular distance to the current heading, i.e. to the current target heading. Subsequently, equation is 6 used to weight the polar cost factors according to their polar position. β denotes the angle of the polar risk; for a certain sector k it becomes $\beta = k\alpha$. The offset $c_{p,\text{ofst}}$ in the denominator is added to eliminate singularities.

$$c_p = \frac{1}{|\beta - \chi|^3 + c_{p,\text{ofst}}} \tag{6}$$

This weighting leads to diminishing costs at the helicopter's rear. In an area of high and constant risk this may lead to the algorithm preferring directions at the helicopters rear if not properly tuned. To eliminate this behavior, the mean over all segments is subtracted, leaving the sectors with the least risk with a negative sign.

Finally, the costs are calculated for each segment and summed up. The feasible direction with the lowest cost is used as target for the next time step.

Dynamic control input smoothing Until now, the target angles fed to the algorithm are inherently non-steady due to the discrete nature of the angular sectors and the globally planned waypoint list. This may lead to aggressive control behavior. To overcome this, this work proposes to add an additional integrator between the local planner's output and the control input. A dynamic target heading indicator system, inspired by a torsional spring as depicted in figure 7 is employed.

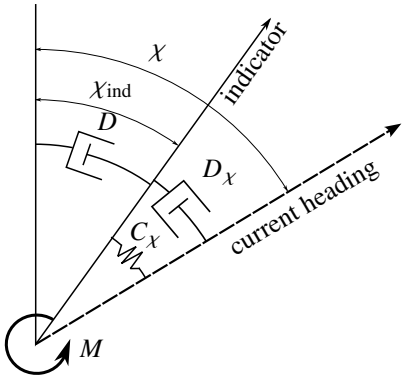

Figure 7. Dynamic target heading indicator system
Dynamic system inspired by a torsional spring. Each time step the "torque" is applied and the position of the target heading indicator χ_{ind} is calculated.

Instead of using the target provided by TVFH+ directly, a "torque" M is applied to the dynamic system. The target heading indicator, i.e. the current position, of the system with inertia I_{ind} is then forwarded to the controller. A damping force $M_D = -D \cdot \dot{\chi}_{\text{ind}}$ is applied to assure stability of the system. To limit the rate of χ_{ind} relative to the helicopter's current heading χ, an additional damper $M_{D,\chi} = -D_\chi \cdot (\dot{\chi}_{\text{ind}} - \dot{\chi})$ is applied. A torque $M_{s,\chi} = C_\chi \cdot (\chi_{\text{ind}} - \chi)$ is introduced to avoid excessive deviations of the target heading relative to the current one. The resulting dynamic is expressed in equation 7.

$$I_{\text{ind}} \ddot{\chi}_{\text{ind}} = M + M_D + M_{D,\chi} + M_{s,\chi} \tag{7}$$

If the direction χ_{ind} yielded by the dynamic system is not traversable, the closest feasible direction is chosen instead. Here, a step in the controls is deemed acceptable, given that the alternative may result in a collision, i.e. a non-viable planner outcome. In this case, the target heading indicator's position is set to the current heading and its angular velocity is set to zero to avoid steering towards the same non-feasible direction again in the next time step. For the vertical axis, an identical system is put in place. However, it is ensured that the commanded climb angle never subceeds the needed climb angle.

The cost introduced in the previous sections can be applied to the dynamic system. For example, to consider the angular distance towards goal, the angular difference to the current heading can be considered as an elastic force: $M = C(\chi_{\text{target}} - \chi_{\text{ind}})$. In other words, the target heading indicator is being controlled towards the target direction, i.e. the TVFH+'s output by a de-facto proportional-controller.
In general, if the cost of a sector is negative, i.e. that direction is encouraged, the helicopter should be pulled towards it, thus an elastic force is added: $M_{\text{cost}} = \text{cost}_k(k\alpha - \chi_{indicator})$. If it is positive, the helicopter should be repelled. Here, it might seem intuitive to directly apply torque to the dynamic system. However, when considering the zoomed-in section of figure 8, as soon as the helicopter starts moving towards the goal, the repelling torques need to start cancelling out, so the overall repelling torque decreases. For this to work, the function needs to be symmetric.

Figure 8 shows the outputs of a purely local planning in a heavily simplified viewshed map without obstacles (i.e. with no elevation) for increasing weight factors c_{vis}. Note, how by increasing the corresponding weight factor the local planner is more biased to follow the map's layout. However, the path length also increases with increasing c_{vis}. This means, that even if the risk per distance decreases, the total risk (i.e. the risk integrated over the path length) increases, once again stressing the necessity of a global planner.

Especially in highly non-steady environments, the dynamic system employed tends to oscillate. This can be mitigated by adapting the damping or inertia of the target heading indicator system, or by using the cost function, to calculate one single target direction as described above. Another option that has proven satisfactory results is to run the local planner multiple times, with decreasing dynamic window sizes and guided by the local planner's previous solution. The green trajectories in figure 8 are planned for only one iteration of the planner. After the planner has terminated, the output is treated like a globally planned waypoint list and the planner is run again; in this example for a total of two times. The resulting paths (orange) keep greater distance to the costly area, due to decreased relative angles to the temporary goal state. In the bottom of figure 8, the resulting bank-angles are displayed in the frequency domain. Note, how the path from multiple runs produces a much smoother output.

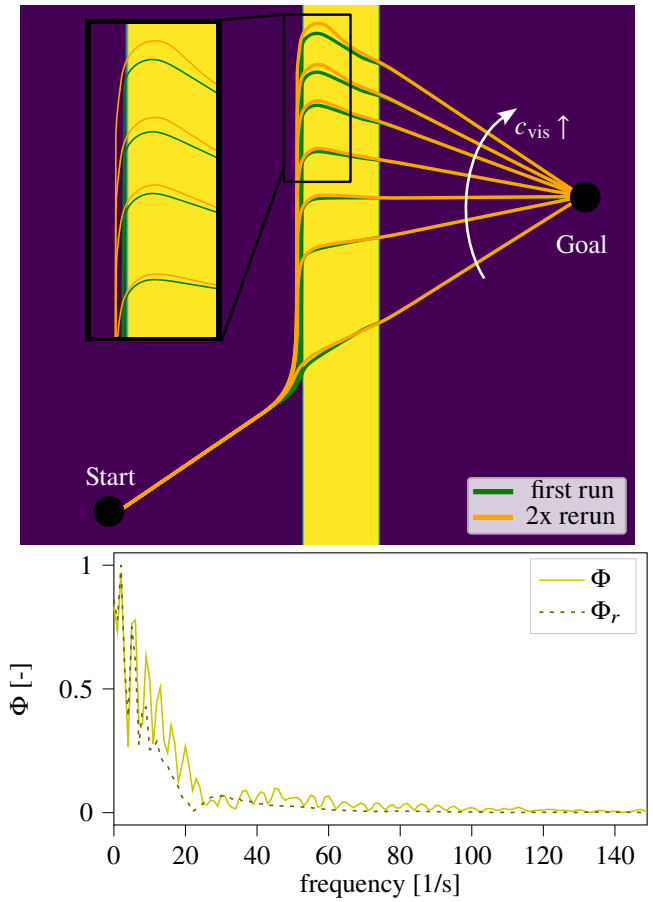

Figure 8. Locally planned trajectories on simplified viewshed map after several resimulations
Top Trajectories on a generated viewshed map: The local planner avoids the area of high risk (yellow). The longer the helicopter travels along the edge, the greater the angle towards the goal, and thus the elastic force increases. Eventually, the elastic force becomes greater than the repelling force and the helicopter moves towards the goal. After the first planning (green) the local planner is employed twice more (orange), using the previous output as guidance.
Bottom Normalized initial (Φ) and re-simulated (Φ_r) bank-angle in the frequency domain. Employing the local planner multiple times smooths the target angles and thus the resulting helicopter movement.

Computation time

The total planning time is composed of the global planner's and the local planner's planning time. The BIT* algorithm used for global planning is configured to terminate after a given time, with longer planning time potentially allowing for better solutions. In general, the planning time needed by the global planner to generate satisfactory waypoint lists increases with the problem's complexity. Since the CM has to be simulated from start to goal, the time needed for local planning is depending on the path length. Furthermore, when increasing w, the size of the active window, or K, the number of angular sectors, one time step becomes more computationally expensive and thus also increases the planning time. Table 1 shows the time needed by the local planner to simulate one kilometer, i.e. 500 time steps for a fixed time step of 50 ms, for different configurations, averaged over 100 runs each. The test have been performed with on an Ubuntu laptop with an i7-1185G7 CPU.

Table 1. Local planning time for 1 km

Planning time [ms]				
K =	32	64	128	256
α =	11.3°	5.6°	2.8°	1.4°
w = 300 m	103.4	125.9	167.2	217.3
w = 350 m	125.92	149.5	231.8	296.5
w = 400 m	158.0	202.2	284.0	378.3
w = 550 m	197.0	239.6	328.2	473.4
w = 500 m	236.3	283.7	384.0	550.0
w = 550 m	288.6	334.9	456.5	618.7
w = 600 m	326.2	405.3	492.8	694.0

DISCUSSION AND RESULTS

The described methodology has been put to test on a scenario placed in the Weser Uplands, a hillscape located in central Germany with elevations up to just over 500 m above sea level. The following paragraphs discuss the resulting trajectories, qualitatively discuss the impact of the different planning parameters, compare the results to a naive alternative, a purely global planner using Dubin's paths, and discuss the method's shortcomings.

General Discussion

Figure 9 shows a trajectory planned by the proposed framework on the DSM, the viewshed map and the semantic map. Additionally, the heading χ, the target heading χ_{target} and the bank angle ϕ, as well as the height profile are plotted over the distance travelled in figure 10. The overall length of the trajectory is 18.7 km. The target velocity of the planned trajectory is 40 m/s. In this example, the global planner was given 40 s; the local planning algorithm needed an execution time of 3.6 s. The radius of the active window w is set to 350 m and the number of angular segments K to 64. Here, the parameter set has been tuned using sensitivity analysis and by experimentation. The zoomed in sections show how the local planner deviates from the global path to avoid terrain and for local optimization: In the viewshed map, one can notice how the local planner deviates from the waypoint list to stay in the center of the valley, i.e. the area with the lowest visibility. In the semantic map, some strong detours of the local planner become reasonable, since it avoids settlements.

To verify that the framework works well with a non-linear helicopter model, paths planned by the method proposed have been transferred into the Air Vehicle Simulator (AVES), the DLR's in house simulator. It shows that the planned paths are indeed traversable with the available engine power for climbing. Furthermore, the available controllers were able to follow the trajectories planned.

Analysis of local optimization

Separate planning objectives can be stressed, either by directly altering the associated cost factors, or indirectly by changing

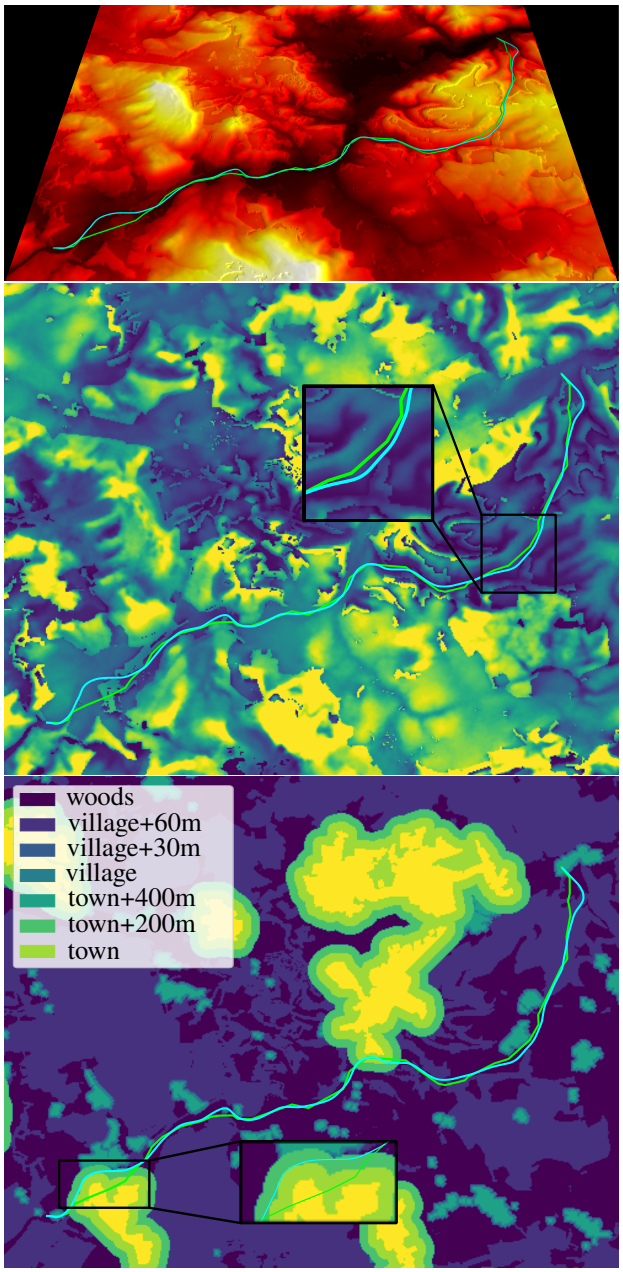

Figure 9. Path planned by the proposed planner on different map layers
Light Green: euclidean waypoints, Light Blue: model based trajectory.
Top: Digital Surface Model
Mid: Viewshed map
Bottom: Semantic map
Note how the local planner differs from the waypoints proposed by the global planner to optimize the path locally.

dependent values. The following measures have been analyzed: the path length L, the overall cost due to visibility cost_{vis} and semantics cost_{sem}, the trajectory's roughness, indicated by the bank angle integrated over the path length $\int_L \phi dl$, and the averaged height over ground h_{agl}. This section discusses the effect of local optimization and the influence of the different weight factors on the just defined measures for local and global planning.

Tests have been conducted to examine how the local optimization affects the path. For random sets of start and goal points, a global waypoint list is planned. Based upon, local planning is executed twice. One time the planner is configured to only follow the global waypoint list, the other time it is configured to also optimize the path locally. For the locally non-optimized runs, all cost factors of the local planner have been set to zero; for the locally optimized runs, a promising parameter set has been used. Averaged over 100 runs, table 2 shows the change of optimized over non optimized measures in percentage terms as well as the standard deviation. It shows, that the cost due to visibility cost_{vis} and semantics cost_{sem} as well as the height over ground h_{agl} can be reduced overall. Contrary, the path length L increases slightly and the path becomes more rough, as indicated by the integrated bank angle $\int_L \phi dl$. The high standard deviation compared to the average indicates that the improvement to be achieved with the given parameter set is strongly dependent on the specific planning problem. For example, in path segments with over areas with no gradient in the viewshed or semantic map, the local planner might not be able to optimize the solution. In some cases the local planner even increases the path costs. For more insight the reader is referred to the methods current shortcomings.

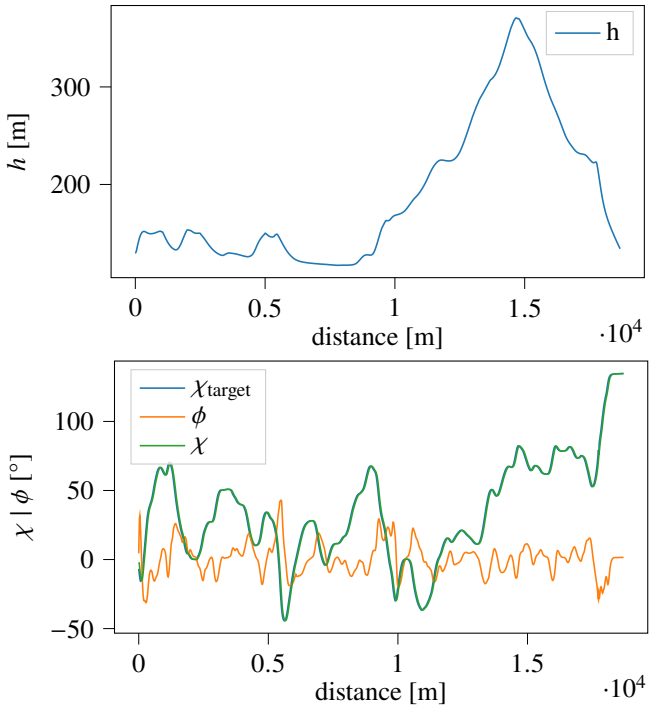

Figure 10. Height and heading of the planned path
Top Height profile of the planned path
Bottom Heading, target heading and bank angles of the planned path

Table 2. Optimizing vs. non-optimizing local planning

$\%L_{\text{opt}}$	$\%\text{cost}_{\text{vis}}$	$\%\text{cost}_{\text{sem}}$	$\%h_{\text{agl}}$	$\%\int_L \phi dl$
+2 ±1%	-2±4%	-4±2%	-9±9%	+45±16%

Qualitatively, increasing the cost factors c_{vis} and c_{sem} decreases the risk due to visibility, respectively due to the se-

mantics. However, when chosen too large, strong deviations from the global path can be observed, leading to increasing path length and thus to increasing overall costs. Similar behavior is shown from c_{climb} and $c_{\text{climb, rel}}$. Furthermore, when directly applying a torque to the target heading indicator system as described in the previous section, large values lead to rough control behavior and even to overshooting, i.e. oscillation of the target heading indicator, and thus also to traversing unfavorable terrain. To smooth the target heading indicator's movement, the dampening factors can be increased, however choosing them too large, leads to sluggish system behavior, making it unsuitable for local optimization. As described above, another method to smooth the path is to run the local planner again based on the previous output; this of course increases planning time. As stated in the previous chapter, increasing w, the radius of the active window, increases computation time. Furthermore, it determines how far the local planner looks ahead: If chosen too small, it might cause the helicopter model to collide with the terrain, leading to a non-viable planner outcome. On the contrary, a large active window leads to unnecessary climbing in certain situations.
When decreasing the distance to the globally planned waypoint that is chosen as the current target, the heading error due to lateral deviation from the globally planned waypoint list increases. Thus, also the corresponding cost increases and as a result the waypoint list is followed more strictly. Contrary, if the distance is increased, the initial path is smoothed out and also allows for a larger margin for local optimization. In other words, when no local optimization should take place, i.e. when the local planner should just follow the globally planned waypoints, it has proven best to choose a small distance to the current goal. This ensures that the waypoints are followed with minimal deviation.

During global planning, increasing a cost factor improves the resulting path with respect to the corresponding measure. However, due to the form of the cost function for global planning, the path might deteriorate with regard to the remaining measures. For example, the path's length will increase in order to reduce visibility.

Comparison of the method to a purely global planner

In order to better contextualize the planner's performance, a global planner using a Dubin's state space is employed as a naive baseline. The proposed planning framework is then compared to the baseline planner.
The baseline planner is set up in the same way as described in the previous chapter, however the proposed edge insertion condition is not needed here, since the planner already yields G1-steady paths. The path's height profile is simplified to a linear interpolation of the height along its length, while still adhering to the given climb angle constraint. Assuming coordinated turns, the Dubin's path's minimal turning radius is determined based on the maximum bank angle and the given velocity.
During planning, the global planner of the proposed framework, using a euclidean state space, processes ~90% more edges per time unit, compared to the baseline planner using Dubin's paths. This is caused by the computationally more expensive edge calculation, as well as the more expensive queue ordering. Furthermore, the resulting state space is four dimensional (three spatial dimensions and the directly sampled heading), reducing the probability of a new sample to improve the solution.

To compare the performance of the planner proposed by this paper with the baseline, the proposed framework is executed until 100 paths with randomized start and goal points are calculated successfully. Afterwards the purely global baseline planner is run on the same planning problems. In order to use Dubin's paths, the heading of the start and goal state have to be given explicitly. Here, it is calculated from the first two, respectively the last two waypoints of the solution yielded by the proposed framework. The baseline planner was given additional planning time, equivalent to the planning time needed by the local planner. Furthermore, since the maximum height of the local planner is not bound, the height envelope of the baseline planner has been relaxed. The relative comparison of the resulting trajectory measures are shown in table 3 in percentage terms. The trajectory roughness is not considered for this test, since the baseline planner only coarsely approximates the helicopter's kinematics. Instead, the ratio in which the baseline planner did not find a solution has been recorded. It shows, that the method proposed yields significant improvement concerning the success rate, path length, as well as the cost due to risk and due to semantics. Conversely, the baseline planner manages to plan paths closer to the ground. This can be explained by the local planner adding additional height by simulating the helicopter model. Furthermore, as explained above, in the vertical axis the helicopter model is controlled to follow the globally planned path or follow the terrain if necessary.

Table 3. Proposed framework vs. purely global baseline

$\%L_{\text{opt}}$	$\%\text{cost}_{\text{vis}}$	$\%\text{cost}_{\text{sem}}$	$\%h_{\text{agl}}$	%success
-31%	-38%	-93%	+4%	66%

Shortcomings

Finally, the current drawbacks of the proposed method shall be discussed: When considering the viewshed map as e.g. seen in figure 1, it stands out that after adding height based on the semantics, some areas are highly unstructured. Gaps in the DSM result in small patches, sometimes single pixels, with very low visibility in the viewshed map. However, since they cannot be traversed at a suitable height, they contribute little to improve the path, but instead the local planner might steer towards them, thus prolonging the path and increasing overall risk. In other instances, local planning aims to avoid patches of high risk opposite to the trajectory's further progress. To further follow the globally planned waypoints, the locally planned trajectory might pass through the high-risk areas or take large detours, increasing overall costs.
As mentioned above, the TVFH+ algorithm tends to climb,

to keep the specified safety margins to the terrain. Especially in tight spaces, this leads to heights exceeding the reference height. This might be solved by using finer resolution maps, allowing for smaller safety margins.
Lastly, as indicated above, the framework overall and especially the local planner optimizes multiple, partially contradicting objectives. It has proven difficult to find parameter settings that work consistently in different planning scenarios. This problem could partially be mitigated, by adapting the planning input (e.g. assign a high visibility value to urban areas).

CONCLUSION

This paper proposes a path planning framework for helicopter contour flight. Using a DTM as well as land-use data for the area to be planned in, the input is preprocessed resulting in a viewshed map and an approximated DSM. For the actual path planning, a two-step hierarchical planner is used. In the first step, a global path is planned using an adapted conventional path planning algorithm (BIT*), creating a waypoint list connecting the start and goal position. This is possible, since the second planning step serves as path smoothing. For that, TVFH+ has been presented: a 2.5-D VFH variant, modified to especially solve the problem of local contour flight planning. Combination with a helicopter model results in trajectories with mostly steady control inputs, directly considering the helicopter's dynamics. Furthermore, the local planner is capable to deviate from the global path for local optimization. The framework has demonstrated to create sensible trajectories that have proven to be cheaper than solutions yielded by a global planner directly approximating the vehicle's kinematics using Dubin's paths.

In future work, the shortcomings discussed in the previous section shall be addressed, and the cost function will be farther optimized. To evaluate the framework, simulator studies could be performed to obtain pilot rating, for the trajectories yielded by planning with different parameter settings.
Regarding global path planning, another algorithm that would go well with the data preprocessing discussed in the previous section, however is not implemented in this work, is Probabilistic Roadmaps (PRM). This algorithm a priori creates a search graph, by randomly sampling in the given state space. Collision checks and edge costs are precomputed. Online, only a graph search algorithm, e.g. A* needs to be applied. (Ref. 20)
Furthermore, TVFH+ shall be analyzed and compared to existing approaches, especially to its three-dimensional variant. Since the resulting trajectories come as a list of full flight states rather than a mere list of coordinates, this allows for future experimentation. For example, the flight states could be used as feed-forward control for a trajectory following controller, or to inform the pilot ahead of time about expected maneuvers. Finally, the local planner's capability for short-term replanning shall be examined.

REFERENCES

1. Victor H. L. Cheng and Banavar Sridhar. Considerations for automated nap-of-the-earth rotorcraft flight. *1988 American Control Conference*, pages 967–976, 1988.

2. Charles A. Gainer and Dennis J. Sullivan. Aircrew training requirements for nap-of-the-earth flight. Technical report, NASA, 1976.

3. Padmanabhan K. Menon. Optimal helicopter trajectory planning for terrain following flight. Technical report, Army Research Institute for the Behavioral and Social Sciences, 1990.

4. Eric N. Johnson and John G. Mooney. A comparison of automatic nap-of-the-earth guidance strategies for helicopters. *Journal of Field Robotics*, 31, 2014.

5. Alexej Dikarew. Model predictive approach for short-term collision avoidance. *Proceedings of the Vertical Flight Society 77th Annual Forum*, 77, 2021.

6. Alexej Dikarew and Tobias Winkler. Rotorcraft guidance with sampling based model predictive control. *Proceedings of the Vertical Flight Society 80th Annual Forum*, 80, 2024.

7. Matthew S. Whalley, Marc D. Takahashi, Jay W. Fletcher, Ernesto Moralez, L. T. C. Carl R. Ott, L. T. C. Michael G. Olmstead, James C. Savage, Chad L. Goerzen, Gregory J. Schulein, Hoyt N. Burns, and Bill Conrad. Autonomous black hawk in flight: Obstacle field navigation and landing-site selection on the rascal juh-60a. *Journal of Field Robotics*, 31, 2014.

8. Chad Goerzen, Marc Takahashi, Matthew Whalley, Gregory Schulein, Nathan Mielcarek, LTC Wesley Ogden, James Carr IV, Carl Ott, Kevin Thomazios, and David Waldman. Hierarchal planning demonstrated in autonomous flight; integration of the risk-aware path planner with a sense-and-avoid planner on a black hawk helicopter. 2023.

9. Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Trans. Robotics Autom.*, 7:278–288, 1991.

10. Iwan Ulrich and Johann Borenstein. Vfh+: reliable obstacle avoidance for fast mobile robots. *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, 2:1572–1577 vol.2, 1998.

11. Iwan Ulrich and Johann Borenstein. Vfh*: local obstacle avoidance with look-ahead verification. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 3:2505–2511 vol.3, 2000.

12. Simon Vanneste, Ben Bellekens, and Maarten Weyn. 3dvfh+: Real-time three-dimensional obstacle avoidance using an octomap. In *MORSE@STAF*, 2014.

13. Qing Liang, Zilong Wang, Ya fang Yin, Wei Xiong, Jingjing Zhang, and Ziyi Yang. Autonomous aerial obstacle avoidance using lidar sensor fusion. *PLOS ONE*, 18, 2023.

14. Zibo Jin, Lu Nie, Daochun Li, Zhan Tu, and Jinwu Xiang. An autonomous control framework of unmanned helicopter operations for low-altitude flight in mountainous terrains. *Drones*, 2022.

15. Xihan Ma, Honglin Sun, Enwei Xu, Song Cui, Boqun Yin, and Mariam Faied. Fast adaptable mobile robot navigation in dynamic environment. *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 347–352, 2020.

16. Dubins Lester E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497, 1957.

17. Mohamed Elbanhawi, Milan Simic, and Reza Nakhaie Jazar. Continuous path smoothing for car-like robots using b-spline curves. *Journal of Intelligent & Robotic Systems*, 80:23 – 56, 2015.

18. Jonathan D. Gammell, Tim D. Barfoot, and Siddhartha S. Srinivasa. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39:543 – 567, 2017.

19. Ioan Alexandru Sucan, Mark Moll, and Lydia E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19:72–82, 2012.

20. Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics Autom.*, 12:566–580, 1996.