Infotech@Aerospace 2012
19 - 21 June 2012, Garden Grove, California

AIAA 2012-2452

# Multi-Query Path Planning for Exploration Tasks with an Unmanned Rotorcraft

Florian-M. Adolf *

**This paper presents an online multi-query path planner for exploration tasks planned onboard an unmanned helicopter. While the desirable properties of roadmaps can be exploited in offline path planning, the dynamic nature of exploration scenarios hinders to utilize conventional roadmap planners. Hence, the presented path planning approach utilizes a deterministically sampled roadmap which is dynamically indexed in real time. To address situations of partial terrain knowledge, the roadmap can be extended from its a priori dimensions towards locations of unknown terrain that are outside its original, a priori boundaries. The multi-query property of the planning system allows for combinatorial optimization such that a rapidly acting decisional autonomy is achievable during exploration flights. D*-Lite is used as dynamic heuristic path searcher in order to re-plan efficiently. Inspired by the original work on this path search algorithm, the roadmap graph is augmented with an exploration vertex which steers the exploration behavior of the vehicle. As a result, the presented roadmap guides an unmanned rotorcraft through a priori unknown urban terrain in real time.**

## Nomenclature

| | | |
|---|---|---|
| ARTIS | = | Autonomous Rotorcraft Testbed for Intelligent Systems |
| CDF | = | Cumulative Distribution Function |
| D*-Lite | = | Heuristic path search algorithm with efficient re-planning for dynamic environments |
| $d_{clear}$ | = | Minimum clearance distance of the vehicle from the closest obstacle [m] |
| $d_{sensor}$ | = | Maximum detection distance of the obstacle sensor [m] |
| FOV | = | Field of View of a sensor |
| LIDAR | = | Light Detection and Ranging range measurement sensor ("laser scanner") |
| MiPlEx | = | Mission Planning and Execution framework for ARTIS |
| NED | = | Local north ($x'$), east ($y'$), down ($z'$) coordinate system |
| $V_{k_{max}}$ | = | Maximum magnitude of the velocity command vector $\vec{V_k}$ $[\frac{m}{s}]$ |
| $\mathcal{C}$ | = | Configuration space of the vehicle |
| $\mathcal{C}_{free}$ | = | All reachable vehicle configurations with $\mathcal{C}_{free} \subseteq \mathcal{C}$ |
| $\mathcal{C}_{obst}$ | = | All unreachable vehicle configurations with $\mathcal{C}_{obst} \subseteq \mathcal{C}$ |
| $\epsilon_{free}$ | = | Number of collision free edge connections to neighbor samples of a vertex |
| $\epsilon_{neighbors}$ | = | Number of neighbor edges connections within a bounded Euclidean range in the roadmap |
| $\epsilon_{local}$ | = | Local visibility of a vertex $\frac{\epsilon_{free}}{\epsilon_{neighbors}}$ [%] |

## I. Motivation

Unmanned rotorcraft are suited to fly automated missions in urban terrain as they can turn and stop inside unforeseen narrow passages. During flight in such environments, an interruption of the control link to the operator is likely to occur and a priori obstacle data is likely to need sensor based updates. Thus, closed loop obstacle detection, flight guidance and control should be performed onboard the vehicle in order to automatically guide it safely inside terrain with unforeseen obstacles. Since unmanned aircraft are primarily

*Institute of Flight Systems, Unmanned Aircraft Department, Braunschweig, Germany, AIAA Senior Member.

utilized to capture a scene on the ground from the aerial perspective, an operator might want to assign a task specification to the unmanned system accordingly (e.g. generate an environment model of a certain search area or search volume). Within the scope of this work, we understand such tasks as *exploration tasks*. Based on the current state of the perceived environment, the vehicle needs to continuously revise previous decisions on where to fly.

## II. Problem Description

Given no a priori terrain information and before any sensor acquired terrain data is available, the exact sequence of the places to fly to and thus the waypoint coordinates remain unknown. Hence, the operator of an unmanned system cannot instruct the vehicle beforehand to which locations the vehicle should fly and in which order. Instead, temporary target positions might be necessary in order to explore a specific terrain section in more detail, while the knowledge about the environment is acquired or supplemented by obstacle sensors during flight.

However, planning the ordering of a multi-goal sequence can yield combinatorial problems that are hard to solve optimally and in real time. For $n$ waypoints, even approximation algorithms like the 2-Opt[1] generate a symmetric path cost matrix and thus require at least $\frac{n^2}{2}$ path queries to be performed rapidly.

Multi-query path planners are especially suited to enable repetitive path queries whenever the number of collision checks should be minimized (e.g. with complex 3-D terrain structures). Among many variants, the general idea behind Probabilistic Roadmaps[2] is to support such multiple queries using a reusable approximation of the free space.

However, the a priori construction of a roadmap is computationally expensive since all collision checks are performed before the first path query occurs. With more obstacles detected by an obstacle sensor, a global re-construction of the roadmap becomes likely more expensive than repetitive single path queries. Thus, conventional roadmaps are not necessarily capable to be applied to the described exploration task where the free space may change significantly over a short time period.

Moreover, even for a roadmap that allows for online replanning and graph edge updates, exploration tasks into unknown terrain can yield problematic situations. If the roadmap does not represent a feasible free space representation towards and around the target position, the target could be rendered unreachable as soon as the initial roadmap is not reachable anymore (Figure 4).

Thus, this work aims on bridging the gap between the offline multi-query path planning and the online nature of exploration tasks, by enhancing a previously developed online multi-query path planning concept.

## III. Related Work

Even small scale platforms can generate a precise terrain model from laser range data[3,4] (Figure 1(b)) or high resolution stereo cameras.[5] A handful of flight tested approaches have presented quasi 2-D mapping tasks in sparse obstacle terrain for farming and object inspection tasks.[6,7] However, performing terrain mapping tasks without tight human supervision in a true 3-D fashion, e.g. from inside a narrow passage in a priori unknown urban terrain, is an active area of research due to an increased environment complexity. Among other challenges during navigation through dense obstacle terrain with buildings, bridges or power lines, the online decision making of the vehicle has to cope with quickly increasing terrain data. Thus it needs to be able to perform repetitive path planning queries in short time spans.

A large number of approaches exists which deal with the problem of path planning, each of them suited for a particular problem instance. These approaches differ in the way they cope with the vehicle navigation capabilities, availability and complexity of environment data and computational load requirements. In general, all of them must discretize the environment, which is inherently continuous and thus not suited for almost any search algorithm. A straightforward discretization is the use of regular, rectangular meshes.[8] Quadtrees and octrees[9] are spatial extensions of rectangular meshes, which condense the discretization of larger amounts of free space into large cells, thus saving memory and reducing runtime. Other planners use the concept of a roadmap, which is a graph whose nodes are free points in space and whose edges are collision-free paths between pairs of points. Pettersson and Doherty[2] describe the generation of such a roadmap using probabilistic methods. Once a roadmap is generated, graph search algorithms can be used. Roadmap-based path planning has generally three main phases:

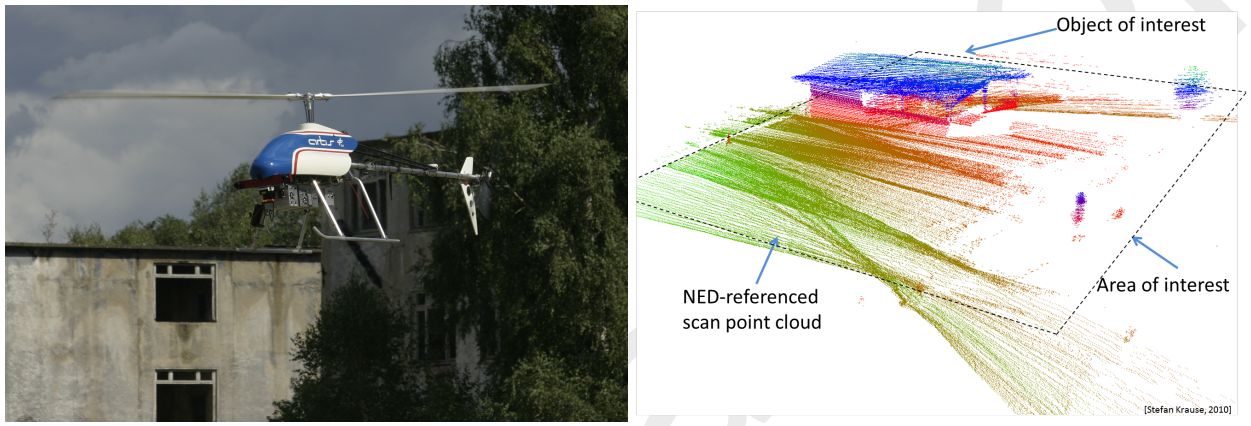
1. A pre-computation step, in which the path search graph (roadmap) is generated that represents paths through free space.

2. A query step, which consists of a search within the generated roadmap using conventional search algorithms.

3. A post-processing step can be performed to optimize a path (e.g. reduce the number of turns or perform model-based optimization).

Moreover, graph search algorithms used on roadmaps can be enhanced in such a way that they also account for limitations in the vehicle dynamics.

A survey of current path planning approaches that have been applied for unmanned aircraft guidance is provided by Goerzen et al.[10] One result of this survey is the suboptimal nature of any real-time capable planning approach. In the context of this work the primary interest is to show that multi-query path planning is not only suitable for online path planning from "A to B", but also, or especially, when rapid re-planning is required like for the exploration task.




(a) One of the ARTIS vehicles: 3 m rotor diameter, 25 kg maximum take-off weight. It is shown in an urban scenario test site and obstacle mapping equipment

(b) A terrain model example represented as a height colored 3-D point cloud that was acquired by manual flight using a LIDAR sensor[4]

**Figure 1. The ARTIS rotorcraft (left) and results of a manually acquired LIDAR terrain model (right).**

## IV. Approach

The approach introduced in this work is implemented for unmanned system platforms as exemplified by Figure 1(a). The path planning system is an extension of the previously proposed online multi-query path planning approach built into the Mission Planning and Execution (MiPlEx) framework[11] that has been successfully flight tested for non-exploration missions.

The initial configuration space is based on a modified roadmap construction phase. More specifically, in this work a configuration for a hover capable vehicle is considered to be a point in 3-D position space (x, y, z) that is enlarged to a sphere with the radius of the required minimum obstacle clearance $d_{clear}$. Based on this notion the concept of a configuration space $\mathcal{C}$ is the vector space of possible configurations and the transitions between them.[12] For a common roadmap implementation $\mathcal{C}$ corresponds to the space of collision free configurations and transitions $\mathcal{C}_{free}$ and a disjoint set $\mathcal{C}_{obst}$ with the remaining configurations.

The building blocks of the extensions to the existing roadmap-based approach summarize as follows:

- Instead of a conventional random function, a deterministic sampling source is used.[13] It is based on a Van-der-Court sequence that provides good sample discrepancy and sample dispersion (here: planner resolution), and reduces the total number of samples.

- Roadmap edges that intersect with a priori known $\mathcal{C}_{obst}$ remain in the roadmap graph. Note that the known world is not necessarily required to contain a priori obstacles, but if any is available it reduces the time needed for exploring the remaining unknown terrain.

- Based on the ideas of D*-lite[14] an exploration vertex is generated at the end of the learning phase that steers the exploration of all roadmap nodes. The exploration of the total environment will then use this vertex until all vertices have been inside the sensor field of view (FOV) at least once.

The next sections explain two building blocks that generate obstacle input for the path planner (Section A) and the path planner basis that is enhanced by this work (Section B). In sections C and D enhancements are presented that make it possible to a) start from and reach waypoint positions outside a known roadmap perimeter, and b) build a 3-D terrain map using the concept of a virtual target position node within the path planner.

## A. Obstacle Mapping

After the obstacle detection (e.g. depth image of a stereo camera or point cloud of a laser scanner), obstacle data information can be noisy and needs to be referenced with respect to the global planning environment map. Thus, it needs to be processed further, such that a consistently referenced map of the terrain can be generated. The path planning system in this work requires polygonal obstacles.

A mapping procedure generating polygonal obstacles for the path plannig system was previously presented in Ref. 15. The resulting prism-based object map is sent to the planning system in a polygonal object format. Figure 2 shows an example of flight test data from an urban-like test site. Original image recordings of a stereo camera are used to compute a depth image and merge every depth image into a map of polygonal approximations (here: prism stacks). No further polygon merging or mapping procedures need to be performed by the planning system. The obstacle map updates are sent every time when an update to a locally bounded subset of the obstacle map occurred.

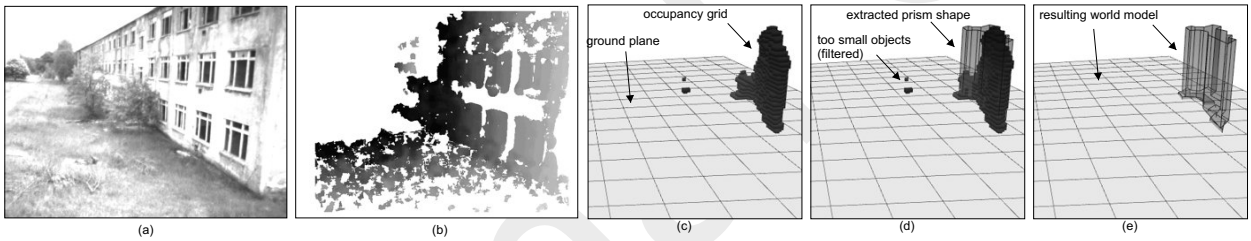



**Figure 2. Snapshot example from the input data generation process: Stereo camera-based depth images are merged into a polygonal world model and fed into the planning system.**

## B. Roadmap-Based Online Path Planning

A feasible approach to address global path planning in real time are graph-search based path planners. They save runtime by a coarse discretization of the configuration space such that the path planning is reduced to a graph search. Among the advantages of using a graph to represent the free configuration space in path planning is that graphs are suitable to handle multiple target configurations. Widely used approaches that perform this discretization are sampling-based,[16] e.g. roadmap-based approaches. For global planners, the planning process is divided into a pre-processing phase, in which the free space representation is built, and in a query phase corresponding to searches with the free space graph.

One crucial component for an online-capable roadmap path planning is a real time update of spatially indexed objects, in order to allow for rapid searches of specific spatial objects (e.g. graph edges or samples). In the used roadmap approach a 3-D voxel grid is used to index polygonal objects for collision checking, and graph edges that need to be updated in the case an unforeseen obstacle is received for free space representation updates. A KD-tree[17] is used to index the sample set. An example is shown in Figure 3. Two spatial indices are used, one for graph edges and one for the polygonal objects. Each node of the graph represents a point in free space, and each edge connecting two nodes represents a collision free path between them.

The maximum flight velocity $V_{k_{max}}$ is automatically determined by the planner (e.g. 50 m sensor range: 4.2 m/s). It is derived from the given FOV sensor geometry and the acceleration capabilities of the vehicle.[11] The determined maximum velocity allows to stop the vehicle well within the instant FOV and within a safety distance away from the obstacle sensing range. The path following method allows to track linear

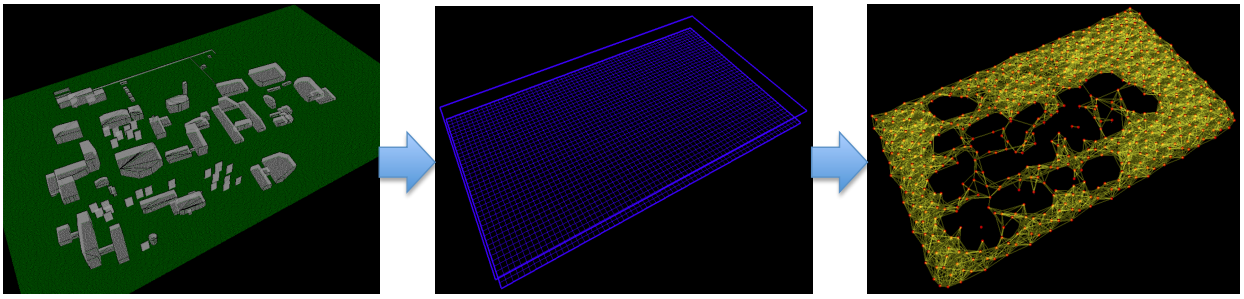


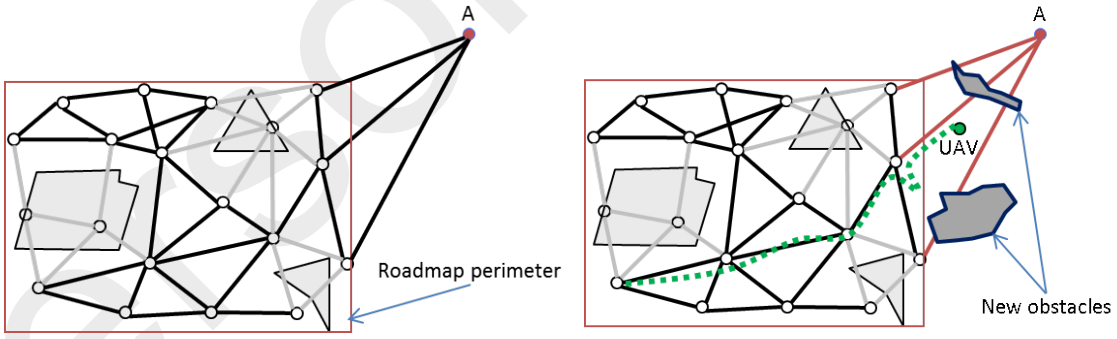
**Figure 3. Initial graph construction process based on a priori polygon models (left), world dimensioned index cells (center) pointing to each polygon, and the search graph (right) where non-free paths are filtered out.**

path segments with smooth turns towards a subsequent segment using a turn anticipation distance. This distance to turn is based on the vehicle acceleration limits while the forward flight velocity avoids unnecessary hover stops at each waypoint. Moreover, the used roadmap connection strategy accounts for the sensor and the vehicle by computing a conservative offset of from the current vehicle state to linear extrapolated stop position. Expected worst case obstacle mapping latencies and the vehicle acceleration limits are considered such that this stop position is located well within the FOV.

### C. Expansion Into Unknown Volumes

If the operator needs to let the vehicle leave the current volume represented by the roadmap, $\mathcal{C}$ needs to be enlarged such that the roadmap allows for the same path planning completeness towards and around the target position as provided by the original roadmap.

Consider the counter example in Figure 4, where the sole connection to the a priori generated roadmap is not sufficient. As soon as obstacles disable the visibility to the roadmap, the graph search on the roadmap cannot find a path to the desired target position anymore. The previously proposed approach in Ref. 11 samples around each newly detected obstacle in order to increase the number of collision free paths in the graph. However, in case of complex urban terrain obstacles, this re-sampling method can cause an unpredictable sample runtime during flight, and an unknown sampling quality (e.g. dispersion and discrepancy).



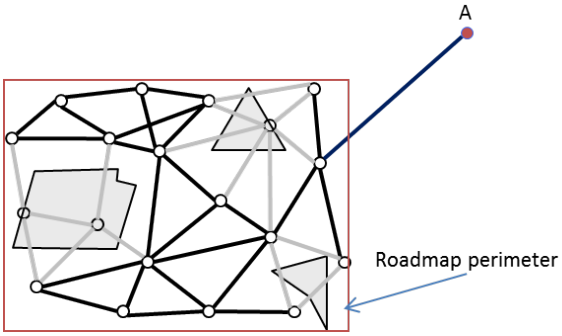

(a) Task vertex outside the roadmap perimeter

(b) Roadmap without paths to target after obstacles are detected

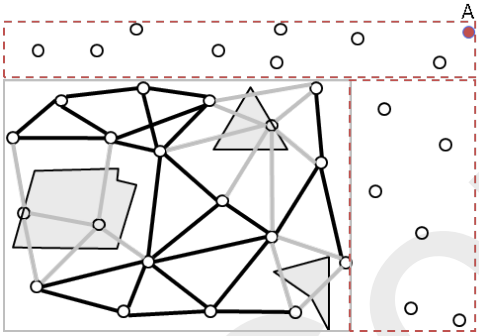
**Figure 4. Top view of a counter example, motivating the efficient roadmap re-sampling prior to the navigation in unknown terrain.**

Thus, an expansion of the current roadmap is performed that preserves the sampling quality of the original roadmap in the volumes of unknown obstacles. It is based on the concept shown in Figure 5. To efficiently sample the new volume added to $\mathcal{C}$, disjoint bounding box volumes are generated. These represent the volumes into which the roadmap has to be extended. This can occur in potentially all three directions. Subfigure 5(b) illustrates the re-sampling process. The deterministic sampling used in the a priori sampling phase is used to sample these volumes as well. Finally all samples are connected in the same way as it

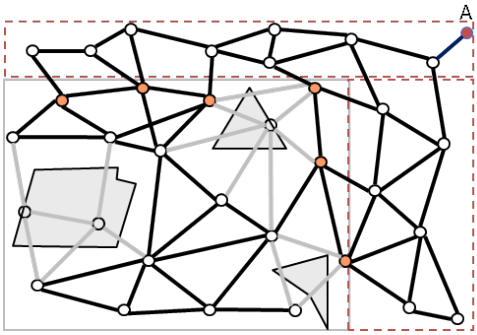
was done for the initial roadmap. Moreover, the edges are spatially indexed as well, in order to account for required real time updates in case of obstacles. Note that within the new sampled volumes, samples are connected without collision checking, since these unexplored volumes are considered as $\mathcal{C}_{free}$. This reduces the computation time to construct each new partial roadmap. Later and if required, the real time edge update procedure will render affected edges as not traversable.



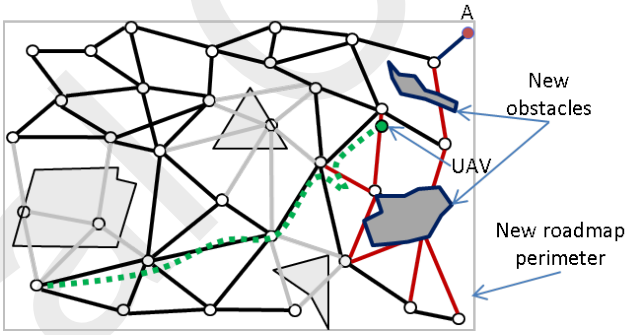

(a) Target vertex outside current roadmap perimeter




(b) Spatial index enlarged and new volume sampled




(c) New samples connected with target vertex and original roadmap




(d) UAV maps new obstacles on its way to "A" and a path in the roadmap still exists

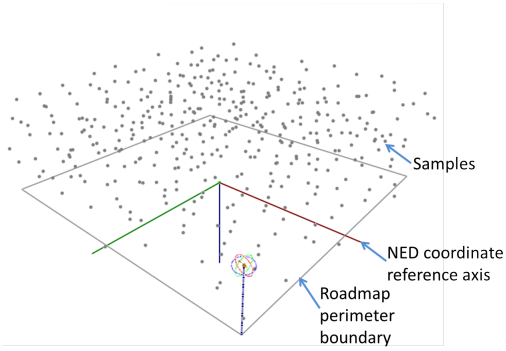
**Figure 5. Top view of the expansion of an a priori generated roadmap towards waypoint "A" that is outside the current roadmap boundary volume: Collision free edges (black), non-traversable edges of the original roadmap (grey) and the expansion graph (red).**

## D. Roadmap-Based Exploration

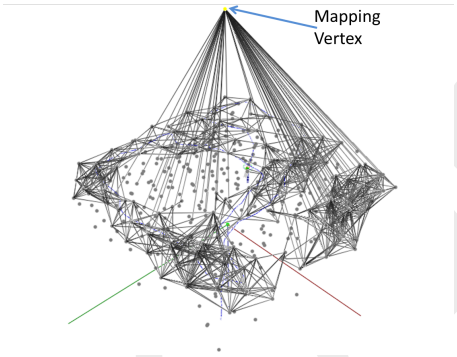
The exploration strategy comprises a *greedy mapping* approach as described by S. Koenig in Ref. 14. This strategy refers to a simple yet powerful incremental mapping and path re-planning concept that comprises the following steps:

1. A virtual *mapping vertex* is added to the free space discretization (here: roadmap graph). That is, a reachable vertex in the search graph that is not supposed to be reached by the vehicle though. This node acts as a global steering vertex and remains purely virtual for this purpose.
2. The cost of all edges to adjacent vertex neighbors is initialized uniformly to one.
3. All vertices are connected to the mapping vertex with initial edge cost one. Note that these edges are not checked for collisions with $\mathcal{C}_{obst}$ since these edges guide the vehicle incrementally from one subpath to another - as long as they exist in the graph.
4. Depending on the variant of greedy mapping, a generic rule disconnects vertices from the mapping vertex once they are marked "mapped". Which conditions trigger this event follows next.

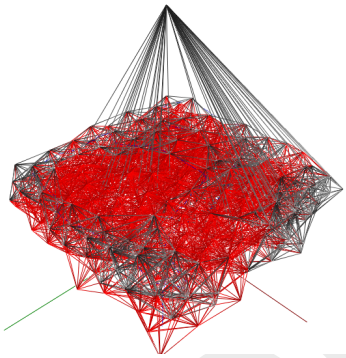
These steps implemented in the roadmap system are shown in Figure 6. Subfigure 6(a) shows the initial sample distribution for a given volume to be explored. The center top vertex shown in Subfigure 6(b) and following, indicates the acquired free space topology after every mapped vertex has been disconnected from the mapping vertex. Subfigure 6(e) indicates the complement set of the free space by non-traversable edges that are kept in the roadmap (e.g. in case a passage was only temporarily not traversable). Finally, the subfigure 6(f) shows the actual polygon obstacle data that changed the graph edges and the mapping vertex accordingly.



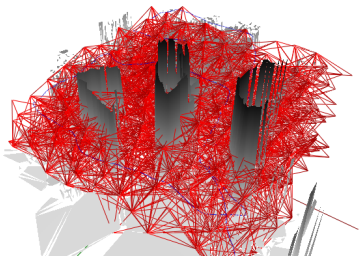

(a) Roadmap samples for the volume to be explored



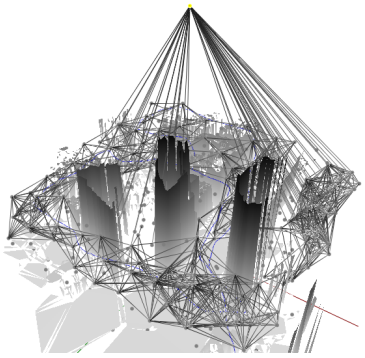

(b) Virtual mapping vertex (top center) connected to all remaining vertices



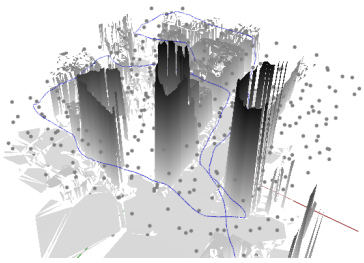
(c) Mapped non-traversable edges (red) after complete exploration



(d) Mapped polygons (greyscale) causing the non-traversable edges (red)



(e) Effective roadmap for free space paths



(f) Effective polygon map (greyscale) and the path flown to acquire this map (blue)

**Figure 6. Roadmap-based exploration approach using a virtual vertex for greedy mapping.**

Using greedy mapping, the path planning problem is reduced to a single target "A to B" problem where the target position remains the virtual mapping vertex. Every time the topology changes (e.g. obstacles render edges non-traversable or vertices are marked) the path is re-planned and guides the vehicle to the

closest neighbor while maintaining a global map coverage using the mapping vertex. The efficient path re-planner used in this approach is D*-Lite.[14]

Three common strategies exist to mark vertices "mapped". The work by Edelkamp[18] highlights graph-based properties that influence the marking state of each vertex:

1. *visited*: This vertex was physically passed or reached by the vehicle with a defined proximity distance.
2. *scanned*: If all edges to and from this vertex have been visited or "seen".
3. *uninformative*: Once this vertex is detected by the mapping sensor (or "seen"), it is considered to not provide any more useful information to the graph topology.

Note that once a vertex is marked it remains in its state (here: uninformative, visited or scanned). Moreover, an alternative variant of greedy mapping exists that always moves to a closest unvisited vertex. But this version is not used in this work.

Given a mapping sensor of at least the range to detect terrain at each of the adjacent neighbor vertices, makes the "visited" property certainly the strategy that yields the longest paths for mapping. The "scanned" property is similar to a graph inversion procedure where vertices become edges and vice versa. This transformation during runtime might become computationally demanding, since for every vertex the neighborhood state needs to be traced until it can be marked "scanned". Uninformative vertices are comparably simple to detect. As soon as a vertex is visible and inside the sensor FOV, it can be marked. The longer the sensor range and the wider the view angles, the more vertices can be marked "mapped" without the need to visit them or wait to visit their neighbors.

The performance of greedy mapping depends on its intrinsic properties. The worst case complexity in terms of the trajectory length is known to be $O(n \log n)$ moves in graphs with $n$ edges. Since greedy mapping is a global planning approach, the worst case trajectory length is generally not reduced by any a priori knowledge of the roadmap graph. However, greedy mapping is known to outperform general uninformed heuristic search that requires $O(n^2)$ moves or even worse. A more practically evaluated greedy mapping in Ref. 19 states that although greedy mapping is super linear in the number of vertices, the worst case trajectory length is at most $O(n^{3/2})$. A conventional search method is depth-first search which moves the vehicle from its current vertex to an adjacent unvisited vertex, as long as it exists. In case none exists, it leaves the current vertex tracks back along the edge from where the vertex was initially reached. Mapping is finished when no unvisited vertices exist anymore. Although depth-first search would only be linear in the number of vertices $n$, there is a practical chance for greedy mapping to perform better than worst-case since the difference is comparably small. If necessary (e.g. for large sparse or dense roadmap topologies), this improved analysis unveiled that it is possible to reduce this difference compared to depth-first search $O(n \ln n)$.

As a result, greedy mapping is a promising technique to exemplify the utility of the presented online roadmap-based path planner for online task planning (here: exploration task). The uninformative strategy is chosen, since the obstacle mapping sensor usually provides a longer detection range than roadmap edge length such that at least two or more samples are within the FOV. For example, conventional laser range finders provide a detection range $d_{sensor}$ of at least 50 m, whereas the expected narrow corridor width of the urban terrain of around 20 meters is the lower bound roadmap sample distance. The expected edge length for nearest neighbors in the graph is of the same length. Thus, the sensor can potentially "see" at least one adjacent neighbor from a given vertex position.

## V. Simulation Results

The following results will provide details of the test runs for an urban terrain section in the city of Berlin, Potsdamer Platz, Germany [a] (Figure 7(a) and Figure 8(a)). Moreover, performance metrics from an unmanned helicopter guidance benchmark framework[20] are used to evaluate the performance (trajectory time), trajectory smoothness (acceleration and turn rate) and motion safety (obstacle clearance).

[a] $http://www.openstreetmap.org/index.html?lat = 52.52564\&lon = 13.40185\&zoom = 11$

### A. Simulation and Planner Setup

While the vehicle senses the obstacles the planner is updated in real time. The vehicle simulation used is based on a software-in-the-loop simulation of the vehicle where the original closed loop flight control software infrastructure is used. The flight control system is based on an adaptive model following approach[21] and ideal vehicle states estimation. A more detailed description on how MiPlEx is integrated with the trajectory following control and the flight control interface can be found in Ref. 22. The vehicle dynamic limits are set to the automatically computed $V_{k_{max}}$ of 4.2 $\frac{m}{s}$ (see section B), a horizontal acceleration limit of 4 $\frac{m}{s^2}$, a vertical acceleration limit of 2 $\frac{m}{s^2}$, and a maximum heading turn rate of 180 $\frac{deg}{s}$. The current implementation suitable for the flight test hardware runs on a conventional x86 hardware with a low memory footprint of around 64 MB and re-plans at 5 Hertz within 0.1 seconds. However, the computer hardware used to generate the results requires computationally demanding obstacle sensor simulation and performance instrumentations that otherwise would not be performed onboard the rotorcraft avionic. Thus, an Intel Core i7 at 2.8 GHz with 4 GB RAM and Windows 7 64 bit is used. The simulation is run with time synchronized planning, control and sensing steps without multi-threading.

The virtual obstacle sensor is a LIDAR sensor model based on the specification of commonly used small and light weight laser range devices[b]. The opening angle of the laser range finder's 2-D scan plane is set to comparably conservative 180°, in case the rotorcraft fuselage limits the horizontal FOV of the original 270°. The sensor range $d_{sensor}$ is set to 50 m. Due to the moderate forward flight velocities the generation of one full 2-D scan is considered quasi instantaneous. To reduce the computation efforts for real-time simulation, a reduced set of scan points represents the scan plane. The distance between each beam at the maximum sensor range is the minimum obstacle clearance $d_{clear}$ (10 m) as it is used for path planning. This results in 32 scan points in the scan plane. Based on the concept of a quasi 3-D LIDAR,[20] the sensor is attached at the front tip of the vehicle. The 2-D scan plane is oriented symmetrically around the vehicle longitudinal axis. Moreover, it is continuously rotated around this longitudinal axis with 1 Hz. This way, a quasi hemispherical FOV is implemented.

Further, the range detection of each laser beam measurement is mimicked using virtual beam lines. The sensor simulation performs collision checking for each virtual laser beam and the current position of the vehicle in the a priori unknown terrain model data. This a priori unknown urban terrain is represented by a polygon set which is the result of a high precision terrain object model combined with a loss-less data reduction post processing.[23] Hence, this obstacle sensor implements an obstacle mapping approach that provides the polygonal input for the planning system.

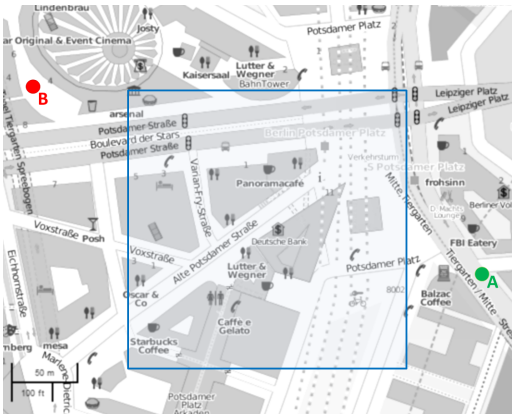
Finally, online sampling-based roadmap planning implies a feasible number of sample connections within the neighborhood of the current vehicle state. Otherwise the planner cannot connect the vehicle vertex to the roadmap and will fail to find a path. Thus we define a roadmap visibility metric $\epsilon_{local}$, which is the percentage out of all collision free edge connections $e_{free}$ to neighbor samples over all neighbor edge connections $e_{neighbor}$ within a Euclidean range $r_{neighbor}$ in the roadmap. The planner resolution is set to 20 m which is the minimal expected corridor width the vehicle is allowed to traverse in the given urban terrain segment.

### B. Navigation in Unknown Terrain Using Roadmap Expansion

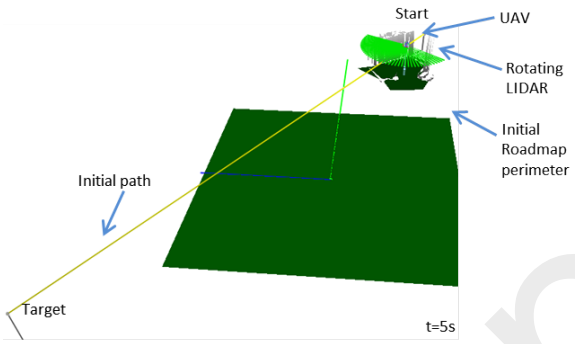
Before the greedy mapping performance is assessed we run trials to see whether the roadmap expansion for tasks exceeding the current roadmap perimeter shows the desired effect. The vehicle is positioned outside an initially empty world at 60 m above the ground level.

The start position "A" and the target position "B" are set outside the initial roadmap perimeter (Figure 7(a)). The target "B" is set into a narrow passage that is a priori unknown to the planner. The initial roadmap is constructed within the given perimeter and has only traversable edges since no polygon obstacles are initially known to the system. As soon as the planner notices that the vehicle start position as well as the target position exceed the roadmap boundaries, the roadmap is expanded according to the procedure discussed in section 5. Note that without the expansion a similar situation as shown in Figure 4 would have occurred: The visibility from the start and the target position to the initial roadmap would have become zero as soon as obstacles outside the current perimeter intersect the line of sight from the vehicle to the roadmap.

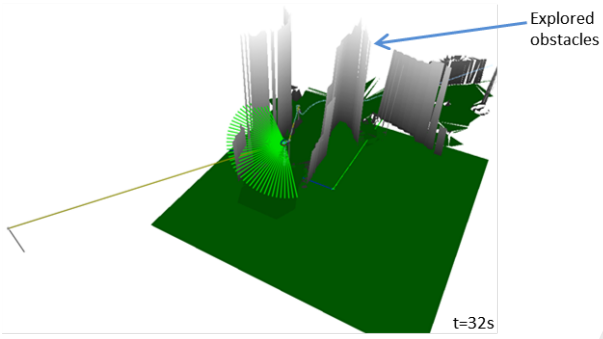
[b] e.g. HOKUYO UTM-30LX, $http://en.manu-systems.com/UTM-30LX.shtml$



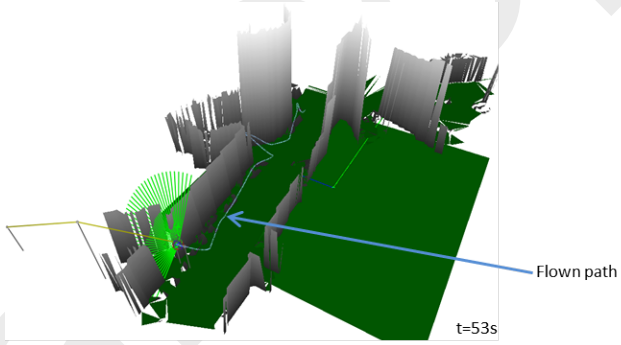

(a) Urban test scenario (Map from: www.osm.org): Start (green) and target (red) exceed the initial roadmap perimeter (blue)




(b) Initial empty world (view from West)




(c) At one half of the way towards the target position




(d) Terminal path through a narrow corridor

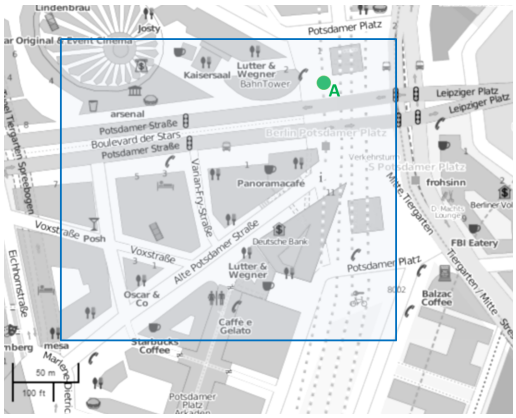
**Figure 7. Exploration scenario for a single waypoint target (a): Obstacles detected after 5 seconds (b), path deviation due to obstacles after 32 seconds (c), and obstacles detected in the course of the flight that exceeded the initial roadmap (d).**

Figure 7 shows the planner performance towards a single target in a worst case scenario. Initially no obstacle information is known for a workspace volume of around 800 m side lengths and 100 m height. Without the roadmap expansion, the re-planning would have failed, since no collision free connection to the initial roadmap is possible. The planner was able to enter the a priori roadmap volume from its start point, and it would have been unable to find a path towards the target position as soon as it left the roadmap perimeter. Once it entered the narrow corridor section it still could find a path (Figure 7(d)) which would not have been possible without an additional sampling step performed by the roadmap expansion.

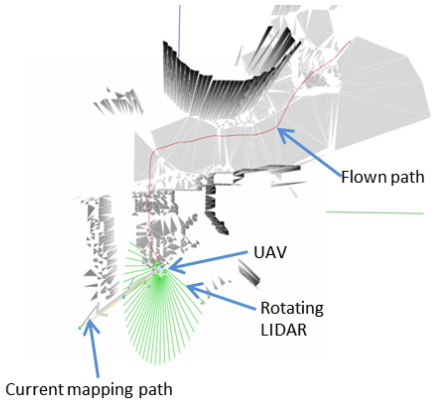
### C. Greedy Mapping of an Unknown Terrain Volume

The mapping procedure is tested in the "unknown" urban terrain from the previous simulation. The volume to be explored is set to 200 $m^2$ ground plane area and 75 m above the ground level. The initial empty roadmap that approximates the given exploration volume comprises 397 vertices and 12630 collision free edges. The start position "A" is set to 50 m above the ground level. Within the given volume height, the vehicle is able to plan paths above the roofs of most of the buildings except for three towers that are taller than 75 m (Figure 8(a), at the main intersection road).

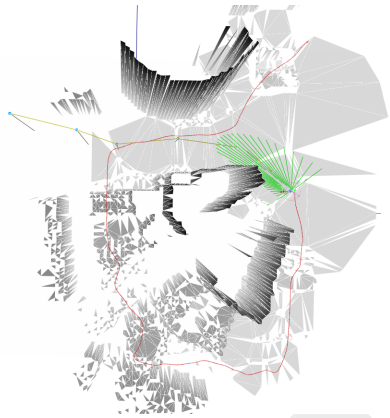
The iterative mapping result of the closed loop simulation with the previously described virtual obstacle mapping sensor and mapping model is shown in Figure 8. The vehicle continues to fly from the altitude where it started (Figure 8(b) and Figure 8(c)). This is behavior is primarily guided by the mapping vertex and supported by the uniform edge cost of all remaining vertices to the mapping vertex. No explicit preference for a certain height above ground is implemented. After that the vehicle closes a loop on that altitude until all roofs are mapped, and then it descents to a lower altitude while flying back to the start area (Figure 8(c)).



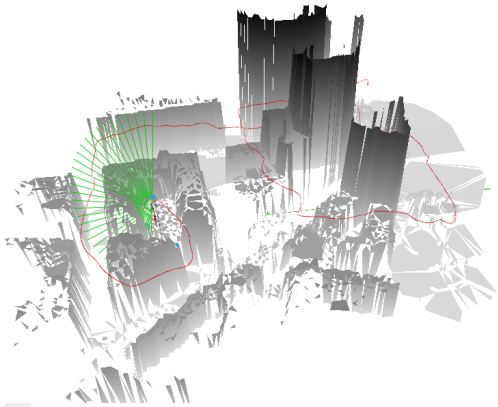

(a) Urban test scenario (Map from: www.osm.org) with start position (green) and the roadmap perimeter (blue) of the exploration volume




(b) map after 31 sec



(c) map after 131 sec



(d) 358 sec, final 3-D map (south perspective)

**Figure 8. Top view for 3D mapping of a priori unknown terrain using original terrain data: Effective trajectory (red line), polygon obstacles (greyscale, with black as highest elevation), and partial path planned using greedy mapping (green).**

Most vertices present at the ground level have been disconnected from the mapping vertex, such that an additional edge transition towards a lower or higher altitude is considered the cheapest path to the mapping vertex. The vehicle continues to move through the same corridor ("Potsdamer Str.") westwards and climbs to a higher distance from the ground (Figure 8(d)).

Finally, by mapping the last "open" roof surface there are no traversable edges left that allow the vehicle to plan a path using the mapping vertex. Note that this includes collision free edges that are located below the previously unknown terrain surface where the vehicle cannot go anymore. Hence, a mapping ratio of 100 % (here: all vertices become uninformative) is practically not possible as soon as a closed terrain surface exists within the given exploration volume.

The given 3-D exploration volume is mapped within a total trajectory time of 358 s and a coverage of over 78 % of the available polygonal area is achieved (Table 1). Given a maximum flight time of the ARTIS flying testbed of at least 10 minutes, the mapping would be possible without an interruption of the task and probably with a return to the start position. Note that even a longer trajectory time (e.g. due to a shorter sensor range $d_{sensor}$ that causes less vertices to be marked "uninformative" at a time), the greedy mapping approach would be beneficial. Its reactive planning nature allows to pause the mapping process at one position and to continue from another position. Hence, this would allow to perform longer flight missions automatically, where a refueling (or recharge) of the vehicle is required in order to achieve the required mission time.

One of the main motivations for an online multi-query path planning is the ability to save path planning

| Criterion | Metric | Result |
|---|---|---|
| Exploration | mission time | 358 s |
| | trajectory length | 1245 m |
| | polygon area mapped | 78.3 % |
| | vertices marked | 85.5 % |
| CPU Duty Time | mapping + roadmap update + path planning | 256 s (72 %) |
| | path planning | 14 s (4 %) |
| Trajectory Smoothness | mean velocity | 2.9 $\frac{m}{s}$ |
| | 95th percentile velocity | 4.1 $\frac{m}{s}$ |
| | mean acceleration | 0.88 $\frac{m}{s^2}$ |
| | 95th percentile acceleration | 1.85 $\frac{m}{s^2}$ |
| Trajectory Safety | mean terrain clearance | 17.5 m |
| | 5th percentile terrain clearance | 10.9 m |
| | mean $\epsilon_{local}$ of vehicle vertex in roadmap | 40.2 % |

**Table 1. Exploration performance indications for the roadmap-based mapping trajectory test.**

time such that planning performance is available for other decision making tasks onboard the vehicle. The CPU duty times in Table 1 indicate that this strategy shows the desired effect: The path planning consumes just $\frac{1}{20}$ of the total computation time spent for greedy mapping-based exploration and updating the roadmap. Although with every new obstacle mapped the greedy mapping procedure sometimes plans paths with significantly different subsequent segments to fly, the flight velocity is close to the maximum FOV velocity $V_{k_{max}}$ in 95 % of the trajectory time.

The cumulative distribution function (CDF) in Figure 9 shows a statistical evaluation of the mapping trajectory from a guidance point of view, i.e. a minimum trajectory time (performance), low accelerations and turn rates (smoothness), and no underrun of the desired obstacle clearance (safety). The green bars indicate the upper limit or lower limit respectively. The vehicle stays within the boundary limits (green) all the time. The probability of an acceleration less than 2 m/s$^2$ is close to 100 % due to the direction changes caused by greedy mapping. This indicates that just 50 % of the allowed maximum acceleration is used. The vehicle does not fall under the required minimum obstacle clearance $d_{clear}$ of 10 m. A mean $d_{clear}$ of 17 m indicates that the roadmap sample distances could even be decreased. However, this would result in a longer mapping time, since more vertices (and thus a more detailed "visit" of the free space) need to be marked by the greedy mapping procedure. Hence, if a map with a higher level of detail and low number of samples in the roadmap are required, the vertex marking strategy should be adapted rather than increasing the number of roadmap samples.

## VI. Conclusion

This paper presented two extensions to an online multi-query path planning approach that enable highly automated 3-D exploration tasks:

1. The first extension expands the roadmap in such a way that the original path planning completeness is assured for the volume of unknown terrain. This way the vehicle can replan paths more safely when leaving the initial roadmap perimeter on its way to a waypoint farther away.

2. The second enhancement enables online task re-planning using the multi-query property of the path planner. It is exemplified using a greedy mapping approach for predefined volumes of unknown terrain.

The used path planner is integrated into the MiPlEx planning and execution framework, such that a small unmanned helicopter is guided automatically through a dense obstacle field.

The simulation results show that the used online roadmap-based path planner can be extended to perform online higher level guidance decisions. The results are based on real urban obstacle terrain models, from which unforeseen obstacles are retrieved during the obstacle sensor emulations.

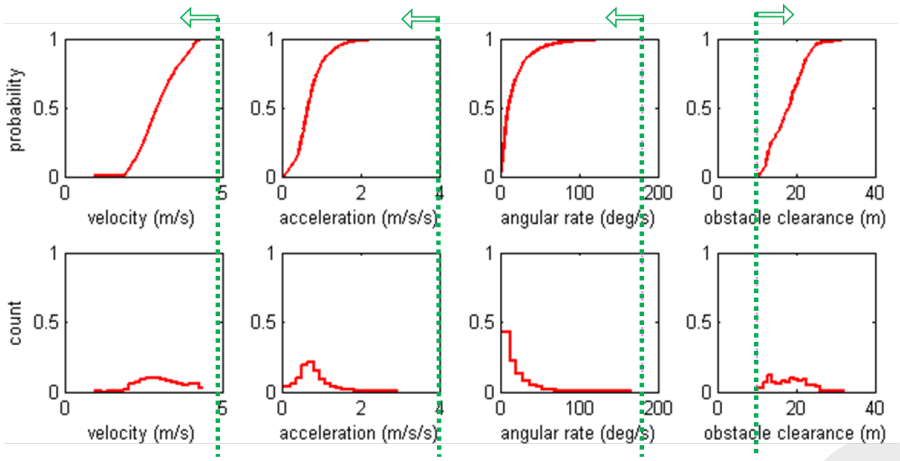



**Figure 9. CDF and histogram of measurement magnitudes for the performed exploration trajectory: Green arrows indicate the desired limitations the vehicle must not violate.**

Currently, the mapping performance could be improved into various directions. First, the comparably high accelerations may be reduced by an omnidirectional sensor FOV, as it is assumed to exist in the original 2-D indoor robot examples for greedy mapping. Second, marking the roadmap vertices as "mapped" is based on the estimated mapping sensor performance. Receiving a direct input from the real sensor could improve maps, e.g. by including the sensor directly into the marking strategy of the roadmap vertices. Furthermore the edge cost function used for greedy mapping could comprise a bias to account for hazardous certain locations (e.g. GPS-denied corridors) or regions of interest (e.g. landing sites). The roadmap costs are currently uniformly distributed and set equal to one, so that they can be exchanged by probability values. A nice property of a roadmap planner is that it can be utilized as a central or multi-master coordination planner for multiple vehicles.[24] It would be interesting to apply the roadmap-based greedy mapping approach to such a coordination problem.

## References

[1]Croes, A., "A method for solving traveling salesman problems," *Operations Research*, Vol. 5, 1958, pp. 791—812.

[2]Pettersson, P. O. and Doherty, P., "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *Journal of Intelligent and Fuzzy Systems*, Vol. 17, No. 4, 2006, pp. 395–405.

[3]Miller, J. R., *A 3D Color Terrain Modeling System for Small Autonomous Helicopters*, Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 2002.

[4]Krause, S., "Multi-Purpose Environment Awareness Approach for Single Line Laser Scanner in a Small Rotorcraft UA," *Journal of Intelligent and Robotic Systems*, Vol. 65, 2012.

[5]Schmid, K., Hirschmüller, H., Döel, A., Grixa, I., Suppa, M., and Hirzinger, G., "View Planning for Multi-View Stereo 3D Reconstruction Using an Autonomous Multicopter," *Journal of Intelligent and Robotic Systems*, Vol. 65, 2012, pp. 309–323, 10.1007/s10846-011-9576-2.

[6]Wu, P. P.-Y., Campbell, D. A., and Merz, T., "Multi-Objective Four-Dimensional Vehicle Motion Planning in Large Dynamic Environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 41, No. 3, 2011, pp. 621–634.

[7]Merz, T. and Kendoul, F., "Beyond visual range obstacle avoidance and infrastructure inspection by an autonomous helicopter," *IROS*, 2011, pp. 4953–4960.

[8]Mettler, B. and Toupet, O., "Receding Horizon Trajectory Planning with an Environment-Based Cost-to-Go Function," *IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

[9]Herman, M., "Fast Three-Dimensional Collision-Free Motion Planning," *IEEE Conference on Robotics and Automation*, 1986.

[10]Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *J. Intell. Robotics Syst.*, Vol. 57, No. 1-4, 2010, pp. 65–100.

[11]Adolf, F.-M. and Andert, F., "Rapid Multi-Query Path Planning For A Vertical Take-Off and Landing Unmanned Aerial Vehicle," *AIAA Journal of Aerospace Computing, Information, and Communication*, Vol. 8, No. 11, November 2011, pp. 310–327.

[12]Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers*, Vol. C-32, No. 2, Feb. 1983, pp. 108–120.

[13]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006, Also available at http://planning.cs.uiuc.edu/.

[14]Koenig, S. and Likhachev, M., "Improved fast replanning for robot navigation in unknown terrain," *IEEE International Conference on Robotics and Automation*, 2002, pp. 968–975.

[15]Andert, F. and Adolf, F., "Online world modeling and path planning for an unmanned helicopter," *Autonomous Robots*, Vol. 27, No. 3, 2009, pp. 147–164.

[16]Ferguson, D., Likhachev, M., and Stentz, A., "A Guide to Heuristic-based Path Planning," *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[17]Friedman, J. H., Bentley, J. L., and Finkel, R. A., "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematics Software*, Vol. 3, No. 3, September 1977, pp. 209–226.

[18]Edelkamp, S. and Schroedl, S., *Heuristic Search - Theory and Applications.*, Academic Press, 2011.

[19]Koenig, S., Tovey, C., and Halliburton, W., "Greedy Mapping of Terrain," *In Proceedings of the International Conference on Robotics and Automation*, IEEE, 2001, pp. 3594–3599.

[20]Goerzen, C., "Benchmarking of Obstacle Field Navigation Algorithms for Autonomous Helicopters," *American Helicopter Society 66th Annual Forum*, Phoenix, AZ, May 2010.

[21]Lorenz, S., *Adaptive Regelung zur Flugbereichserweiterung des Technologiedemonstrators ARTIS*, Ph.D. thesis, Technische Universität Braunschweig, Shaker Verlag, Aachen, 2010.

[22]Lorenz, S. and Adolf, F. M., "A Decoupled Approach for Trajectory Generation for an Unmanned Rotorcraft," *Advances in Aerospace Guidance, Navigation and Control*, edited by F. Holzapfel and S. Theil, Springer Berlin Heidelberg, 2011, pp. 3–14, 10.1007/978-3-642-19817-5_1.

[23]Adolf, F.-M. and Hirschmüller, H., "Meshing and Simplification of High Resolution Urban Surface Data for UAV Path Planning," *J. Intell. Robotics Syst.*, Vol. 61, No. 1-4, Jan. 2011, pp. 169–180.

[24]Adolf, F., Langer, A., de Melo Pontes e Silva, L., and Thielecke, F., "Probabilistic Roadmaps and Ant Colony Optimization for UAV Mission Planning," *6th IFAC Symposium on Intelligent Autonomous Vehicles*, 2007.