Semantic Trajectory Planning for Long-Distant Unmanned Aerial Vehicle Navigation in Urban Environments

Markus Ryll[1], John Ware[1], John Carter[1] and Nick Roy[1]

***Abstract*— There has been a considerable amount of recent work on high-speed micro-aerial vehicle flight in unknown and unstructured environments. Generally these approaches either use active sensing or fly slowly enough to ensure a safe braking distance with the relatively short sensing range of passive sensors. The former generally requires carrying large and heavy LIDARs and the latter only allows flight far away from the dynamic limits of the vehicle. One of the significant challenges for high-speed flight is the computational demand of trajectory planning at sufficiently high rates and length scales required in outdoor environments. We tackle both problems in this work by leveraging semantic information derived from an RGB camera on-board the vehicle. We first describe how to use semantic information to increase the effective range of perception on certain environment classes. Second, we present a sparse representation of the environment that is sufficiently lightweight for long distance path planning. We show how our approach outperforms more traditional metric planners which seek the shortest path, demonstrate the semantic planner's capabilities in a set of simulated and excessive real-world autonomous quadrotor flights in an urban environment.**

I. INTRODUCTION

Autonomous navigation of unmanned micro-aerial vehicles (MAVs) in an urban environment is a challenging problem and an active field of research [1]–[4]. An aerial vehicle, operating at a low altitude, in an unknown and unstructured environment has to simultaneously solve several problems including perception, self-localization, trajectory planning and control in order to ensure safe motion. Despite the increased likelihood of collisions resulting from operating at a low altitude is a requirement for many tasks (e.g. search and rescue under tree canopy). While most aerial applications benefit from high-speed navigation ($\gg 10\,\mathrm{m\,s^{-1}}$) by extending their coverage of the environment (e.g., monitoring and surveillance), high-speed navigation is essential to others (e.g., urgent medical goods transportation [5]).

Although there are many reasons to want autonomous, high-speed navigation on MAVs, it presents many inherent challenges. First, the size, weight, and power constraints (SWaP) presented by MAV platforms results in myopic perception and demands relatively slow flight in order to ensure safety. The maximum depth perception of stereo imaging devices mainly depends on the baseline, focal length and image resolution. Standard stereo cameras that fit on MAVs have a limited perception range of approximately $10\,\mathrm{m}$ [6], [7]. Stereo cameras with a larger depth perception are heavier and suffer from a higher minimum depth perception range, creating difficulties in sensing to close-in obstacles. Structured light sensors, such as the Asus Xtion Pro, have similar perception ranges [8], but do not work in sunlight. LIDARs have significantly longer sensing ranges, but also much larger SWaP requirements. Second, high-speed flight requires the ability to plan over much longer trajectories in order to actually use long-range perception. The computational complexity of trajectory planning typically grows exponentially with the length of the trajectory, creating challenges for real-time applications on computationally limited hardware. Third, high-velocity flight demands that the mapping process remains fast and efficient as additional parts of the environment are revealed. Common approaches for high-speed navigation in urban environments either use special long range sensors [2] or fly at the safe limit of the perceptual range. To give an example, a quadrotor with a weight to thrust ratio of 1.5, flying at a forward velocity of $10\,\mathrm{m\,s^{-1}}$ demands approx. $9\,\mathrm{m}$ to come to a stop. Both of these solutions result in the vehicle being unable to exploit the inherently fast dynamics of an MAV [4] and provide performance that remains far behind the skills of a trained human pilot. In this work, we show how semantic information in the perception-estimation-planning loop enables longer perception horizons and therefore enables faster flight. Furthermore, we demonstrate how semantic information can

Fig. 1. Quadrotor performing high-speed, autonomous navigation mission in an urban environment as well as the graph representation and the road center line, learned during a mission on-board of the vehicle and in real-time. Start- 'S' and end-location 'E' are highlighted.

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, USA
{ryll,jakeware,jcarter,nickroy}@csail.mit.edu
This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-17-2-0181. Their support is gratefully acknowledged.

be used to build a sparse representation of the environment. In particular, we will focus on mapping the road network within an urban environment. We selected this environment because urban spaces tend to be built of two distinct types, roads and structures. In this setting, an agent can gain significant advantage by remaining on the sparsely occupied roads instead of taking the often shorter route through the interior of an obstacle rich city block. A simplifying property that will be exploited in this work is that roads are often flat and we will provide a sensitivity analysis to show the robustness of our approach to violations of this assumption. Finally, note that we are only using roads as an example for the more general principle of leveraging semantic information for trajectory planning. Similarly, we could use the classes of grass or terrain in a forest environment.

Contributions: In this work, we seek to demonstrate high-speed navigation (e.g., from a starting point to one or multiple goal locations) in a large, initially unknown, urban environment. This is difficult due to the aforementioned challenges presented by the perception, planning, and mapping problems. To address these challenges, our approach considers semantic information found in the color camera image and provides two primary contributions.

- We use semantic information drawn from a segmented color camera image to increase perception range and allow for long-distance mapping and planning. Our experiments will show this assumption to hold well in urban environments.
- We present an approach based on inference by a deep neural network to generate traversability networks represented by a combination of lines, arcs, and clothoids (as well known as Euler spirals) which outperforms other state-of-the-art methods and allows for highly efficient long-distance trajectory mapping and planning.

In Sec. II, we will discuss how our work is motivated by previous work on the topic. Then, in Sec. III present the process for generating the graph representation of the environment. In Sec. IV we will briefly present the software framework and in Sec. V we discuss thoroughly simulated and initial real-world results. In the last section we will discuss current limitations and future work.

II. RELATED WORK

There exists a broad spectrum of techniques for MAV path planning in unknown, outdoor environments. Most of these techniques can be placed in one of two groups defined by a combination of target environment and sensors suite. The first group is GPS-enabled, high-altitude flight in clutter-free environments [9], [10] while the second group is GPS-denied, low-altitude flight in cluttered environments [3], [4]. Because urban environments introduce errors in GPS measurements due to multipath and occlusion, the latter group and the work presented here relies on cameras for state estimation and perception.

In this setting, the most common approach to mapping is on-board visual SLAM where the agent geometrically reconstructs the environment and simultaneously localizes itself within it [11]. Once a metric map has been built, it can be used for trajectory planning. Although common, planning for high-speed flight in these dense metric maps can be challenging and it is often not clear how best to select control inputs to satisfy the myriad constraints present in this problem. Although not alone, the work done in [4] discusses an approach to navigating cluttered, unknown environments by reasoning about obstacle proximity in a multi-tiered planning system to provide a flexible velocity planning strategy. Other recent efforts to overcome this challenge have focused on learning direct control policies from raw sensor measurements to motion primitives or steering angles [12], [13]. These methods (e.g., [13] used car dashcam video data as they are widely available) require a large amount of training data, which is costly and often difficult to acquire, or use simulated data, which often does not generalize well.

In our work we seek to find a different approach by learning a representation of the environment that simplifies the trajectory planning problem with the help of semantic information. Although some SLAM approaches such as [14], [15] jointly consider geometric and semantic information (acquired by an image segmentation deep neural network) to improve the localization, our approach treats these as independent components. We first obtain a state estimate to use during local, geometric map fusion and then use a segmentation image generated from the color camera to infer the semantic class of the environment beyond our depth perception range while also building a lightweight map representation to enable long-range, efficient planning.

III. METHODOLOGY

In this work, we seek to control an MAV at low altitude, in a cluttered, urban environment, equipped with an RGB-D camera and IMU, to a goal location in an unknown, urban environment and focus on two primary technical challenges. First, given the number of possible vehicle states and map configurations, solving this trajectory finding problem beyond toy examples is hardly viable on large scale maps. Second, the perceptual range limitation coupled with a desire for safety limits the vehicle's top speed. We solve the former problem by learning and tracking a graph representation of the environment that simplifies the trajectory planning even for long range trajectory planning. The latter problem is addressed by expanding the effective perception range with the use of a semantic segmentation image from the RGB camera. To enable both of these capabilities, the computational complexity of the solution must be lightweight enough to be computed in real-time on-board, with a sufficient rate, on a resource constrained MAV. The high-level steps of the approach are as follows:

First, we use a semantic segmentation network[1] to obtain a segmented image from the current camera image and the state estimation pipeline described in [16] to obtain a state estimate. Second, we fuse the latest semantic image

[1] https://docs.openvinotoolkit.org/latest/_intel_models_semantic_segmentation_adas_0001_description_semantic_segmentation_adas_0001.html

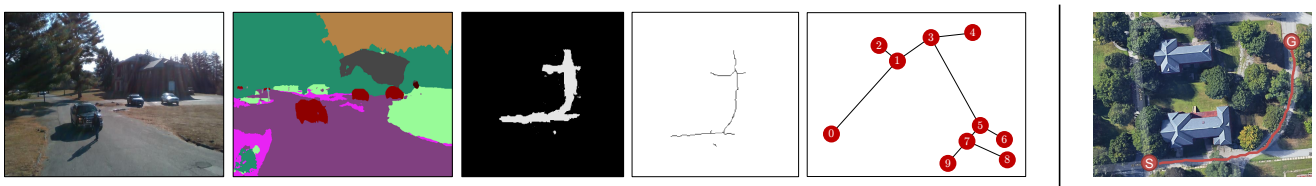

Fig. 2. Overview of the graph network learning pipeline: First, the camera image is segmented into pixelwise semantic labels. Based on the semantic segmentation, we generate a probabilistic top-view projection of the current camera image. The top-view projection is integrated into a probabilistic map of the local environment. The center line is computed using Zhang's image skeletonization method. Finally, the graph-based representation of the center line is inferred, which is used for trajectory planning. On the right side is a top-view image of the environment with the starting and the goal location highlighted.

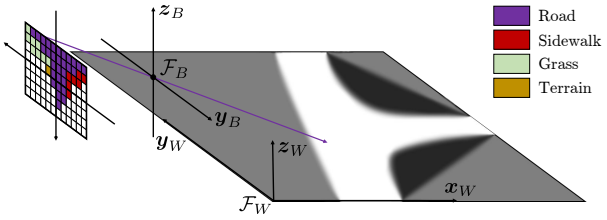

Fig. 3. Schematic visualization of the ray-casting, probabilistic mapping (brighter color symbolizes a higher probability to be road), and semantic labels. When fusing the semantic projections into a semantic map, we give road labeled pixels positive weights and other labels (Sidewalk, Grass, and Terrain) negative weights.

into a Bayesian top-view representation of the environment. In this work, we infer a representation of the surrounding roads. Third, we transform this top-view representation into a lightweight graph representation. Fourth, we generate a trajectory based on the graph representation. An overview of these steps is depicted in Fig. 2. Each of these steps is described in detail below.

A. Learning Graph Representation

In this work, we make a flatness assumption that allows us to place certain environment classes (e.g., roads, sidewalks, grass, terrain) on a single x-y-plane in the world frame $\mathcal{F}_W$ (see Fig. 3)[2].

When developing our graph network representation for fast MAV flight, we sought three key properties:

- Sparsity: A good representation of the underlying road network strongly sparsifies perceived data (e.g. camera information). Note that this precludes solutions that rely on dense metric discretization such as voxel-grids.
- Efficiency: A good representation allows fast, efficient path planning over long distances.
- Control: A good representation allows control inputs to be derived directly.

Based on these requirements, we represent the learned road network as a graph derived from the road's center-line. In this graph, each node symbolizes a road intersection and every edge represents a traversable path connecting those

[2]Even in case that the road plane is not perfectly flat this approach can be beneficial as we do not want to track a certain global altitude but an altitude with respect to the ground plane.

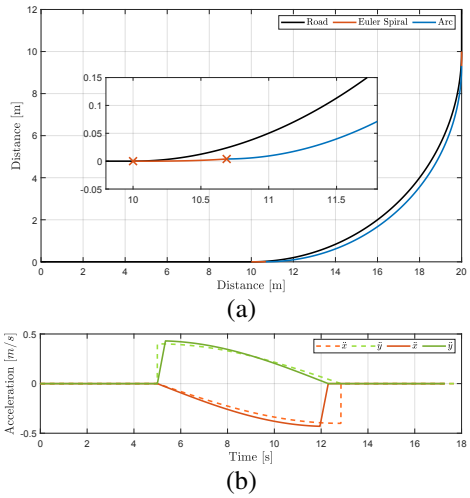

Fig. 4. a) The black line shows the desired road center-line profile for a line-arc-line combination. The red and blue lines show the center-line profile with clothoids between the segments. b) The x and y acceleration profiles generated by tracking the line-arc profile (dashed) and the line-clothoid-arc profile (solid). Note that the desired acceleration is not continuous in case of pure line-arc combinations, while it is continuous for the line-clothoid-arc combination.

intersections with an associated transport cost. In previous literature, road center-lines have mainly been modeled by concatenated line segments [17], [18]. Recently, many other segment types have been used to model road center-lines. For a full survey we refer the reader to [19]. A common approach is to combine line- and arc-segments, e.g. [20]. While combining these two elements seems sufficient for recovering the center-line, a line-arc trajectory suffers from an infinite lateral acceleration at the connection points and fails to meet our smoothness objective. We therefore represent the road center-line as a combination of lines and arcs that are connected with clothoids. This allows for smooth transitions. Clothoids are a natural choice as they are well known for their smoothness properties and are often used in the design of roadways [21]. We follow the work presented in [22] to compute the clothoids. A typical example of a line-clothoid-curve segment is depicted in Fig. 4-(a). Fig. 4-(b) compares the lateral accelerations with and without using clothoids.

1) Top-view projection and probabilistic map: Given the vehicle's estimated attitude $\boldsymbol{R}$, its uncertainty, and the seman-

tic image, we can generate a probabilistic top-view projection of the visible road. By assuming flatness, we use a beam model to estimate the probability distribution of a road pixel in the 2D top-view projection. By projecting and fusing other segmentation classes (sidewalk, grass and terrain) into the top-view image with a negative weight, we are able to significantly increase the quality of the top-view map. Using the drone's position and a binary Bayes filter [23] we fuse the instantaneous top-view into a sliding-window semantic map with the drone at its center.

2) Graph network representation: To generate the sparse segment representation, described at the beginning of this section, we first threshold and then skeletonize the probabilistic map using an implementation of Zhang's method [24]. This gives the center-line of the current probabilistic map. In a next step, we decompose this center-line map into the desired graph structure where every intersection represents a node, and the segments of the graph contain the road center-line. To accomplish this, we use the scikit image processing toolbox.

In the next step, we decompose the pixel-based center-line representation into the sparse line-clothoid-arc representation. Decomposing curves into sub-elements is an active field of research. To decompose curves, it is customary to first detect dominant points [25] and then fit the desired segment types to those points [26]. The performance of these algorithms degrades with increasing noise along the curve and is too slow for a real-time application since finding the dominant points is computationally expensive. To address these short-comings, we use a deep neural network (DNN) to classify the center-line into dominant points, curves, and line segments. The deep neural network is based on a lightweight and shallow image segmentation network and its input is a 64x64 pixel image (see Fig. 5). The DNN was trained on a set of >250.000 artificially generated curves with added noise. The segmentation of a single image takes $4.1\,\mathrm{ms}$ ($s = 3.0\,\mathrm{ms}$) on an Intel i7-8650U CPU. After training, the DNN achieves a per-image pixel classification accuracy of $99.85\,\%$. Interestingly the DNN's true-positive classification of dominant point achieves an accuracy of 73.13. Additionally, we compared true-positive classification of dominant points in a one pixel radius around the true dominant point. Here the DNN achieves an accuracy of $94.75\,\%$, and an accuracy of $97.81\,\%$ in a two pixel radius. A typical example of the segmentation is shown in Fig. 5. To compare the accuracy of our approach, we tested the dominant point detection with the approach presented in [26] where, with a two pixel radius, an accuracy of only $38.02\,\%$ could be achieved. A detailed comparison is shown in Tab. I.

IV. SOFTWARE FRAMEWORK

The presented mapping and planning framework is integrated into an existing flight stack presented in detail in [4]. An overview of the flight stack including the semantic trajectory planning is presented in Fig. 6. We will now summarize the major components of the original flight stack [4] (we will use the original flight stack as a baseline in

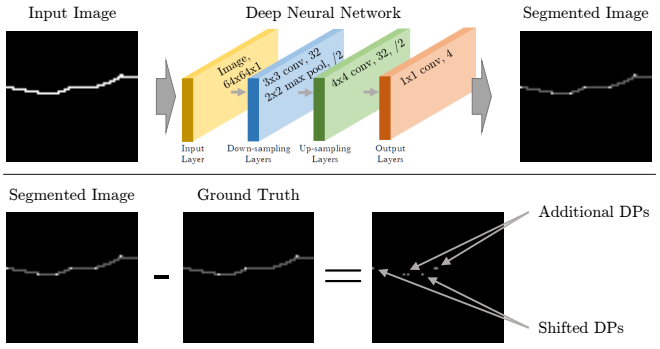

Fig. 5. We use a Convolutional Neural Network that predicts road segments consisting of arcs, lines and dominant points from single 64 by 64 pixel black and white frames. In the notation above, we indicate for each convolution first the kernel's size, then the number of filters, and eventually the stride if it is different from 1. Top: Exemplary input image and segmented output image. Detected dominant points (DPs) are highlighted. Bottom: A comparison of the segmented image and ground truth. Note that two additional DPs are found and two DPs are slightly shifted.

TABLE I

TRUE POSITIVE CLASSIFICATION ACCURACY OF DOMINANT POINTS COMPARING OUR APPROACH AND [26] AT DIFFERENT RADIUS TOLERANCES.

	0 pixel	1 pixel	2 pixel	5 pixel
Our Approach	73.13 %	94.75 %	97.81 %	99.41 %
Nguyen et al. [26]	17.89 %	18.49 %	37.82 %	66.94 %

Sec. V-A) and explain how the new semantic trajectory planning is integrated. The data from the depth sensor are sorted into a k-d tree for efficient data query and at the same time integrated into an occupancy and a distance grid. We unify the grids into a cost grid and use Dijkstra's Shortest Path First (SPF) algorithm to find a traversable global path through the cost grid. For computational reasons the grids are implemented as rolling maps with the quadrotor in the center and with a size of $60 \times 60 \times 6\,\mathrm{m}$ and a resolution of $0.5\,\mathrm{m}$. If the final goal is located outside of the cost grid, the global path ends on the grid cell, nearest to the global goal. Additionally, we define a local goal that moves on the found global path in front of the MAV. The travel speed of the local goal depends on the clutter of the local environment and proximity to obstacles. We track the local goal following a pure pursuit strategy. Finally, with every new depth sensor measurement, we compute more than 300 minimum jerk motion primitives, dependent of the current state of the MAV (different in velocity, final position and height). We assign a cost to the primitives depending on collision with obstacles, proximity of primitive's final position to the local goal and proximity to obstacles and track the motion primitive with the smallest cost. Tracking a local goal and more importantly, being able to locally diverge from the local goal allows to avoid obstacles that have not been detected by the global planner, which runs at a lower updated frequency (e.g., dynamic obstacles). While the motion primitive generation is triggered by every new depth sensor measurement ($30\,\mathrm{Hz}$) the computation of the global path runs only at $2\,\mathrm{Hz}$. The

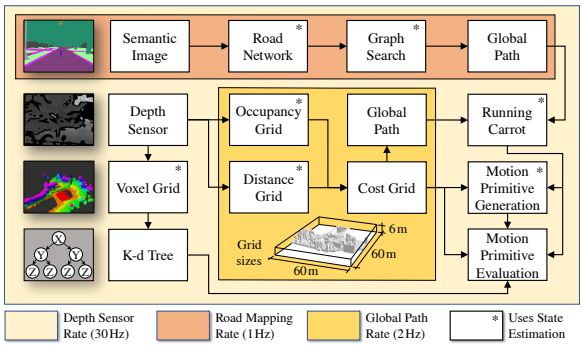

Fig. 6. An overview of the planning stack framework. The bright yellow components consume the depth image and are updated at 30 Hz. The dark yellow components are part of the global path planning and run at 2 Hz.

limited perception range of the depth sensor reduces the maximum safe velocity of the platform.

Given this baseline motion planner we extend it with the semantic trajectory planner by offering an alternative global path while the SPF-algorithm runs in parallel (see Fig. 6). The semantic trajectory planner is triggered by every new semantic image and runs at 1 Hz (detailed information on the computational load of the semantic trajectory planner are summarized in Tab. III). As presented in Sec. III we search in the graph the trajectory that brings the robot the closest to the global goal. As long as this trajectory is not significantly longer than the shortest path we track the trajectory based on semantic information. The combination of both approaches makes the framework very powerful as the semantic planners enables a longer range perception and allows building of a global map over very long time horizons while the SPF planner allows approaching goals that are not on streets or if tracking a road would be a significant detour.

V. EXPERIMENTAL RESULTS

For this work we conducted two different sets of experiments. The first set presents simulated results, that will highlight certain aspect of the approach while the second set, real-world experiments from a quadrotor platform controlled in closed loop, present results of the overall approach and real-world limitations.

A. Simulation results

The simulations have been conducted in urban environments with a road network of a total length of 3.8 km. All simulations leveraged the real flight stack that as well runs on the actual quadrotor platform, which included low-level control, perception and state estimation. Both, sensor simulation and physics were simulated with a high fidelity Unity3D[3] environment (see Fig. 7-b). We conducted four different simulated experiments. The first one is a single flight while tracking five waypoints, demonstrating the general capabilities with respect to the sole SPF-planner. In the next two simulated experiments we demonstrate specific capabilities of our approach.

[3]Game engine by Unity Technologies: https://unity3d.com

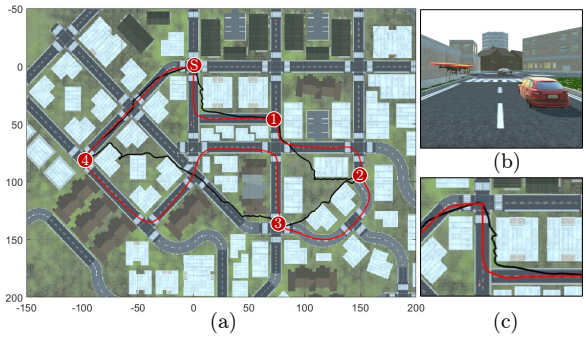

Fig. 7. A top-view of the urban environment with a total road length of 3.8 km. Start location (S), waypoints (#), and trajectory paths (black - shortest path planner, red - our planner) are highlighted. b) Camera image of the high-fidelity city environment. c) A close-up view of the two trajectories. Due to the short visibility, the naive planner (black) is forced to repeatedly dodge the building. Our planner (red) tracks the road smoothly and at a significantly higher speed.

TABLE II

COMPARING OUR APPROACH AND A SHORTEST PATH PLANNER. WHILE THE SHORTEST PATH PLANNER FINDS A SHORTER TRAJECTORY, OUR APPROACH ENABLES FLIGHT WITH A MUCH HIGHER SPEED, RESULTING IN A SIGNIFICANTLY SHORTER TOTAL TIME.

	Total Distance	Total Time	Mean Speed	Max. Speed
Our Approach	764.3 m	**173.9 s**	**4.20 m s^{-1}**	**5.01 m s^{-1}**
Ryll et al. [4]	**681.4 m**	530.5 s	1.22 m s^{-1}	2.23 m s^{-1}

1) Single Flight: For this experiment, the MAV navigated to five waypoints in the initially unknown urban environment and then returned to the start location. The waypoints were defined with respect to the start position and the resulting flight paths are depicted on an top-view image of the environment in Fig. 7-a.

The tracked trajectory of the semantic trajectory planner is highlighted in red and the SPF-planner [4] is highlighted in black. A comparison between the two approaches is presented in Tab. II. First and most obvious, we can see that our approach is able to track roads. As expected, we can see from Tab. II that the SPF-planner finds a path that is approximately 10 % shorter. Our approach finds a longer path due to a detour taken between waypoint 3 and 4. This detour was taken because the road map had not yet been fully built and the shorter path was not yet known to the planner. Despite the longer path, our approach outperforms the shortest-path planner by achieving a higher average and maximum speed. Thanks to the higher velocity, our approach needed only one third of the time to successfully complete the mission.

2) Graph Representation and Graph Search: In this experiment we want to demonstrate the benefits of representing the data in a graph structure over representation in common voxel grid representation. We assumed full knowledge of the environment and compared the pure path search time for SPF-algorithm and RRT* (Rapidly-exploring random tree*) in the voxel grid representation with the search in the learned graph representation. The RRT*-algorithm was stopped as

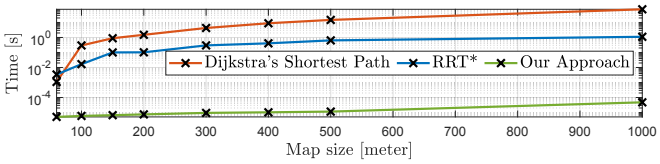

Fig. 8. A comparison of the trajectory search time in a geometric voxel-grid representation with Dijkstra's Shortest Path First (SPF) algorithm (red), RRT* (blue) and our sparse graph representation (green) over increasing map sizes. SPF stopped as soon as a feasible path is found.The trajectory search in the graph is significanlty faster than the other two grid searches.

soon as any feasible trajectory was found. We randomly picked 1000 locations in the city environment and compared the search time at different map sizes (see Fig. 8). The graph search is generally faster by two or three orders of magnitudes than the other two approaches. More important, it becomes obvious that the search in the voxel grid with both SPF and RRT* becomes infeasible in real-time for larger maps. For the largest representation ($1000\,\mathrm{m}$ by $1000\,\mathrm{m}$), a query needed an average of $75.52\,\mathrm{s}$ with SPF and $1.11\,\mathrm{s}$ with RRT* while the same search in the graph representation only needed $47\,\mu\mathrm{s}$. The results were computed on the flight computer equipped with an i7-8650U CPU. We would like to emphasize that our planner, unlike the voxel-based planners, returns a smooth path that can be directly tracked by a quadrotor.

3) Benefit of Map Learning: In this last simulated experiment we want to demonstrate the map inferring capabilities and how the general task benefits from our approach. As presented in the first simulated experiment (see Sec. V-A.1) we saw that the overall trajectory length of our approach was longer, compared to the baseline (SPF) planner, mainly due to the detour taken. Both approaches seek a shortest path in their own representation of the environment. For this experiment, we randomly placed 40 waypoints in the city environment and had the drone autonomously navigate to them, allowing our approach to fully disclose the map over time and to infer the full graph network representation of the environment. As shown in the previous experiment, a voxel grid representation of the complete environment cannot be used for trajectory planning as the computational demand is too high for the real-time trajectory planning. We then compared our planner with the SPF-planner that uses a local ($60\,\mathrm{m}$ by $60\,\mathrm{m}$) sliding-window map. The results while tracking 100 randomly placed waypoints, depicted in Fig. 9. Fig. 9-a) show the mean traveled distance between waypoints. Once the semantic planner built a sufficiently rich map, it competes with the shortest path planner regarding the found trajectory length. Fig. 9-b) depicts the time needed between waypoints. Here, the benefits of following semantic planning become obvious. Thanks to the clutter free environment higher speeds are possible, resulting in shorter flight times between waypoints.

4) Robustness to non-flat surfaces: As discussed in Sec. III-A.1 the projection of the semantic classes assumes local flatness. To test the robustness to this simplification we

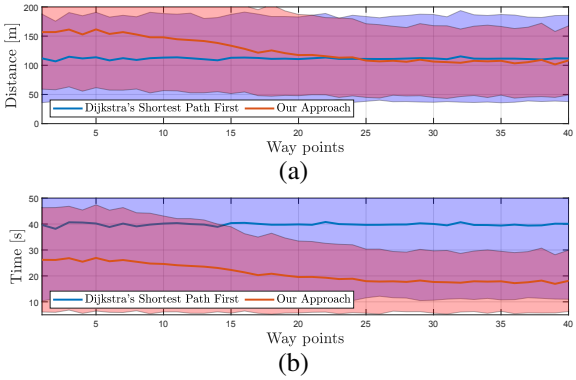

Fig. 9. A comparison of the naive planner (shortest path) against our approach. For this representation 100 trials have been averaged. Our approach infers the road map over time and reaches comparable distances as the naive planner while reaching the waypoints significantly faster. a) Traveled distance between waypoints. b) Average time between two waypoints.

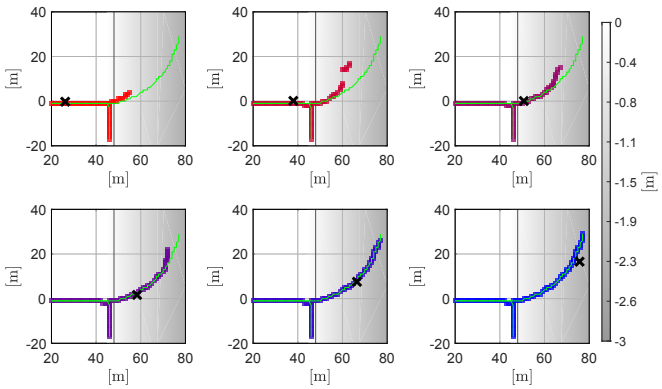

Fig. 10. Comparing road centerline estimation (red-blue) with ground truth (green) on a non-flat ground while tracking road. The right part of the map (altitude indicated by background color) declines with $5°$. The black 'x' indicates the MAV position. Far distant projections are estimated incorrectly. While approaching initially incorrectly estimated segments, the position is corrected, with subsequent measurements over time. The altitude is indicated in gray.

simulated an uneven surface and compared the estimation of the road centerline with the ground truth. The results are depicted in Fig. 10. While the initial part of the road is flat, the road tilts downwards by $5°$ as soon as the curve element starts. By simple geometric reasoning it is clear that the centerline distance will be underestimated if the road tilts downwards and overestimated if the tilting is positive. Additionally the error becomes larger with an increasing distance to the road. As the MAV is tracking the road, the position of the curve in the road is initially underestimated. As the MAV approaches the curve in the road the error is corrected. A very similar scenario is tested in Sec. V-B.2 on the actual platform with similar results.

B. Real flight results

In this section, we present two experiments of fully autonomous flights in an urban environment. The experiments have been conducted on a custom quadrotor MAV platform equipped with an D435i Intel RealSense depth camera (RGB image and depth image: 640 by 480 pixel, 30 Hz), and a quad-core Intel NUC flight computer with an i7-8650U CPU.

TABLE III

COMPUTATIONAL UTILIZATION ON THE INTEL NUC (I7-8650U CPU) DURING REAL WORLD EXPERIMENTS. SEMANTIC SEGMENTATION DOMINATES THE COMPUTATIONAL LOAD OF THE ALGORITHM.

	Mean Value	Standard deviation
Semantic segmentation	794.1 ms	120.4 ms
Skeletonization	72.3 ms	16.1 ms
Dominant point detection	4.1 ms	3.0 ms
Graph search	$\ll$1 ms	$\ll$1 ms
Total	903.8 ms	148.7 ms

The Intel D435i was equipped with an infra-red projector, which was disabled for our use — making it a passive sensor. The total weight of the platform is approximately 1.48 kg (excluding battery) allowing a flight time of 6 min. The computational utilization of the flight experiments on the Intel NUC flight computer are summarized in Tab. III. On the basis of the results of the first experiment we will discuss the precision of the map building process in a real-world environment. The second results will demonstrate the benefits with respect to mission time. To stress the semantic planner the experiments have been conducted in an environment that is clearly not flat but with a significant change of terrain elevation. We recommend the reader to watch the attached video to fully appreciate the experiments. Finally, we would like to remind the reader that in this work we described our approach on the example of mapping roads, which is handy in an urban environment, but this approach can be easily extended to different environments and goals, for example the much more general class of *free space*.

1) Experiment 1: Long distant exploration: The quadrotor's task was to autonomously fly to two given waypoints (see Fig. 11) specified with respect to the take-off location and then return to the goal location. The take-off location was on a street while both waypoints (P1(80m, 40m), P2(-5m, 78m)) where close but not on the street, requiring the vehicle to leave streets during the mission. A top-view image of the experiment, the mapped road center-line and the actually tracked trajectory are depicted in Fig. 11. The total trajectory length from take-off to landing was 375 m. The overall quality of the road center-line is good with locally worse results (e.g., in the upper left corner and near the starting location). The decrease of mapping quality is originated by incorrect classifications during semantic segmentation (see Fig. 12. Here we saw two effects causing erroneous segmentation results. First, the experiments were conducted in winter, where vegetation is reduced and less colorful, causing vegetation to be classified as road. Second, most of-the-shelf segmentation pipelines are trained with dash-cam videos. As soon as the video stream significantly differs from standard dash-cam videos (e.g., quadrotor not centered on road, extreme attitude angles) the segmentation quality drops.

Two key points can be drawn from this experiment. First, let us compare the perception ranges (see last plot in Fig. 11). In our experience, the Intel RealSense D435i depth camera provided the best and most reliable depth perception of up to 10 m in outdoor environments. While the semantic

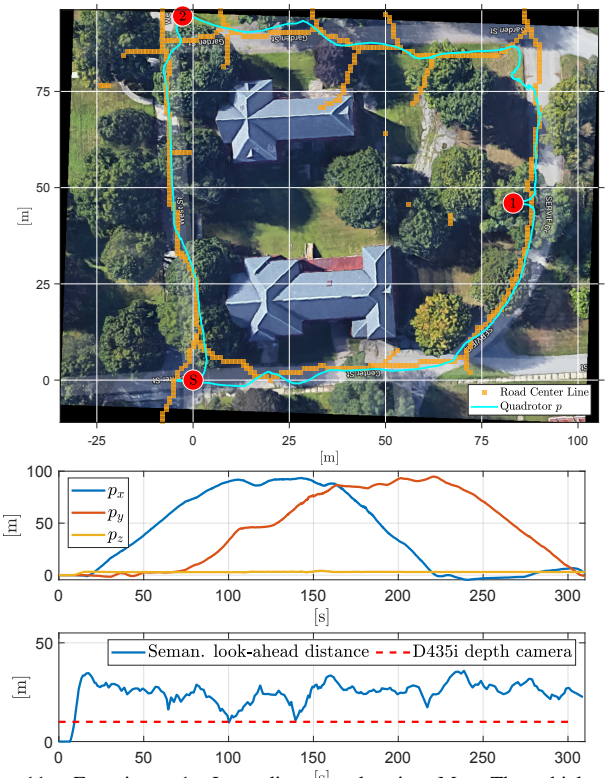

Fig. 11. Experiment 1 - Long distant exploration: Map: The vehicle took off at starting location 'S' and was commanded to fly fully autonomously towards waypoints '1' and '2' and to return to the starting location. The detected center-line of the road is highlighted in orange, while the quadrotor position is printed in cyan. Plot 1: Position of the quadrotor along the trajectory. Plot 3: Perception range of semantic planner and depth camera.

segmentation planner pipeline integrated road pixels of up to a distance of 45 m it required multiple measurements until newly seen areas appeared in the planning graph. We now compared the perception range of the depth camera with the range of the planning graph while flying into unknown space. Our approach provided a mean perception range of 25.3 m and a max range of 35.7 m during the experiment, an average perception increase of a factor of 2.5. Second, our approach was able to generate a sparse planning graph representation of the environment. The final learned graph representation consists of 26 nodes and 26 edges, allowing for an highly efficient trajectory search (in avg. 8.2 elements per edge).

2) Experiment 2 - Out and back: In the second experiment we used the same start and goal location as in Experiment 1 (see Fig. 11) but only tracked waypoint 1 and returned afterwards. The goal of the experiment was to demonstrate that leveraging the learned graph allows a quicker completion of the task. The results are summarized in Fig. 13, where the exploration phase (flight through unknown space) is highlighted in yellow and the exploitation phase (flight through known space) is highlighted in green. From the first plot in Fig. 13 it becomes obvious that the length of the trajectory during exploration is longer than on the way back (130.3 m vs. 118.12 m). The second plot depicts that the vehicle's velocity is increased during the return (1.28 m s^{-1} vs. 1.58 m s^{-1}). Thanks to leveraging the semantic graph planning the mission time during exploitation is decreased by 29 % (103 s vs 72.9 s).

Fig. 12. Semantic segmentation quality during flight. In the top row semantic segmentation quality is high - Roads are well detected (lila). In the bottom row the semantic segmentation quality of roads is worse, large parts of the grass are segmented as road.

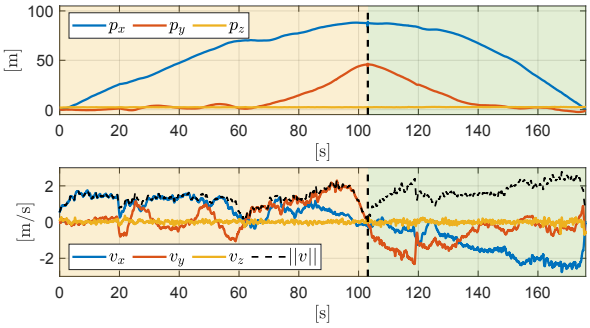

Fig. 13. Experiment 2: Exploration phase (flight into unknown space) is highlighted in yellow and exploitation phase (returning to start location) is highlighted in green. Top: Position of quadrotor during mission. Bottom: Component wise and norm of velocity during mission. The mean velocity is higher during exploitation than during exploration.

VI. Conclusion and Future Work

In this paper, we demonstrated how to use semantic information to improve the perception-estimation-planning loop for high speed MAV flight in urban environment. Specifically, we used semantic information to extend perception range and demonstrated how to learn a sparse representation of an environment in order to plan long distance trajectories. We presented both simulated and real-world results which showed how our approach outperforms standard, shortest-path planners. Additionally we aim to show how the algorithm works in other environments beyond the city and road environment. Furthermore, we aim to avoid the flatness constraint, which works well in city environments but might cause problems in less structured environments.

References

[1] S. Schopferer and F.-M. Adolf, "Rapid trajectory time reduction for unmanned rotorcraft navigating in unknown terrain," in *2014 International Conference on Unmanned Aircraft Systems*. IEEE, 2014.

[2] K. Mohta, K. Sun, S. Liu, M. Watterson, B. Pfrommer, J. Svacha, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Experiments in fast, autonomous, GPS-denied quadrotor flight," *arXiv preprint arXiv:1806.07053*, 2018.

[3] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1.

[4] M. Ryll, J. Ware, J. Carter, and N. Roy, "Efficient trajectory planning for high speed flight in unknown environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[5] T. K. Amukele, L. J. Sokoll, D. Pepper, D. P. Howard, and J. Street, "Can unmanned aerial systems (drones) be used for the routine transport of chemistry, hematology, and coagulation laboratory specimens?" *PloS one*, vol. 10, no. 7, 2015.

[6] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[7] Y. Song, S. Nuske, and S. Scherer, "A multi-sensor fusion MAV state estimation from long-range stereo, IMU, GPS and barometric sensors," *Sensors*, vol. 17, no. 1, 2017.

[8] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics & Automation Magazine*, no. 99, 2018.

[9] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA guidance, navigation and control conference and exhibit*, 2007.

[10] S. L. Waslander, G. M. Hoffmann, J. S. Jang, and C. J. Tomlin, "Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005.

[11] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization." in *Robotics: Science and Systems*, vol. 1, 2015.

[12] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[13] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, 2018.

[14] K. Ok, K. Liu, K. Frey, J. P. How, and N. Roy, "Robust object-based slam for high-speed autonomous navigation," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[15] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[16] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, Apr 2018.

[17] F. Tupin, B. Houshmand, and M. Datcu, "Road detection in dense urban areas using sar imagery and the usefulness of multiple views," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 11, 2002.

[18] C. Poullis and S. You, "Delineation and geometric modeling of road networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 2, 2010.

[19] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, 2014.

[20] Y. Jiang, F. Gao, and G. Xu, "Computer vision-based multiple-lane detection on straight road and in a curve," in *2010 International Conference on Image Analysis and Signal Processing*. IEEE, 2010.

[21] K. Baass, "Use of clothoid templates in highway design," in *Transportation Forum 1*, 1984.

[22] D. Meek and D. Walton, "Clothoid spline transition spirals," *Mathematics of Computation*, vol. 59, no. 199, 1992.

[23] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, p. 5257, Mar. 2002.

[24] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, 1984.

[25] Y. Li, S. Wang, Q. Tian, and X. Ding, "A survey of recent advances in visual feature detection," *Neurocomputing*, vol. 149, 2015.

[26] T. P. Nguyen and I. Debled-Rennesson, "Decomposition of a curve into arcs and line segments based on dominant point detection," in *Scandinavian Conference on Image Analysis*. Springer, 2011.