DeepSemanticHPPC: Hypothesis-based Planning over Uncertain Semantic Point Clouds

Yutao Han*, Hubert Lin*, Jacopo Banfi*, Kavita Bala, and Mark Campbell

***Abstract*— Planning in unstructured environments is challenging – it relies on sensing, perception, scene reconstruction, and reasoning about various uncertainties. We propose DeepSemanticHPPC, a novel uncertainty-aware hypothesis-based planner for unstructured environments. Our algorithmic pipeline consists of: a deep Bayesian neural network which segments surfaces with uncertainty estimates; a flexible point cloud scene representation; a next-best-view planner which minimizes the uncertainty of scene semantics using *sparse* visual measurements; and a hypothesis-based path planner that proposes multiple kinematically feasible paths with evolving safety confidences given next-best-view measurements. Our pipeline iteratively decreases semantic uncertainty along planned paths, filtering out unsafe paths with high confidence. We show that our framework plans safe paths in real-world environments where existing path planners typically fail.**

I. Introduction

Path planning for complex outdoor environments is challenging due to the unstructured nature of environments that do not fall neatly into discretized space. Moreover, different terrain surface types can be difficult to detect with traditional sensing modalities. In indoor environments, a grid space representation with lidar sensors is sufficient [1]–[3]. Outdoor environments exhibit complex geometries and surface types, which are difficult – if not impossible – to differentiate using just lidar data. Therefore, a more flexible representation of the scene, surface classification using computer vision techniques, and scene reasoning are necessary.

Previous work for outdoor planning has focused on classifying terrain and surface roughness using SVM classifiers [4], [5], neural networks [6], [7], and other computer vision techniques [8], [9]. While these techniques can differentiate between simple terrain types, they do not model the inherent uncertainties and ambiguities in complex scenes where it can be difficult to differentiate between terrain types (e.g., an offroad robot driving through a patch of grass with small rocks). Many current outdoor planning approaches still rely on grid maps which do not model the complex geometry of an outdoor scene (e.g., a field with irregular bumps and rocks) [10], [11]. Recent work has modeled outdoor maps for planning with a point cloud [12], which is a more flexible and suitable map for unstructured scenes; however, [12] uses traditional lidar sensing which cannot differentiate between different surface types as broadly as computer vision.

*Equal contribution; order determined randomly.
All the authors are with Cornell University, Ithaca NY, USA {yh675,hl2247,jb2639,mc288,kb97}@cornell.edu.
This work is funded by the ONR under the PERISCOPE MURI Grant N00014-17-1-2699.

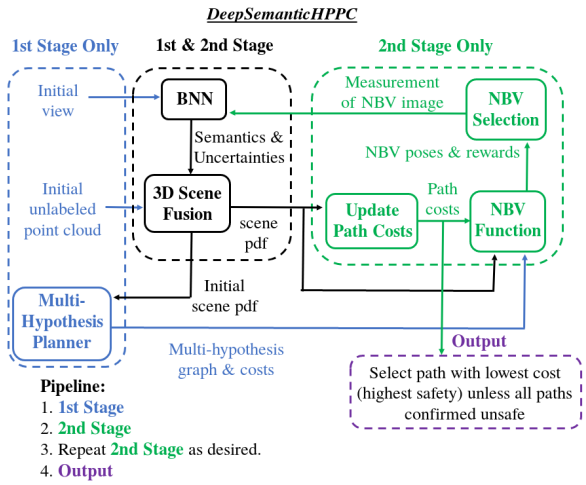


Fig. 1: The DeepSemanticHPPC pipeline. (1) In the first stage, initial inputs are used to generate a multi-hypothesis graph of possible paths. (2) In the second stage, the uncertainty in the scene is reduced and path costs are updated. (3) The second stage is repeated for a set number of iterations. This is terminated early if a safe path is confirmed or all paths are confirmed as unsafe. (4) A path is selected.

In this paper we present DeepSemanticHPPC (Deep Semantic Hypothesis-based Planner over Point Clouds), a novel algorithmic pipeline for planning over uncertain semantic point clouds, which leverages a Bayesian neural network (BNN) [13], [14] to extract principled estimates of segmentation uncertainty. This allows our framework to reason about ambiguous terrain as well as robustly handle false positive detections by taking additional measurements to reduce semantic uncertainty in the scene. However, each measurement is costly due to the computationally expensive nature of Bayesian neural networks operating on a robotic platform with limited computing power. Our planner hence employs next-best-view (NBV) techniques [15]–[17] to select the best possible measurements to reduce scene uncertainty while minimizing the total number of measurements taken. DeepSemanticHPPC includes:

- the employment of a deep Bayesian neural network [13], [14] to obtain surface and obstacle semantics with uncertainty estimates for unstructured outdoor environments;
- a flexible point cloud scene representation;
- a next-best-view planner which minimizes the uncertainty of terrain semantics using *sparse* visual measurements;
- a hypothesis-based path planner (extending [12]) that proposes multiple kinematically feasible paths with evolving safety confidences given the NBV measurements.

Experimental results with real environments show that our pipeline is able to plan safe paths in real-world environments where existing path planners typically fail. Fig. 1 illustrates DeepSemanticHPPC. In the first stage, a multi-hypothesis planner generates multiple hypotheses of possible safe paths given a scene belief. In a second stage, a NBV function calculates NBV poses and associated rewards. These poses and rewards are input to a NBV selection block which selects the best *feasible* NBV. A BNN extracts from the NBV measurement semantic segmentations and associated uncertainties, which are used to generate a new scene belief. The new scene belief reduces hypothesis uncertainty, and the second stage is repeated for a set number of iterations. Finally, a safe path (hypothesis) with high confidence selected; if all paths are determined to be unsafe, then no path is selected. The algorithm terminates once a path is confirmed safe or all paths are confirmed unsafe.

II. Background

A. RRT-Based Non-Holonomic Planning over a Point Cloud

We build upon an existing rapidly-exploring random tree (RTT) [18] based planner for finding kinematically feasible trajectories over non-planar point cloud environments [12]. 6D robot poses are expressed by transformation matrices belonging to the Special Euclidean Group SE(3). A matrix $\mathbf{T}_{\text{MR}}$ specifies the position and orientation of a robot-fixed coordinate frame R expressed in a given reference map frame M. [12] considers the following planning problem: given start and goal poses $\mathbf{T}_{\text{MS}}$, $\mathbf{T}_{\text{MG}}$ and a point cloud $\mathcal{M} = \{\mathbf{m}^i\}$ with $\mathbf{m}^i \in \mathbb{R}^3$, compute a connecting trajectory $\pi : \mathbb{R}_{>0} \rightarrow \text{SE(3)}$. The trajectory has to satisfy a number of constraints up to a given degree of approximation: contact with the terrain surface, static traversability (e.g. bounded roll and pitch angles), and kinematic constraints – including bounded continuous curvature. Trajectories are represented as piecewise continuous functions in the 6D space of robot poses, and are specified by a sequence of nodes $\widehat{\pi} = [\mathcal{N}^k]$, where each $\mathcal{N}^k$ is a tuple $(\mathbf{T}_{\text{MR}^k}, \tau^k, \mathbf{w}^k, \kappa^k)$. Here, $\mathbf{T}_{\text{MR}^k}$ is a 6D pose attached to the terrain surface, $\tau^k \in [0, 1]$ is the associated static traversability value, $\mathbf{w}^k$ is a parameter vector specifying a short *planar* trajectory segment connecting $\mathbf{T}_{\text{MR}^k}$ to the next pose in the sequence, and κ^k is the curvature at the beginning of the trajectory segment. The vector $\mathbf{w}^k$ specifies a trajectory segment as a cubic curvature polynomial [19] evolving along the planar patch defined by the xy plane of the coordinate frame R attached to $\mathbf{T}_{\text{MR}^k}$. The end point of such trajectory segment gives the subsequent pose $\mathbf{T}_{\text{MR}^{k+1}}$ through a projection on the terrain surface via $f : (\mathcal{M}, \mathbf{T}_{\text{MR}}) \mapsto \mathbf{T}_{\text{MR}}$; f queries $\mathcal{M}$ for the K nearest-neighbors of such end point, which can be thought of as the points the robot will lie on at $\mathbf{T}_{\text{MR}^{k+1}}$ (K depends on the size of the robot and on the point cloud density). We use $\phi(\mathcal{N}^{k+1})$ to denote such points.

Leveraging the above trajectory representation, [12] proposes to define a small set of *motion primitives* (short trajectory segments) and use them to grow two RRTs, one from the start pose and one from the goal pose, and iteratively try to connect them. Each new pose is associated with a node $\mathcal{N}^k$, which is accepted in the tree only if $\tau^k > 0$. [12] also proposes a technique to derive a better trajectory (in terms of smoothness and distance) starting from an initial one. This second optimization stage is not explicitly considered in this work, because it can be easily generalized and applied to all "safe" trajectories according to our method.

B. Segmentation with Bayesian Neural Networks

Although segmentation networks for 3D point clouds exist [20]–[24], 3D data repositories are focused on object recognition and part segmentation (e.g. [25]) or only contain a small number of scenes [26]. In contrast, existing large-scale image segmentation datasets [27]–[30] contain varied surfaces and obstacles in diverse real-world outdoor scenes. Therefore, we leverage a state-of-the-art image segmentation network [31] (Section IV-A), and update the point cloud environment from per-pixel image segmentations (Section IV-B). Furthermore, we augment the network to estimate output uncertainty, allowing uncertainty in surface predictions to be stored in the point cloud. Uncertainty in surface type is used to guide path-safety evaluation.

At inference time, forward passes with active dropout layers can be interpreted as an approximation of the posterior distribution of model weights [13], [14]. The uncertainty of predictions can be computed by taking the sample standard deviation across multiple forward passes. For each pixel $(i, j)_X$ of image X, the mean softmax vector over T forward passes is:

$$\mathbf{p}^{(i,j)_X} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_t^{(i,j)_X}(y|X) \tag{1}$$

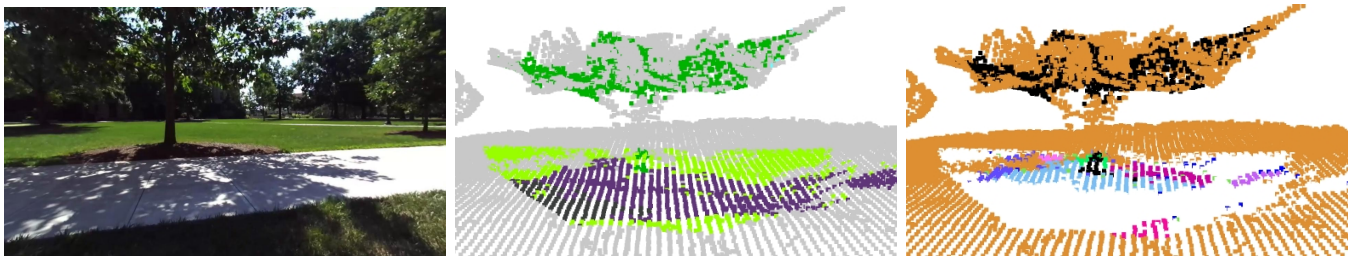
where $\mathbf{s}^{(i,j)_X}(y|X) \in \mathbb{R}^C$ is the softmax output of the network. The corresponding uncertainty vector on $\mathbf{p}^{(i,j)_X}$ is:

$$\boldsymbol{\sigma}^{(i,j)_X} = \sqrt{\frac{\sum_{t=1}^{T} \left(\mathbf{s}_t^{(i,j)_X}(y|X) - \mathbf{p}^{(i,j)_X}\right)^2}{T-1}} \tag{2}$$

In our framework, image X corresponds to a view with known camera parameters. $\mathbf{p}^{(i,j)_X}$ and $\boldsymbol{\sigma}^{(i,j)_X}$ are combined with existing measurements for each point $\mathbf{m} \in \mathcal{M}$ that maps to pixel $(i, j)_X$ (Section IV-B).

III. Approach Overview

The planner presented in Section II-A does not leverage critical semantic information about terrain types. In this work, we assume an initial point cloud is given in the form $\mathcal{M} = \{\mathbf{e}^i\}$, where each element $\mathbf{e}_i$ is a tuple $(\mathbf{m}^i, \mathbf{p}^i, \boldsymbol{\sigma}^i)$. Here, $\mathbf{m}^i \in \mathbb{R}^3$ as before; $\mathbf{p}^i = (p_1^i, p_2^i, \ldots, p_C^i)$ is a vector specifying the probabilities that the point belongs to each one of the C possible semantic classes (gravel, water, etc.); $\boldsymbol{\sigma}^i = (\sigma_1^i, \sigma_2^i, \ldots, \sigma_C^i)$ is a vector specifiying the uncertainties of $\mathbf{p}^i$, as discussed in Section II-B. Points are initialized with uniform semantic probabilities and maximum uncertainties. Updating the semantic point cloud is discussed in Section IV-B. Fig. 2(b) shows a pointcloud obtained in



(a) (b) (c)

Fig. 2: An example point cloud. (a) Image view of a portion of the environment. (b) Point cloud colored with the most likely class predicted from image (a) (bright green: "grass"; dark green: "tree"; purple: "sidewalk"; dark grey: "road"; light grey: no information available). All the classes except "tree" belong to the set S. The region around the tree is actually mulch/woodchips, which should be classified as "dirt" (belonging to U). (c) Point cloud colored to show safe (white), unsafe (black), and unclear regions $\mathcal{R}$ (random colors).

a real (ambiguous) environment, where each point $\mathbf{m}^i$ is associated with the color corresponding to the class label j whose p_j^i is maximum.

We assume that the semantic classes have been partitioned into two sets: the safe set S (e.g. gravel, grass) and the unsafe set U (e.g. water, snow). For each point $\mathbf{m}^i$, the points are defined as $p_S^i = \sum_{j\in S} p_j^i$, $p_U^i = 1 - p_S^i = \sum_{j\in U} p_j^i$, and $\sigma^i = \min(\sqrt{\sum_{j\in S} \sigma_j^{i2}}, \sqrt{\sum_{j\in U} \sigma_j^{i2}})$. Each point is then classified as:

- **safe** if $p_S^i - w_\sigma \sigma^i \geq \theta_s$;
- **unsafe** if $p_U^i - w_\sigma \sigma^i \geq \theta_u$;
- **unclear** otherwise.

Intuitively, this implies that points are safe/unsafe given high probability (p_S^i, p_U^i) and low uncertainty (σ^i), and unclear otherwise. w_σ, θ_s, θ_u are defined by the mission planner (with $1-\theta_s < \theta_u$). We use $\mathcal{M}_{\text{safe}}$, $\mathcal{M}_{\text{unsafe}}$, $\mathcal{M}_{\text{unclear}}$ to denote the partition of $\mathcal{M}$ obtained from the above classification. Note that a point is labeled safe (unsafe) even when a significant uncertainty on p_S^i (p_U^i) is present, provided that there is small uncertainty on p_U^i (p_S^i). This is captured by the $\min$ in the definition of σ^i. For example, the network may be uncertain between gravel and grass (both safe), but it is sure that the point is neither water nor snow (both unsafe).

Consider now a trajectory $\widehat{\pi} = [\mathcal{N}^k]$, and recall that $\phi(\mathcal{N}^k)$ denotes the set of points on which the robot lies when at pose $\mathbf{T}_{\text{MR}^k}$. Depending on the semantic information initially available, it might be very difficult –if not impossible– to immediately find a trajectory whose node points $\phi(\mathcal{N}^k)$ all belong to $\mathcal{M}_{\text{safe}}$. DeepSemanticHPPC works in two stages:

1) **Compute a set of candidate paths** traversing different *unclear regions*, and
2) **Reduce the uncertainty of such paths** by taking *new views* in the proximity of the robot's starting position of the *most promising path*.

We relax the path planning problem in a natural way – instead of reaching a specific goal pose, we require the robot to reach a goal pose region G defined around $\mathbf{T}_{\text{MG}}$. Then, the points in $\mathcal{M}_{\text{unclear}}$ are organized into a set $\mathcal{R}$ of *unclear regions*. To build the set $\mathcal{R}$, we use the following two-stage clustering process: first, we use DBSCAN [32] to perform a large-scale clustering of the points in $\mathcal{M}_{\text{unclear}}$, obtaining a set of large unclear regions $\widehat{\mathcal{R}}$. Then, the points of each $\hat{r} \in \widehat{\mathcal{R}}$ are further partitioned according to their most likely class (treating the points not associated with any prediction as belonging to a special class); DBSCAN is called again on each partition. Fig. 2(c) shows the result of this process on our example with $\theta_s = 0.9, \theta_u = 0.3, w_\sigma = 3$.

The remaining task is to compute a set of candidate paths from $\mathbf{T}_{\text{MS}}$ to G. Section IV-C presents a variant of a standard RRT algorithm which is able to construct multiple hypothesis for the safest path, traversing different unclear regions. These paths are stored in the form of a directed graph $G = (V, A)$, where each $v \in V$ is associated with a potential trajectory node $\mathcal{N}^v$ and each $a \in A$ represents the existence of a short trajectory segment connecting two poses. The cost for each node is based on how far it is from satisfying our safety constraint: $\bar{p}^v = \frac{1}{|\phi(\mathcal{N}^v)|} \sum_{i\in\phi(\mathcal{N}^v)} \min(1, \max(0, \theta_s - p_S^i + w_\sigma \sigma^i))$ for for each $v \in V$. However, if the node region sufficiently intersects with an unsafe region, the cost is infinite; and if the node lies entirely in a safe region, the cost is zero.

Summarizing, $c(v)$ is defined as:

$$c(v) = \begin{cases} 0 & \text{if } |\mathcal{M}_{\text{safe}} \cap \phi(\mathcal{N}^v)| = |\phi(\mathcal{N}^v)| \\ \infty & \text{if } |\mathcal{M}_{\text{unsafe}} \cap \phi(\mathcal{N}^v)| \geq \phi_v \\ \bar{p}^v & \text{otherwise,} \end{cases} \tag{3}$$

where ϕ_v is a user-defined threshold. The first condition can be relaxed for the nodes in close proximity of the starting pose. Although a vertex with infinite cost is never obtained when the candidate paths are initially computed, its cost might tend to infinity when additional views are taken during the NBV stage (Section IV-D). A predefined number of NBV iterations are run. At each iteration, the m most promising (shortest) paths according to the above cost function are computed by a k-shortest paths algorithm (we use Yen's [33]). The associated vertices v such that $0 < c(v) < \infty$ are then considered in the function that computes the best additional view. The value of m is also decided by the mission planner: $m = 1$ corresponds to an aggressive setting, while $m > 1$ should be preferred given a large temporal budget for taking additional views.

IV. Technical Details

A. Predicting Semantic Labels

We curate a segmentation dataset for outdoor navigation in unstructured environments from the existing large-scale COCO panoptic dataset [27]. First, images of unstructured outdoor scenes are selected using a Places365 [34] classifier. An image is kept if (a) the classifier's top one (highest) prediction is an unstructured outdoor category with $> 50\%$ probability, or (b) two or more of the top five predictions are unstructured outdoor categories. Second, the 133 categories in COCO panoptic are merged: (a) all outdoor terrains (e.g. grass, dirt, snow, pavement) are retained; (b) obstacles are merged into four categories: fixed obstacles (e.g. buildings), moving human-made obstacles (e.g. vehicles), humans, and animals; (c) all indoor categories are removed. Our final dataset consists of 34K training images with 22 categories. Our navigation segmentation categories (from COCO panoptic), and filtered list of COCO panoptic images are at: https://drive.google.com/file/d/11MIN6n9NJV8IZIvReSgQGXZHS2qcSjvg

For our network architecture, we use DeepLabv3+ [31] with Xception65 [35] backbone augmented with dropout in the middle and exit flow blocks for semantic segmentations. At inference time, 50 forward passes are used to predict semantics and uncertainties (Eqs. (1)-(2)).

B. Associating Semantic Labels to a Point Cloud

Given a viewpoint with known pose and camera intrinics, image segmentation probabilities and uncertainties are mapped to the point cloud. To map pixel $(i,j)_X$:

1) Estimate depth map $\mathbf{D}_X$ for view X.
2) Each pixel is backprojected to a single point corresponding to the center of the projected pixel. This approximation does not hold for pixels with very large depths, but typically produces good results. Backprojected pixel point $\tilde{\mathbf{m}}^{(i,j)_X}$ is computed as:
$$\tilde{\mathbf{m}}^{(i,j)_X} = \mathbf{P}_X [\mathbf{D}_X^{(i,j)} \mathbb{I}_{3\times 3} | [0\ 0\ 1]^T]^T \mathbf{K}_X^{-1} [i\ j\ 1]^T \quad (4)$$
where $\mathbf{K}_X$, $\mathbf{P}_X$ are intrinsic and pose matrices.
3) Each backprojected point is merged with its nearest neighbor $\mathbf{m}^{\text{nn}}$ in the point cloud $\mathcal{M}$ within threshold distance R. If a backprojected point does not have a neighbor within the threshold, the point is discarded.
4) $\mathbf{p}^{(i,j)_X}$ and $\boldsymbol{\sigma}^{(i,j)_X}$ are merged with existing measurements for point $\mathbf{m}^{\text{nn}}$. The combined measurement is the best linear unbiased estimator under the simplifying assumption that the per-class predictions are independent. Given a set of K measurements $\{(\mathbf{p}^{(k)}, \boldsymbol{\sigma}^{(k)})\}$, the combined measurement $(\tilde{\mathbf{p}}, \tilde{\boldsymbol{\sigma}})$ for class c is:
$$\left(\tilde{p}_c = \frac{1}{Z} \sum_{k=1}^{K} w^{(k)} p_c^{(k)} \quad , \quad \tilde{\sigma}_c = \sqrt{\sum_{k=1}^{K} (w^{(k)})^2 (\sigma^{(k)})^2} \right)$$
$$\text{where } w_c^{(k)} = \frac{(\sigma_c^{(k)})^{-2}}{\sum_{c' \in C} (\sigma_{c'}^{(k)})^{-2}} \quad , \quad Z \text{ s.t. } \sum_{c' \in C} \tilde{p}_{c'} = 1 \quad (5)$$

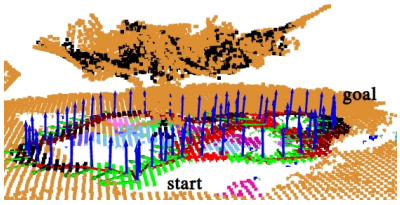


Fig. 3: An example graph $G = (V, A)$, with poses. Vertices with $c(v) = 0$ are in green, while vertices with $0 < c(v) \leq 1$ are in red (the darker, the closer to 1).

C. RRT-Based Multi-hypothesis Planner

The algorithm to compute $G = (V, A)$ starts by building an initial RRT from a root vertex v_s associated with the projected start pose $\mathbf{T}_{\text{MS}}$ via a predefined set of motion primitives. Instead of building two RRTs as is done in [12], we build a single RRT from the start pose and bias the sampling to the point that is closest to the projected ideal goal pose. Sampling is performed on the points in $\mathcal{M}$ not lying in the *forbidden points set* $\mathcal{F}$, which is initialized as $\mathcal{F} \leftarrow \mathcal{M}_{\text{unsafe}}$. Once the first path $\hat{\pi} = [\mathcal{N}^v]$ is found, the algorithm examines which regions in $\mathcal{R}$ are traversed by the vertices of $\hat{\pi}$ by checking their intersections with each set of points $\phi(\mathcal{N}^v)$. Except for the regions containing points belonging to the K-nearest neighbors of $\mathbf{T}_{\text{MS}}$ and $\mathbf{T}_{\text{MG}}$, all regions traversed by $\hat{\pi}$ are placed into the *removal candidates set* $\mathcal{C}$. A heuristic is then used to decide which region(s) should be removed from subsequent planning stages. In this work, we simply remove the largest region $\hat{c}$, and $\mathcal{F}$ is updated as $\mathcal{F} \leftarrow \mathcal{F} \cup \hat{c}$. When $\hat{c}$ is removed, the vertices, including those of $\hat{\pi}$, are removed from the RRT. The algorithm then proceeds to optimize and find a new path to the goal, by continuously expanding the RRT component containing v_s. This time, however, the algorithm also tries to connect (by computing *ad hoc* trajectory segments) new vertices to those that were disconnected from the RRT due to the removal of $\hat{c}$. When a new path is found, the process repeats. If at any iteration, the algorithm finds a path only contained in the regions of $\mathbf{T}_{\text{MS}}$ and $\mathbf{T}_{\text{MG}}$, it can be restarted with a different random seed. The two graphs can be merged at a later stage. Fig. 3 shows an example multi-hypothesis graph computed on the example of Section III (Fig. 2).

If any path computed on the multi-hypothesis graph $G = (V, A)$ has zero cost, the robot can start following that path since all the underlying points lie in $\mathcal{M}_{\text{safe}}$. Otherwise, the robot enters the NBV stage described below.

D. Next-Best-View (NBV) Planning

A set of n viable NBV poses $\mathbf{T}_{\text{viable}} = \{\mathbf{T}_{\text{MV}_1}, \ldots, \mathbf{T}_{\text{MV}_n}\}$ is computed by growing a RRT starting from $\mathbf{T}_{\text{MS}}$. The candidate poses $\mathbf{T}_{\text{viable}}$ are a subsample of the RRT vertices. All points lying within $\mathcal{M}_{\text{unclear}}$ are treated as *unsafe*, and sampling is performed on the safe points lying within a given radius r from $\mathbf{T}_{\text{MS}}$. The robot should not travel too far to take a new view; otherwise it is more appropriate to follow the most promising path of $G = (V, A)$.

A reward $J(\mathbf{T}_{\text{MV}_j})$ is calculated for each candidate pose $\mathbf{T}_{\text{MV}_j}$. The NBV pose $\mathbf{T}_{\text{NBV}}$ is selected by picking the pose with the highest value of J that also admits a safe path back to the current pose.

$$J(\mathbf{T}_{\text{MV}_j}) = \beta_d D + \beta_\gamma \gamma + \beta_{\text{vis}} N_{\text{vis}} + \beta_Q \overline{Q}, \tag{6}$$

where D is a distance metric, γ is a viewing angle metric, N_{vis} is the number of visible vertices from $\mathbf{T}_{\text{MV}_j}$, and $\overline{Q}$ is the average information gain over visible vertices from $\mathbf{T}_{\text{MV}_j}$. The weight $\beta_d, \beta_\gamma, \beta_{\text{vis}}, \beta_Q$ sum to 1 and D, γ, N_{vis}, and $\overline{Q}$ are normalized. Distance and change in viewing angle are used based on the assumption that closeness and view diversity will reduce vertex uncertainty. N_{vis} puts more weight on candidate poses which have higher chances of reducing the uncertainty of multiple segments of the multipath graph $G = (V, A)$. $\overline{Q}$ represents the expected reduction in uncertainty in the graph vertices given $\mathbf{T}_{\text{MV}_j}$. Each of these components are defined as follows.

Begin by defining the set of vertices $v \in V_{\text{NBV}}$, where V_{NBV} is the set of unclear vertices belonging to the m most promising paths (see the end of Section III). For each candidate pose $\mathbf{T}_{\text{MV}_j} \in \mathbf{T}_{\text{viable}}$, only vertices visible from $\mathbf{T}_{\text{MV}_j}$ are considered in calculating the reward. Visible vertices are defined as vertices which occupy greater than a predefined number of pixels in the image plane rendering of the point cloud from $\mathbf{T}_{\text{MV}_j}$. Visible vertices are added to the set $v \in V_{\text{vis},j}$, where $V_{\text{vis},j} \in V_{\text{NBV}}$.

To calculate D, the distances from $\mathbf{T}_{\text{MV}_j}$ to $v \in V_{\text{vis},j}$ are normalized. Since a lower distance should correspond to a higher reward, we subtract the normalized distances from 1. To calculate γ: the set of negative cosine distances of the angle between $\mathbf{T}_{\text{MS}}$ and $\mathbf{T}_{\text{MV}_j}$ to $v \in V_{\text{vis},j}$ are used. N_{vis} is the size of $V_{\text{vis},j}$.

The information gain metric $Q(v)$ is calculated for each $v \in V_{\text{NBV}}$. $Q(v)$ represents the expected reduction in uncertainty for each $v \in V_{\text{NBV}}$, and is a function of the visibility and uncertainty of the points lying in $\phi(\mathcal{N}^v)$.

The visibility $I(v, \mathbf{T}_{\text{MV}_j})$ of a vertex $v \in V_{\text{NBV}}$, given a candidate pose $\mathbf{T}_{\text{MV}_j}$, is the pixel coverage of $\phi(\mathcal{N}^v)$ in the rendered image plane of $\mathbf{T}_{\text{MV}_j}$. Per-point bounding squares of size equal to half the point cloud resolution are used to compute occlusions and pixel coverage for surface points. The number of pixels that $\phi(\mathcal{N}^v)$ occupies in the rendering is the predicted visibility $I(v, \mathbf{T}_{\text{MV}_j})$ of v at $\mathbf{T}_{\text{MV}_j}$.

The uncertainty $\sigma(v)$ of a vertex $v \in V_{\text{NBV}}$ is the average sum of the uncertainties of the points in $\phi(\mathcal{N}^v)$:

$$\sigma(v) = \frac{1}{|\phi(\mathcal{N}^v)|} \sum_{i \in \phi(\mathcal{N}^v)} \sum_{j=1}^{C} \sigma_i^j \tag{7}$$

The information gain metric $Q(v)$ of vertex $v \in V_{\text{NBV}}$ can be formally written as

$$Q(v) = \alpha_i I(v, T_{MV_j}) + \alpha_\sigma \sigma(v), \tag{8}$$

where α_I, α_σ are weights that sum to 1 and $I(v, \mathbf{T}_{\text{MV}_j})$ and $\sigma(v)$ are normalized.

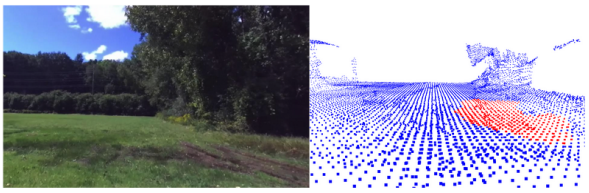


Fig. 4: **Left**: Image from Cass Park in Ithaca. There are multiple different terrains in the scene including grass, mud, and water. **Right**: Annotated safe (blue) and unsafe (red) regions for the Cass Park point cloud.

V. Validation

A. Validation Scenes and Overview

We implement our full pipeline in the AirSim simulator [36]. However, preliminary experiments show that the BNN trained on the real-world dataset (Section IV-A) performs poorly on the synthetic AirSim environment. The full pipeline with a simpler BNN trained on the AirSim environment and corresponding experimental results are shown in the accompanying video.

For real world validation, we collect data for two different scenes using the ZED stereo camera from Stereolabs. The first scene is next to the Mann Library in Cornell University (Fig. 2(a)) and the second scene is at Cass Park in Ithaca (Fig. 4). These scenes are selected due to their varying (but common) terrain types. The scenes are representative of common unstructured outdoor environments without pathological geometry or terrain. The ZED camera API is used to extract depth maps and generate point cloud reconstructions of the scenes. For these scenes, we heuristically select a set of candidate NBV poses instead of growing a RRT from $\mathbf{T}_{\text{MS}}$, and assume a path between these poses and the start pose exists. Candidate NBV poses are selected to be oriented in the general direction of the goal while covering a wide range of the scene. Our method (and baseline methods) choose NBVs from the set of candidate NBV poses.

B. NBV evaluation

To evaluate the performance of the NBV function, we examine the change in uncertainty of the path vertices as the number of NBVs increases. The complete NBV reward function (Eq. **??**) is compared against: (a) random selection, (b) geometry-only reward, and (c) uncertainty-only reward. For the geomtry-only NBV reward, we set $\{\beta_Q\}$ to zero, and for the uncertainty-only NBV reward, we set $\{\beta_d, \beta_\gamma, \beta_{\text{vis}}\}$ to zero. The change in uncertainties summed across all the classes and points for each path vertex averaged over 500 trials is shown in Fig. 5. In our experiments, the full NBV reward weights are set as follows: $\{\beta_d = 0.4, \beta_\gamma = 0.05, \beta_{\text{vis}} = 0.25, \beta_Q = 0.3, \alpha_I = 0.5, \alpha_\sigma = 0.5\}$ for the Mann Library scene, and $\{\beta_d = 0.15, \beta_\gamma = 0.05, \beta_{\text{vis}} = 0.2, \beta_Q = 0.6, \alpha_I = 0.3, \alpha_\sigma = 0.7\}$ for the Cass Park scene. A higher uncertainty weight is assigned to the Cass scene because the boundaries between surface types (e.g. water, mud, grass) are more ambiguous than in the Mann scene.

For both scenes, the full NBV reward function consistently achieves the lowest uncertainty with 2 or more NBVs (Fig.

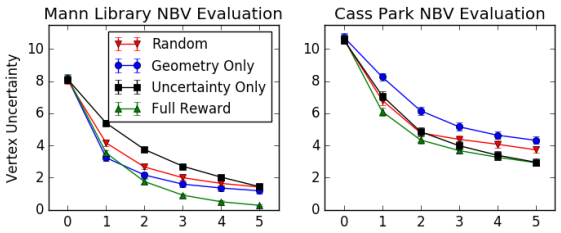


Fig. 5: Change in uncertainty of path vertices (y-axis) as the number of NBV measurements increase (x-axis).

5). This illustrates the importance of both geometry and uncertainty terms in the reward function. In the Mann scene, the baseline reward functions converge to a higher uncertainty than the full reward function. Due to the small size of the scene and the nature of all candidate NBVs being oriented towards the goal pose, random selection performs quite well. It initially outperforms uncertainty-only which does not take into account point visibility or viewpoint diversity, while random sampling implicitly selects diverse views. In the Cass scene, the baseline reward functions converge to a higher uncertainty than the full reward function, except for uncertainty-only which converges to point as the full reward function. The overall uncertainty in the BNN predictions are much higher at Cass park, so heavily weighing the uncertainty in the reward function performs well. In the Mann scene, geometry-only outperforms uncertainty-only, whereas the opposite result holds for the Cass scene. Mann has less inherent ambiguity so geometry terms are more important, while Cass has more ambiguous regions so uncertainty terms are more important.

C. Path Safety Evaluation

To evaluate the real world application of DeepSemanticHPPC, we study the safety of selected paths. We annotate a point cloud (Fig. 4 **right**) with MeshLab [37] into ground truth safe and unsafe regions. For Mann library mulch (dirt) is labeled as unsafe, and for Cass Park mud (dirt) and water are labeled as unsafe. Any path which contains a vertex that overlaps with the unsafe region with over N_{unsafe} points is unsafe. We set $N_{\text{unsafe}} = 4$. Two baselines are considered: (a) B1: planning without semantic information (based on [12]) and (b) B2: planning with semantic information from a single initial view without taking any NBV measurements to reduce path uncertainty. We also study the performance of DeepSemanticHPPC as the number of NBVs increase.

Table I shows the path safety results for the Mann Library and Cass Park scenes over 500 trials. Since both safe and unsafe terrain surfaces can be geometrically similar, baseline B1 cannot reliably avoid unsafe semantic regions. Baseline B2 performs significantly better than B1 because of the inclusion of semantic surface types in the planner. However, because the semantic segmentations can be incorrect, especially in regions with high uncertainty, this planner still plans over unsafe terrain.

Our DeepSemanticHPPC framework is significantly better than the two baselines. NBVs allow the planner to discard unsafe paths as semantic uncertainties decrease. With just one NBV, the percentage of safe paths taken increases drastically from 23.4% to 81.6% (Mann) and increases from 33.4% to 54.0% (Cass). As NBVs increase, the percentage of safe paths selected generally increases while the percentage of unsafe paths selected decreases. With 5 NBVs, 86% (59.2%) of paths selected for Mann (Cass) are safe, and only 3.8% (36.6%) of paths selected for Mann (Cass) are unsafe. With 5 NBVs, uncertainty is sufficiently reduced so that in 10.2% (4.2%) of trials, all multipaths are confirmed to be unsafe for Mann (Cass) and no path is selected. The complexity of the Cass scene is reflected in these results.

Mann	**B1**	**B2**	**1N**	**2N**	**3N**	**4N**	**5N**
Safe %	0	23.4	81.6	79.8	81.4	83.6	**86.0**
Unsafe %	100	76.6	18.4	15.4	11.2	6.6	**3.8**
CS %	N/A	N/A	0	0	4.0	18.0	28.0
CN %	N/A	N/A	0	4.8	7.4	9.8	10.2
Cass	**B1**	**B2**	**1N**	**2N**	**3N**	**4N**	**5N**
Safe %	13.2	33.4	54.0	57.4	57.4	59.0	**59.2**
Unsafe %	86.8	66.6	46.0	41.6	39.8	37.4	**36.6**
CS %	N/A	N/A	0	0	0	0	0
CN %	N/A	N/A	0	1.0	2.8	3.6	4.2

TABLE I: 500 trials of path safety evaluation. The columns are the path planning methods used: **B1** is the planner based on [12], **B2** is the variant of our framework which does not utilize any NBVs, X**N** is DeepSemanticHPPC (ours) with X NBVs. The rows are the metrics: **Safe** is the number of trials where a safe path is selected, **Unsafe** is the number of trials where an unsafe path is selected (lower is better), **CS** is the number of trials where a safe path is confirmed, **CN** is the number of trials where all multipaths are confirmed as unsafe (and no path is selected).

VI. Conclusion

In this paper we presented DeepSemanticHPPC, a novel framework for planning in unstructured outdoor environments while accounting for uncertain terrain types. Our experiments show that DeepSemanticHPPC is able to reduce semantic uncertainty in planned paths and increase the safety of paths planned in environments with unsafe terrains. We plan to implement DeepSemanticHPPC on a robot for physical experiements. Other interesting directions include exploring the ability to build the point cloud online and incorporating geometric uncertainties.

Acknowledgments

We thank Eric Wu (Cornell) and Hadi AlZayer (Cornell) for insightful discussions and assistance with preliminary software implementation.

References

[1] G. J. Stein, C. Bradley, and N. Roy, "Learning over subgoals for efficient navigation of structured, unknown environments," in *Proc. CORL*, 2018, pp. 213–222.

[2] A. Ivanov and M. E. Campbell, "Uncertainty constrained robotic exploration: An integrated exploration planner," *IEEE T Contr Syst T*, vol. 27, pp. 146–160, 2016.

[3] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[4] K. Walas, "Terrain classification and negotiation with a walking robot," *J Intell Robot Syst*, vol. 78, pp. 401–423, 2015.

[5] K. A. Mahadhir, S. C. Tan, C. Y. Low, R. Dumitrescu, A. T. M. Amin, and A. Jaffar, "Terrain classification for track-driven agricultural robots," in *Proc. SYSINT*, 2014, pp. 775 – 782.

[6] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE RAL*, vol. 3, pp. 1695–1702, 2017.

[7] J. A. Delmerico, A. Giusti, E. Mueggler, L. M. Gambardella, and D. Scaramuzza, ""on-the-spot training" for terrain classification in autonomous air-ground collaborative teams," in *Proc. ISER*, 2016.

[8] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Auton Robot*, vol. 18, no. 1, pp. 81–102, 2005.

[9] P. Filitchkin, "Visual terrain classification for legged robots," Ph.D. dissertation, University of California, Santa Barbara, 2011.

[10] M. W. Otte, S. G. Richardson, J. Mulligan, and G. Z. Grudic, "Path planning in image space for autonomous robot navigation in unstructured environments," *J Field Robot*, vol. 26, pp. 212–240, 2009.

[11] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *J Field Robot*, vol. 23, pp. 79–101, 2006.

[12] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *J Field Rob*, vol. 34, no. 5, pp. 940–984, 2017.

[13] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. ICML*, 2016.

[14] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Proc. NIPS*, 2017, pp. 5574–5584.

[15] R. Border, J. D. Gammell, and P. Newman, "Surface edge explorer (SEE): Planning next best views directly from 3d observations," in *Proc. ICRA*, 2018, pp. 6116–6123.

[16] B. Hepp, D. Dey, S. N. Sinha, A. Kapoor, N. Joshi, and O. Hilliges, "Learn-to-score: Efficient 3d scene exploration by predicting view utility," in *Proc. ECCV*, 2018, pp. 437–452.

[17] J. Daudelin and M. Campbell, "An adaptable, probabilistic, next best view algorithm for reconstruction of unknown 3d objects," *IEEE RAL*, vol. 2, no. 3, pp. 1540–1547, 2017.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[19] B. Nagy and A. Kelly, "Trajectory generation for car-like robots using cubic curvature polynomials," in *Proc. FSR*, 2001.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. CVPR*, 2017, pp. 77–85.

[21] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. NIPS*, 2017, pp. 5099–5108.

[22] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. CVPR*, 2018, pp. 4490–4499.

[23] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proc. CVPR*, 2018, pp. 918–927.

[24] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[25] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[26] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, and M. Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark," *ISPRS Annals*, vol. 4, no. 1, pp. 91–98, 2016.

[27] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. CVPR*, 2019, pp. 9404–9413.

[28] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Proc. CVPR*, 2018, pp. 1209–1218.

[29] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proc. CVPR*, 2017, pp. 633–641.

[30] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *Proc. CVPR*, 2015, pp. 3479–3487.

[31] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.

[32] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proc. KDD*, 1996, pp. 226–231.

[33] J. Yen, "Finding the k shortest loopless paths in a network," *Manage Sci*, vol. 17, no. 11, pp. 712–716, 1971.

[34] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017, pp. 1251–1258.

[36] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[37] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool." in *Eurographics Italian chapter conference*, vol. 2008, 2008, pp. 129–136.