Enabling Topological Planning with Monocular Vision

Citation: Stein, Gregory Joseph, Bradley, Christopher, Preston, Victoria and Roy, Nicholas. 2020. "Enabling Topological Planning with Monocular Vision." Proceedings - IEEE International Conference on Robotics and Automation.

As Published: http://dx.doi.org/10.1109/ICRA40945.2020.9197484

Publisher: IEEE

Persistent URL: https://hdl.handle.net/1721.1/137314

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Enabling Topological Planning with Monocular Vision

Gregory J. Stein*, Christopher Bradley*, Victoria Preston*, and Nicholas Roy

***Abstract*— Topological strategies for navigation meaningfully reduce the space of possible actions available to a robot, allowing use of heuristic priors or learning to enable computationally efficient, intelligent planning. The challenges in estimating structure with monocular SLAM in low texture or highly cluttered environments have precluded its use for topological planning in the past. We propose a robust sparse map representation that can be built with monocular vision and overcomes these shortcomings. Using a learned sensor, we estimate high-level structure of an environment from streaming images by detecting sparse "vertices" (e.g., boundaries of walls) and reasoning about the structure between them. We also estimate the known free space in our map, a necessary feature for planning through previously unknown environments. We show that our mapping technique can be used on real data and is sufficient for planning and exploration in simulated *multi-agent search* and *learned subgoal planning* applications.**

I. Introduction

Autonomous navigation is a ubiquitous problem in the field of mobile robotics. In order to reduce the number of actions available to an agent, it is often easier to describe the problem, and subsequently generate plans, using the language of topology. In cases where the map is fully known, recent work in the field leverages topological constraints for a variety of different objectives, such as multi-agent search though known maps [1], in which robotic agents are constrained to navigate through different homology classes to search more efficiently.

Intelligent decision-making, particularly in the context of navigating unknown space, can be modelled as a Partially Observable Markov Decision Process (POMDP) [2]. Topology provides a means of specifying *high-level* actions available to the agent (e.g., "go left around the building" or "enter unknown space through this boundary"), allowing the planner to reduce the computational challenges of the optimization problem implicit in the POMDP. Planning under topological constraints reduces the space of actions, making it easier to plan [3], [4] or encode heuristic or learned priors [5], [6]. Critically however, to plan with topological constraints, a robot must be able to extract them from its environment.

The ubiquity of cameras and the richness of information vision provides makes monocular images an attractive candidate for both informing high-level autonomy and building the maps autonomous agents use to represent their environment. Monocular, feature-based methods have proven to be effective tools for localization and navigation [7] and have shown promise in building occupancy maps sufficient for exploration of small environments [8], [9]. These approaches build sparse point-clouds of visual features, and bin them into voxels to represent obstacles which a robot could use for path planning. However, such approaches have limitations when planning with topological constraints in general; spurious detections can block potential paths, and a lack of detections in featureless regions (e.g., a white wall in a hallway) can lead a robot to believe free space exists where it does not. Resolving the discrepancies between the point cloud and the true underlying geometry of an environment is a challenge for such systems. Furthermore, due to the high dimensionality of point-cloud-based SLAM, robustly eliminating erroneous features is an open problem [7].

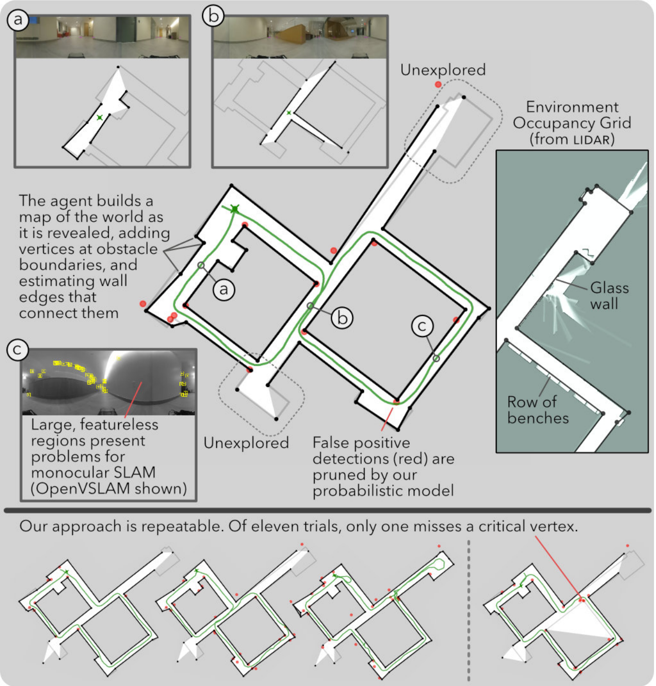

a
b
Unexplored
Environment Occupancy Grid (from LIDAR)
The agent builds a map of the world as it is revealed, adding vertices at obstacle boundaries, and estimating wall edges that connect them
a
b
c
Glass wall
Row of benches
c
Large, featureless regions present problems for monocular SLAM (OpenVSLAM shown)
Unexplored
False positive detections (red) are pruned by our probabilistic model
Our approach is repeatable. Of eleven trials, only one misses a critical vertex.

Fig. 1. **Map Building with Real Monocular Camera Data:** Here we show that our procedure builds a sparse map of an environment which is partially observed by a robot equipped with a monocular camera (green line is the robot path). A learned sensor is provided panoramic images (top of panels `(a)`–`(c)`) and returns a list of *vertices* which represent corners in the world, in addition to estimates of the *edges* that exist between them. By fusing observations between frames, we can reject detection errors (red circles) and extract the most likely map (solid black lines and circles). Known (shaded white regions) and unknown (shaded gray regions) space is additionally tracked, which can be used by a topological planner.

Planning with topological constraints necessitates a representation which is robust to the types of failures described:

*Equal contribution.

All authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, in Cambridge, USA. `{gjstein, cbrad, vpreston, nickroy}@mit.edu`

This work was supported by the Office of Naval Research and Marc Steinberg under the PERISCOPE MURI Contract #N00014-17-1-2699. Their support is gratefully acknowledged. V. Preston acknowledges support by a NDSEG Fellowship.

all obstacles must be added so that dead ends can be detected and planned around, while spurious obstacles (that may block paths or add unnecessary options) should be eliminated. Specifically for the exploration problem, even state-of-the-art work via monocular vision acknowledges the challenges (and failures) of extracting frontiers from their dense map reliable enough to enumerate the different ways the agent can enter unknown space during planning [8]. Relatedly, most planning strategies that leverage *homology* or *frontiers* rely on ideal sensors [5], [10] or specialized hardware, like a laser range finder, to build a map and generate plans [11], [12].

During exploration and navigation, planning with a map may be done via computation of a *visibility graph* [13], which is a minimal, optimal representation for planning in a known environment and can be extracted directly from a map. Maps generated using sparse features contain *clutter* or voids of features (e.g., in low-texture regions), which yield visibility graphs that are not useful for planning. Recent progress in deep learning for computer vision affords opportunities to overcome these faults comparatively easily by only adding sparse features where necessary for developing useful graphs for planning, without reliance on surface textures and by ignoring clutter or outlying detections.

The contribution of this paper is to extend recent work on planning with topological constraints to maps constructed with monocular vision, with an emphasis on planning through unknown environments. In this work, we present a novel map representation built from sparse detections from monocular vision, which is robust to noise in a way that enables navigation and exploration using topological constraints in both simulated and real-world environments. From our representation, we compute a visibility graph corresponding to the sparse, clutter-free structure of the known environment from which we can efficiently navigate using topological constraints. To build our map, we use a Convolutional Neural Network (CNN) to detect the edges of locally visible structure, which are fused to estimate regions of known and unknown space, and track obstacles that define topological constraints.

We first study *multi-agent search*; our method succeeds in 99% of trials across two simulated environments, demonstrating the robustness of our procedure (Sec. IV). We additionally show that our representation enables *visual learned subgoal planning*, demonstrating that our map is sufficiently stable to enable learning-based selection of topological actions for improved goal-directed navigation (Sec. V). Finally, we show that our work extends beyond simulation by building maps in three real-world environments (Sec. VI).

II. Planning with Topological Constraints

In this section, we examine requirements of topological planners and motivate our map representation. In order to efficiently navigate through both known and unknown environments, topological constraints can be employed to reduce the space of actions available to a robot. Two trajectories are said to be in different *homology classes* if the area they bound contains obstacles; planning approaches may use homology—a topological property—to constrain possible actions the agent can take (e.g., "go left around the building"). Storing a single point per contiguous obstacle is a minimally sufficient representation for computing the homology class of a trajectory [14]. Of interest in this paper is planning through unknown environments during map construction, for which *relative homology* [15] considers only partially revealed obstacles. *Frontiers* are boundaries between free and unknown space, and constrains planning to relative homology class. *Frontier-based planning* restricts trajectories to pass through the frontier of interest, thus trajectories are guaranteed to have unique relative homology. To use planners based on relative homology (or map frontiers), we need a representation that tracks both obstacles and free space.

The *Gap Navigation Tree* [16], [17] is a mapping technique for building a sparse, minimal, tree-structured graph of the world by detecting *gaps*, discontinuities in depth, and maintaining a list of unexplored branches for later exploration. *Gaps* store the high-level actions available to the robot, yet their practical utility is limited by sensor noise [18] and, due to its tree structure, Gap Navigation is limited to simply-connected environments. Moreover, gap locations are viewpoint-dependent, creating challenges for data association and robust map construction. To overcome these challenges, we approximate the environment as a polygon, so that the location of gaps become viewpoint independent, enabling reliable data association. As such, our sensor detects polygonal *vertices* in view of the agent; by localizing vertices and estimating the presence of walls or other impassible obstacles that connect them, our agent is capable of reconstructing the polygonal representation of its environment as it travels (see Sec. III).

Our map representation consists of the vertices and edges that define obstacles and an estimate of the free space observed by the robot. Planning within this representation is straightforward via computation of a visibility graph [13], which is minimal and optimal in a known polygonal environment. For exploration tasks, the representation of free space can be used for frontier-based planning by extracting frontiers from the difference between map edges and the boundary of free space. Additionally, we can easily compute the relative homology class of a trajectory from our representation by tracking partially revealed obstacles, making it sufficient for planning with topological constraints.

III. Probabilistic Map-building from Learned Monocular Sensing

Our agent is equipped with a learned sensor (Sec. III-A) that returns a list of detected vertices in view of the agent and, for each detection, the likelihood an obstacle exists between it and each of its neighbors (e.g., if the vertex and its neighbor are endpoints of a shared wall). We learn to identify where the robot can and cannot travel in order to explore its environment, and do this by detecting the boundaries of untraversable obstacles. Our map representation therefore stores three types of information: (a) **vertices**, each a cluster of vertex detections; (b) **edges**, connections between two

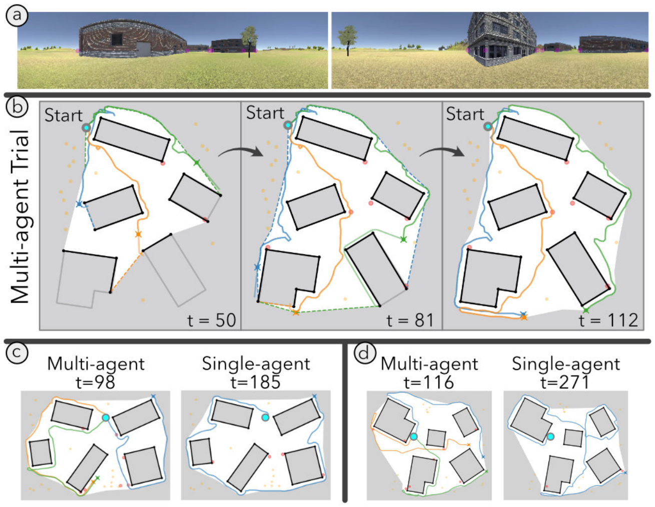


Fig. 2. **Search in Simulated Office Park:** Here we show three representative search scenarios in a simulated outdoor environment (a) populated with buildings and trees. In (b)–(d) the progress of multiple agents is shown, where the agents are constrained to pursue unique homology classes. We successfully identify and map all buildings in 99 of 100 simulated trials, illustrating the robustness of our technique. See Sec. IV for more details.

vertices which form an untraversable boundary; and a (c) **known space polygon**, a polygon defining what space has been observed to be free.

a) Vertices: Each vertex detection in an observation $z^{(i)}$ is compared to the vertices that exist in the map in order to determine how each detection should be associated. Either a detection is matched with an existing vertex, or it is added as a new vertex to the map with 2D pose and covariance. The Mahalanobis distance [18] is used to associate vertex detections between observations in order to cluster detections into vertices. We additionally apply a mutual exclusion constraint [19] such that no two vertex detections from a single observation are associated with the same vertex. Successful associations update the position and covariance of vertices using a Kalman filter.

b) Edges: Edges between vertices impose topological constraints necessary for planning. However, directly detecting the structural connectivity of an environment using vision is a challenging problem [20]. For each vertex detection our sensor instead returns a likelihood that an untraversable obstacle exists between each of its covisible neighbors (an "edge likelihood"). For an observation $z^{(i)}$, we average these likelihoods for each pair of detections to define the probability an edge exists between the vertices; over many observations we average these probabilities to determine whether or not an edge exists in the map.

c) Known Space Polygon: Each sensor observation $z^{(i)}$ reveals a *star-shaped* region surrounding the agent $s(z^{(i)})$. The function s constructs a polygonal region by sorting the vertex detections by the angle at which they were observed. The union of these polygons, $S = \bigcup_{i=0}^{N_{\text{obs}}} s(z^{(i)})$ for all sensor observations N_{obs} defines the space revealed by the agent.

When the sensor is perfect, the map representation can be built directly from the descriptions provided. In practice however, the sensor is noisy—both vertex and edge detections will not only be imperfect, but false positive and negative detections may also be introduced. In order to handle sensor noise, we define a probabilistic model from which we can assess the likelihood of a proposed map; the posterior likelihood of maps is discussed in Sec. III-C. To propose and evaluate the posterior over maps, we sample over the set of polygonal vertices and their most likely edges, then compute free space in the presence of noise. This discussion is included in Sec. III-B.

A. Vertex Detection with Convolutional Neural Networks

In order to detect vertices, we train a CNN to estimate the vertex and two edge likelihood terms. The network takes a 128×512 pixel RGB panoramic image and returns the 3 likelihoods over a 32×128 grid, where each point corresponds to a range-bearing coordinate from the sensor[1]. We use a fully-convolutional encoder-decoder network structure composed of blocks. Each encoder block consists of 2 convolution layers with 3×3 kernels, followed by a batch norm operation and a ReLU activation function, and terminated by a 2 × 2 max-pooling operation. Decoder blocks are similar, however the final convolutional layer has stride 2 (so that the output is upsampled) and the max-pooling operation is eliminated. The network consists of 5 encoder blocks, with output channels $[64, 64, 128, 128, 256]$, and 3 decoder blocks, with output channels $[128, 64, 64]$. The output of the final decoder layer is passed through a final 1×1 convolutional layer with 3 output channels, corresponding to the 3 likelihoods. A weighted cross-entropy with an empirically chosen positive-weight 8 is used as the loss function for the vertex likelihood, so that the sparse positive detections are not overwhelmed by the negative background. The edge likelihoods each have a sigmoid cross-entropy loss, yet are masked by the training vertex label so that loss is only non-zero where a vertex exists. The edge likelihood loss is weighted by a factor of 1/16 versus the vertex likelihood so that the network prioritizes whether a vertex exists before trying to estimate its properties. The Adam optimizer is used to train the network for 100k steps, with an initial learning rate of 2.5 and a learning rate decay of 0.5 every 10k steps.

B. Generating Proposal Maps

Associating detected vertices across many observations results in a set of *potential vertices*, of which only a subset may appear in the final map. By iteratively including and removing potential vertices, and the walls connected to them, we explore the space of possible maps via sampling[2]. Computing the map also requires computing the known space polygon and, as we change the vertices and edges included in the map, the known space must be updated. To construct this polygon for a given set of vertices and edges, we first compute a hypothetical observation z_h by ray casting against the proposed structure (z_h is what the robot would see if the proposed structure were accurate). However, this detection does not include unobserved obstacles. As such, we instead

[1] To recover a discrete set of vertices from the grid based output produced by the CNN, we use a peak-detection procedure with a threshold of 0.5.

[2] We randomly sample over vertices uniformly.

compute a *conservative estimate* of known space S_c when proposing maps: the intersection of the polygons computed from both the real and hypothetical sensor observation:

$$S_c = \bigcup_{i=0}^{N_{\text{obs}}} S_c^{(i)} = \bigcup_{i=0}^{N_{\text{obs}}} \left(s(z^{(i)}) \cap s(z_h^{(i)}) \right) \tag{1}$$

Without this conservative approach, the agent may incorrectly mark unobserved space as explored, leading to incomplete exploration.

We probabilistically accept proposed maps using an MCMC criteria and the map posterior. After a fixed number of samples (50 samples were used for all experiments presented in this work) we return the maximum likelihood map. In general, sampling requires iterating over the inclusion of walls in the proposed map. In practice, we use a heuristic to determine whether a wall exists: we compute an approximate marginal edge likelihood by computing individual observations of the edge likelihood using co-visible vertices in the *real* detections (rather than the *conservative* detections). We have found this heuristic sufficient for effective map-building and planning and considerably reduces the search space.

C. Defining the Posterior Over Maps

Given N observations $\{z^{(i)}\}_{1 \le i \le N}$, the posterior distribution over maps m can be expanded using Bayes Rule:

$$\log P(m|z^{(1)}, \cdots, z^{(N)}) = \\ \sum_{i=0}^{N} \log P(z^{(i)}|m) + \log P(m) - \log P(z^{(1)}, \cdots, z^{(N)}) \tag{2}$$

where we have assumed that the sensor observations are i.i.d. to factor $\log P(z^{(1)}, \cdots, z^{(N)}|m)$. The final term in Eq. (2), the likelihood of a sensor measurement, is a normalizer and can be ignored. The prior distribution over maps, $P(m)$, is intractably difficult to obtain in practice. However, we can instead incorporate heuristic priors about the map, namely biases against the addition of new vertices and walls, which mitigates the impact of false positive detections.

To evaluate the likelihood of each observation given a proposed map, we compare the sensor observation against the conservative space estimate for individual observations $S_c^{(i)}$. When evaluating the likelihood of a sensor observation $P(z^{(i)}|m)$, we reason about the likelihood of the vertices $P_v(z^{(i)}|m)$ and edges $P_e(z^{(i)}|m)$ independently. The contribution from the vertices depends only on the number of false positive and false negative detections: any real vertex detections that do not appear in $S_c^{(i)}$ are considered false positive detections and any vertices within $S_c^{(i)}$ that were not detected by the sensor are considered false negatives. Therefore, $P_v(z^{(i)}|m) = R_{\text{FP}}^{N_{\text{FP}}} R_{\text{FN}}^{N_{\text{FN}}}$, where $R_{\text{FP}}, R_{\text{FN}}$ are the rates of false positive/negative detections and $N_{\text{FP}}, N_{\text{FN}}$ are the number of observed false positive/negative detections.

The likelihood contribution from the edges $P_e(z^{(i)}|m)$ also makes use of the conservative space estimate $S_c^{(i)}$. For each vertex our sensor returns the likelihood it is connected via an obstacle to each of its covisible neighbors. The average of these defines the edges in our map. False negative detections, which do not have an associated real vertex detection, are assigned a uniform prior over edge likelihood.

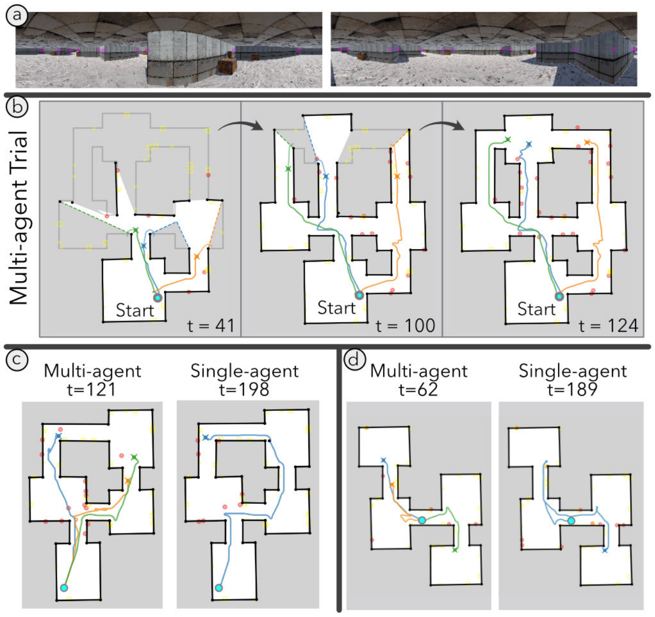


Fig. 3. **Search in Simulated Indoor Environment:** Here we show three representative search scenarios in a simulated indoor environment populated with random clutter (a). In (b) the progress of multiple agents is shown, where the agents are constrained to enter unknown space via a unique frontier. In panels (b)–(d), just as in the *Office Park* trials in Fig. 2, multi-agent teams complete exploration in roughly half the time (on average) that single agents use.

Our posterior codifies a number of behaviors we would expect to see during our map-building process. First, the posterior allows us to omit false positive vertices in a principled manner. Since false positive vertices can result in unwanted false negative observations as the robot travels, it ultimately makes it more likely that the vertex does not appear in the underlying geometry. Second, the existence of a wall depends on both the detected edge likelihood *and* its ability to occlude vertices. By occluding portions of the map, the edges also influence the number of false positive and false negative detections recorded during the evaluation of map likelihood.

IV. Multi-Agent Search

To demonstrate the reliability of our approach, we test our map-building procedure with *multi-agent search*, in which agents must be able to maintain an accurate estimate of observed space in order to reach an unseen goal. Multi-agent search strongly benefits from planning with topological constraints, so that each agent is encouraged to explore different regions of space and reveal the environment more quickly [1]. The challenges in detecting features in textureless regions and accurately resolving depth in cluttered regions make traditional monocular SLAM approaches impractical for this task [8], [9].

We conduct 200 simulated experiments across two environments we created in the Unity Game Engine [21]: (1) an *Indoor Environment*, a small labyrinth of rooms connected by corridors, and (2) an *Office Park*, an outdoor environment of randomly placed buildings and trees. Training data is gen-

TABLE I

PERFORMANCE METRICS (AVERAGED) FOR THE SEARCH TASK

	Indoor Environment			Office Park		
	Success	Steps	IoU	Success	Steps	IoU
Multi-agent	99/100	**137.6**	0.979	100/100	**128.0**	0.990
Single-agent	99/100	230.6	0.979	97/100	254.1	0.990

erated via simulated travel through a subset of environments not included in the test set. To plan with multiple agents, a joint planner encourages the agents to minimize the net cost of entering unknown space yet via different frontiers, thereby ensuring that agents select trajectories in different relative homology classes. Map construction with multiple agents is straightforward: observations are received in sequence from each of the three agents and added to the map in turn. Since the probabilistic model treats each measurement as i.i.d., no modifications are necessary to our map-building procedure.

We report performance in terms of coverage—the agent's map of the world should match the underlying map. As such, we use *Intersection over Union* (IoU) between the final reconstructed map and the true map as a measure of coverage[3]. Examples of our results are shown in Figs. 2 and 3, and a data table summarizing the results of all simulated trials—100 randomized maps for each simulated environment—are in Table I. Of our 200 trials, 198 succeed in completely exploring their environment, achieving an IoU above 0.95. The two remaining trials miss a single wall and the agents became stuck in place. This results in identical input images fed to our model, which will produce identical outputs. This violates the i.i.d. assumption of our sensor model and the maximum likelihood map may not match the underlying environment. This does not occur often and, despite this occasional limitation, our representation is sufficient for mapping and planning in unknown simulated environments with monocular vision, even in the presence of sensor noise. Data from the 200 multi-agent trials can be found in Table I; we show performance alongside an additional 200 trials from single-agent planning, reinforcing that imposing topological constraints with our map enables more efficient search of unknown environments.

V. VISUAL LEARNED SUBGOAL PLANNING

Our final simulated experiments demonstrate that our representation enables Learned Subgoal Planning (LSP) [6] without the need for specialized hardware. LSP involves estimating properties of contiguous boundaries between free and unknown space (frontiers), including the likelihood the goal can be reached through the boundary of interest, and uses these properties to approximate the expected cost of topological actions. In [6], the authors showed that their approach is a computationally tractable means of computing expected cost and demonstrated improvements for planning through different types of environment. Because the learning procedure relies upon stable, well-defined boundaries between free and unknown space, previous work on LSP relied on a planar laser scanner to build an occupancy grid map. We demonstrate here that our map representation from monocular vision is sufficiently reliable to enable LSP, thus showing that vision can be used for both enumerating the high-level actions available to the agent during exploration and deciding between these actions.

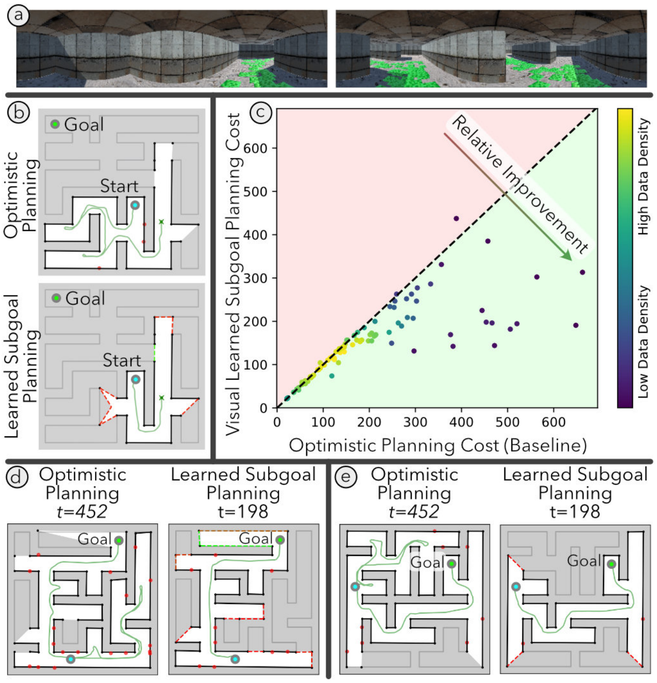
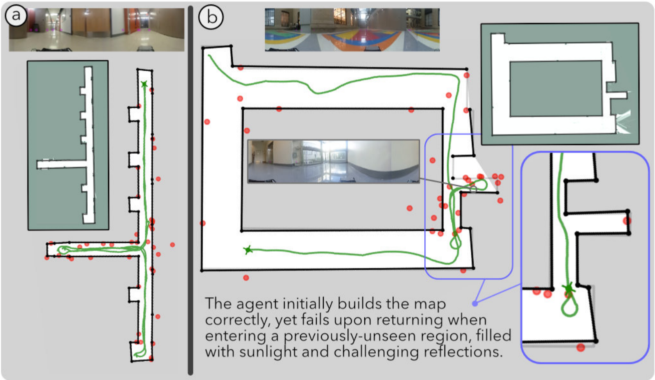


Fig. 4. **Optimistic and Learned Subgoal Planning in Simulated Guided Mazes:** Our map representation is used by an optimistic planner and a learned subgoal planner (LSP) that are compared in 200 simulated guided maze environments in which green "breadcrumbs" lead to the goal (a). In panel (b), snapshots of the two planners in progress are shown, wherein the optimistic planner is greedily searching nearby paths and the learned subgoal planner is pursuing frontiers marked with dashed lines (red are unlikely to reach the goal, green are more likely to reach the goal). Panels (d, e) show completed missions for each planner. As shown in panel (c), robots using the learned subgoal planner travel a total of 25.6% less distance than those using the optimistic planner. This result highlights that our method is not only suitable for navigation, it is stable enough to support LSP.

The LSP experiments require three frontier properties [6]—the likelihood a frontier will lead to the goal, the expected cost of reaching the goal (if the goal can be reached, and the expected cost of exploration (if the goal cannot be reached)—that we estimate alongside the vertex and edge likelihoods by adding output channels to our network architecture described in Sec. III-A. A few other changes to the neural network are made when estimating frontier properties. The frontiers can overlap in range-bearing space, so we add an additional decoder block to create a higher-resolution output. To ease initialization of this larger network, we found it more effective to use leaky-ReLU for the convolution layer activations. The goal location, needed to estimate frontier properties, is encoded into a 2-channel image and appended to the input of the third encoder block; each pixel corresponds to $[r_g \sin(\theta_g), r_g \cos(\theta_g)]$, where r_g is the relative range and θ_g is the relative angle of the goal versus the bearing of the pixel of interest. See [6] for a complete discussion of the

[3] We compute IoU differently in the outdoor environment, since we should not expect that the agent should mark all space around the buildings as free. Instead, we compute IoU with respect to the placement of the building obstacles. Mapping is perfect when all buildings are detected and in the correct locations and that no spurious buildings are added.



The agent initially builds the map correctly, yet fails upon returning when entering a previously-unseen region, filled with sunlight and challenging reflections.

Fig. 5. **Additional Real-World Mapping Trials:** Beyond the real-world result shown in Fig. 1, we show here mapping trials on real-world data in two different environments. In (a), the robot successfully built a map which accurately represents the underlying environment. In (b), despite a small alcove with challenging lighting for which no training data was collected, the learned sensor was able to accurately build much of this atrium environment.

frontier properties.

To evaluate performance, we created a visually-oriented variant of the *guided maze* environment from [6]. Each randomly-generated map is a minimum spanning tree maze, yet the path between the start and goal is marked by a green path on the ground. An agent using the learned subgoal planning algorithm should learn that the highlighted path indicates actions that are more likely to lead to the goal and preferentially pursue those. Vision therefore provides a strong signal for the correct path.

We conducted 400 simulated trials in our *guided maze* environments: 200 in which the agent builds the map and uses the LSP algorithm to select the lowest expected cost action, and 200 using a naive baseline that plans as if all unknown space is free. All 400 trials reach the unseen goal, speaking to the reliability of our map-building and navigation procedure. Results showing a side-by-side comparison of the LSP procedure versus the naive baseline are shown in Fig. 4. As expected, the learned planner matches or outperforms the baseline planner in nearly all trials (25.6% reduction in net cost), highlighting that our representation is not only able to support navigation, but also is stable enough to enable learning the properties necessary for LSP.

VI. Mapping with Real Monocular Camera Data

To demonstrate our learned sensor and mapping procedure on real data, a teleoperated robot equipped with a Ricoh Theta S [22] panoramic camera and Hokuyo LIDAR [23] (used to generate training data) was driven in three distinct indoor environments on the MIT campus. We drove the robot through our target environments multiple times (fewer than 10 traversals for each environment; on the order of 10k images) and trained a sensor for each environment. We generated ground truth occupancy grids with Cartographer [24] and hand-built the polygonal map of each environment with clutter removed to pair with localized images for training. The training and testing sets were mutually exclusive, though the generalization performance of our planner to unseen environments is the subject of future work.

a) Bld. 36-2, Fig. 5 (a): Our first environment was a simply-connected hallway intersection. Despite a high number of false-positive detections (red circles), we correctly built the underlying map, capturing the high-level structure and building a single contiguous boundary.

b) Bld. 6C Atrium, Fig. 5 (b): This environment consisted of one large loop and a small annex with large windows and glass walls. Although we were able to correctly map the central loop, we failed to accurately represent the annex, where light glare saturated the images. Notably, during the trial, the robot at one point had an accurate representation of the annex (highlighted in Fig. 5 (b)), yet the challenges in detection overwhelmed and resulted in many false positive detections (red circles) in this region.

c) Bld. E52-2, Fig. 1: This environment consisted of two loops and several branching hallways populated with clutter (benches, chairs) and challenging surfaces (glass and textureless walls). In these trials we show that we are able to consistently build a map which, qualitatively, well represents the underlying structure of the environment. Quantitatively, 10 of 11 trials built the two loops in the environment completely and accurately. In the unsuccessful trial, there was a failure of data association in the rightmost loop; where there should be 1 vertex, there were 3 rejected vertices (red circles). As compared to a state-of-the-art monocular SLAM package [25] for panoramic input, highlighted in Fig. 1, our learned sensor performed robustly in predominantly textureless regions.

VII. Conclusion

In this work, we present a sparse map that enables planning with topological constraints using monocular vision. We introduce a learned sensor to detect vertices from panoramic images and estimate the presence of large obstacles that connect them in order to extract a polygonal representation of the environment that includes only high-level structure. Our map also includes an estimate of known free space, which we use to define topological constraints—e.g., homology and frontier constraints—for planning through unknown environments. Upon completion, our procedure yields a polygonal map which is sufficient for optimal navigation via computation of a visibility graph. Our results motivate future work that generalizes to novel environments, extends to actions beyond the domain of navigation, and incorporates localization within our representation.

We demonstrate the utility and robustness of our representation in both *multi-agent search* and *visual learned subgoal planning*. In simulated trials, we show that our representation can be used to efficiently search unknown space. Moreover, we demonstrate that our map is sufficiently stable to enable visual learned subgoal planning. Through trials in several representative indoor environments, we further show that our map can be generated from real, noisy visual data under conditions in which other methods relying on sparse visual features have been shown to fail.

References

[1] V. Govindarajan, S. Bhattacharya, and V. Kumar, “Human-robot collaborative topological exploration for search and rescue applications,” in *Distributed Autonomous Robotic Systems*, 2016, pp. 17–32.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[3] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, “Topological exploration of unknown and partially known environments,” in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 3851–3858.

[4] J. Park, S. Karumanchi, and K. Iagnemma, “Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015.

[5] R. Korb and A. Schöttl, “Exploring unstructured environment with frontier trees,” *Journal of Intelligent and Robotic Systems*, vol. 91, no. 3, pp. 617–628, Sep. 2018.

[6] G. J. Stein, C. Bradley, and N. Roy, “Learning over subgoals for efficient navigation of structured, unknown environments,” in *Conference on Robot Learning*, 2018, pp. 213–222.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] L. von Stumberg, V. Usenko, J. Engel, J. Stückler, and D. Cremers, “From monocular SLAM to autonomous drone exploration,” in *IEEE European Conference on Mobile Robots*. IEEE, 2017, pp. 1–8.

[9] C. Mostegel, A. Wendel, and H. Bischof, “Active monocular localization: Towards autonomous monocular exploration for multirotor MAVs,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 3848–3855.

[10] B. Tovar, S. M. La Valle, and R. Murrieta, “Optimal navigation and object finding without geometric maps or localization,” in *IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 2003, pp. 464–470.

[11] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Computational Intelligence in Robotics and Automation*, 1997, pp. 146–151.

[12] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. How, “Search and rescue under the forest canopy using multiple UAS,” in *International Symposium on Experimental Robotics*, 2018.

[13] T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[14] S. Bhattacharya, M. Likhachev, and V. Kumar, “Topological constraints in search-based robot path planning,” *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[15] F. T. Pokorny, K. Goldberg, and D. Kragic, “Topological trajectory clustering with relative persistent homology,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 16–23.

[16] B. Tovar, R. Murrieta-Cid, and S. M. LaValle, “Distance-optimal navigation in an unknown environment without sensing distances,” *IEEE Transactions on Robotics*, vol. 23, pp. 506–518, 2007.

[17] B. Tovar, L. Guilamo, and S. M. LaValle, “Gap navigation trees: Minimal representation for visibility-based tasks,” in *Algorithmic Foundations of Robotics VI*. Springer, 2005, pp. 425–440.

[18] L. Murphy and P. Newman, “Using incomplete online metric maps for topological exploration with the gap navigation tree,” in *International Conference on Robotics and Automation*, 2008, pp. 2792–2797.

[19] M. Montemerlo and S. Thrun, “Simultaneous localization and mapping with unknown data association using FastSLAM,” in *IEEE International Conference on Robotics and Automation*, vol. 2, Sep. 2003, pp. 1985–1991 vol.2.

[20] W. N. Greene and N. Roy, “Flame: Fast lightweight mesh estimation using variational smoothing on Delaunay graphs,” in *International Conference on Computer Vision*, 2017, pp. 4696–4704.

[21] Unity Technologies, “Unity game engine,” https://unity3d.com, 2019.

[22] Ricoh Theta S, “Ricoh Theta S Panoramic Camera,” https://theta360.com/en/about/theta/s.html, 2019.

[23] Hokuyo, “Hokuyo UTM-30LX Scanning Laser Rangefinder,” https://www.hokuyo-aut.jp/search/single.php?serial=169, 2019.

[24] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D LIDAR SLAM,” in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 1271–1278.

[25] S. Sumikura, M. Shibuya, and K. Sakurada, “OpenVSLAM: A versatile visual SLAM framework,” in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.