Planning Beyond The Sensing Horizon Using a Learned Context

Michael Everett and Jonathan P. How

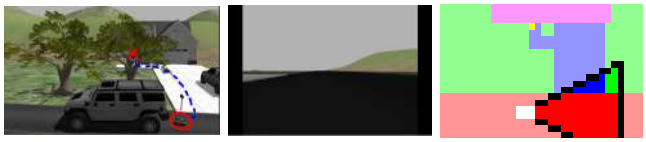
***Abstract*— Last-mile delivery systems commonly propose the use of autonomous robotic vehicles to increase scalability and efficiency. The economic inefficiency of collecting accurate prior maps for navigation motivates the use of planning algorithms that operate in unmapped environments. However, these algorithms typically waste time exploring regions that are unlikely to contain the delivery destination. Context is key information about structured environments that could guide exploration toward the unknown goal location, but the abstract idea is difficult to quantify for use in a planning algorithm. Some approaches specifically consider contextual relationships between objects, but would perform poorly in object-sparse environments like outdoors. Recent deep learning-based approaches consider context too generally, making them difficult to train. Therefore, this work proposes a novel formulation of utilizing context for planning as an image-to-image translation problem, which is shown to extract terrain context from semantic gridmaps, into a metric that an exploration-based planner can use. The proposed framework has the benefit of training on a static dataset instead of requiring a time-consuming simulator. We demonstrate our trained algorithm in simulated environments that have similar structure to real last-mile delivery domains. The robot reaches its goal 63% faster than with a context-unaware planner.**

I. Introduction

A key topic in robotics is the use of automated robotic vehicles for first- and last-mile delivery. A standard approach is to visit and map delivery environments ahead of time, which enables the use of planning algorithms that guide the robot toward a specific goal coordinate in the map. However, the economic inefficiency of collecting and maintaining maps, the privacy concerns of storing maps of people's houses, and the challenges of scalability across a city-wide delivery system are each important drawbacks of the pre-mapping approach. This motivates the use of a planning framework that does not need a prior map. In order to be a viable alternative framework, the time required for the robot to locate and reach its destination must remain close to that of a prior-map-based approach.

Consider a robot delivering a package to a new house's front door (Fig. 1). Many existing approaches require delivery destinations to be specified in a format useful to the robot (e.g. position coordinates, heading/range estimates, target image), but collecting this data for every destination presents the same limitations as prior mapping in general. Therefore the destination should be a high-level concept, like "go to the front door." Such a destination is intuitive for a human, but without actual coordinates, difficult to translate into a planning objective for a robot. The destination will often be beyond the robot's economically-viable sensors' limited range and field of view. Therefore, the robot must explore [1], [2] to find the destination; however, it is well-known that pure exploration is slow because time is spent exploring areas unlikely to contain the goal. Therefore, this paper investigates the problem of efficiently planning beyond the robot's line-of-sight by utilizing *context* within the local vicinity.

Aerospace Controls Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA, USA. {mfe, jhow}@mit.edu



(a) Oracle's view (b) Robot's view (c) Semantic Map

Fig. 1: Robot delivers package to front door. If the robot has no prior map and does not know where the door is, it must quickly search for its destination. Context from the onboard camera view (b) can be extracted into a lower-dimensional semantic map (c), where the white robot can see terrain within its black FOV.

Many existing algorithms focus on a single type of context. Exploration algorithms typically find frontiers in occupancy gridmaps as a geometric form of context to guide planning [1], [2]. Object search papers leverage semantic object classification to provide context in the form of relationships between objects, rooms, or places [3]–[11]. Such representations of context are too specific for environments where the relevant clues come in a variety of formats. On the other hand, recent approaches [12]–[19] that use deep learning to directly map from the current onboard camera image to the optimal action have the opposite problem: they are too general. Learning to extract *and* utilize context in one process requires massive amounts of task-specific training data (or a photo-realistic simulator), and significant computation time/resources for training due to the learning task's high-dimensionality.

This work proposes a solution to efficiently utilize context for planning. Scene context is represented in a semantic map, then a learning algorithm converts context into a search heuristic that directs a planner toward promising regions in the map. Semantic gridmaps are sufficiently descriptive to represent both geometric and semantic context clues, yet more compact than a time-history of camera images (millions of pixels), allowing for efficient, context-based learning. The context utilization problem (determination of promising regions to visit) is uniquely formulated as an image-to-image translation task, and solved by a Conditional Generative Adversarial Network (cGAN) [20] that was previously shown to be useful for geometric context extraction [21]. By learning with semantic gridmap inputs instead of camera images, the planner proposed in this work could be more easily transferred to the real world without the need for training in a photo-realistic simulator. Moreover, a standard local collision avoidance algorithm can operate in conjunction with this work's global planning algorithm, making the framework easily extendable to environments with dynamic obstacles.

The contributions of this work are i) a novel formulation of utilizing context for planning as an image-to-image translation problem, which is shown to convert the abstract idea of scene context into a metric that a planner can use, ii) an algorithm

to efficiently train a cost-to-go estimator on partial semantic maps of typical environment layouts, which enables a robot to learn from a static dataset instead of a time-consuming simulator, and iii) demonstration of a robot reaching its goal 63% faster than a context-unaware algorithm in simulated environments based on real last-mile delivery domains.

II. Related Work

1) Planning & Exploration: Classical planning algorithms rely on knowledge of the goal coordinates (A*, RRT) and/or a prior map (PRMs, potential fields), which are both unavailable in this problem. Receding-horizon algorithms are inefficient without an accurate heuristic at the horizon, typically computed with goal knowledge. Rather than planning to a destination, the related field of exploration [1], [2] is a conservative search strategy, and pure exploration algorithms often estimate *information gain* of planning options using geometric context. However, exploration and search objectives differ, meaning the exploration robot will spend time gaining information in places that are useless for the search task.

2) Context for Object Search: Leveraging scene context is therefore fundamental to enable object search that outperforms pure exploration. Many papers consider a single form of context. Geometric context (represented in occupancy grids) is used in [22]–[25], but these works also assume knowledge of the goal location for planning. Works that address true object search usually consider semantic object relationships as a form of context instead. Decision trees and maximum entropy models can be trained on object-based relationships, like positions and attributes of common items in grocery stores [3]. Because object-based methods require substantial domain-specific background knowledge, some approaches automate the background data collection process by using Internet searches [4], [5]. Object-based approaches have also noted that the spatial relationships between objects are particularly beneficial for search (e.g. keyboards are often *on* desks) [6]–[8], but are not well-suited to represent geometries like floorplans/terrains. Hierarchical planners improve search performance via a human-like ability to make assumptions about object layouts [10], [11]. To summarize, existing uses of context either focus on relationships between objects or the environment's geometry. These approaches are too specific to represent the combination of various forms context that are often needed to plan efficiently.

3) Deep Learning for Object Search: Recent works use deep learning to represent scene context. Several approaches consider navigation toward a semantic concept (e.g. go to the kitchen) using only a forward-facing camera. These algorithms are usually trained end-to-end (image-to-action) by supervised learning of expert trajectories [12]–[14] or reinforcement learning in simulators [15]–[19]. Training such a general input-output relationship is challenging; therefore, some works divide the learning architecture into a deep neural network for each sub-task (e.g. mapping, control, planning) [14], [17], [18].

Still, the format of context in existing, deep learning-based approaches is too general. The difficulty in learning how to extract, represent, and use context in a generic architecture leads to massive computational resource and time requirements for training. In this work, we reduce the dimensionality (and therefore training time) of the learning problem by first leveraging existing algorithms (semantic SLAM, image segmentation) to extract and represent context from images; thus, the learning process is solely focused on context utilization. A second limitation of systems trained on simulated camera images, such as existing deep learning-based approaches, is a lack of transferability to the real world. Therefore, instead of learning from simulated camera images, this work's learned systems operate on semantic gridmaps which could look identical in the real world or simulation.

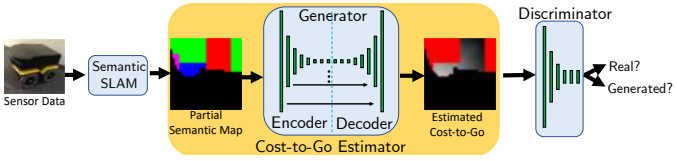


Fig. 2: GAN for semantic-to-cost image translation. During training, generator takes in partial semantic maps (left) and produces cost-to-go maps (middle) that are supposed to look like the analytical cost-to-go, in order to trick the discriminator that is trying to determine if an image is real or fake. After training, the generator alone can be used to translate from semantic maps to cost-to-go estimates.

4) Reinforcement Learning: Reinforcement learning (RL) is a commonly proposed approach for this type of problem [15]–[19], in which experiences are collected in a simulated environment. However, in this work, the agent's actions do not affect the static environment, and the agent's observations (partial semantic maps) are easy to compute, given a map layout and the robot's position history. RL is useful when there is a complicated mapping between observations, actions, and rewards, but is very time-intensive to train, and is not necessary for this problem.

5) GANs for Context Extraction: This work's use of GANs [20], [26], [27] is motivated by experiments that show GANs can imagine unobserved regions of occupancy gridmaps, suggesting that they can be trained to extract significant geometric context in structured environments [21]. However, the focus of that work is on models' abilities to encode context, not context utilization for planning.

III. Approach

The input to this work's architecture is a partial semantic gridmap (from an existing semantic SLAM system), as shown in Fig. 2. An image-to-image translation model is trained to estimate the planning cost-to-go given the semantic gridmap. Then, the estimated cost-to-go is used to inform a frontier-based exploration planning framework.

A. Training Data

A typical criticism of learning-based systems is the challenge and cost of acquiring useful training data. This work has two key features for data generation: recycling of existing, related data by using a common representation format, and an automated pipeline to produce (input, output) training pairs from the common context representation. These features enable learning of context extraction without the need of training in a time-consuming environment simulator.

Fortunately, domain-specific data already exists for many exploration environments (e.g., satellite images of houses, building floor plans for indoor tasks); however, a robot with limited sensing would likely use a SLAM algorithm in a new environment to localize interesting context clues. To

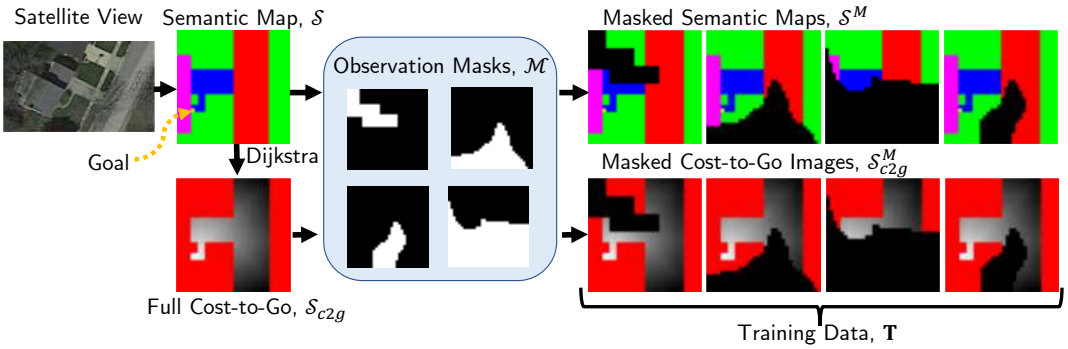


Fig. 3: Training data creation. A satellite view of a house's front yard is manually converted to a semantic map (top left), with colors for different objects/terrain. Dijkstra's algorithm gives an analytical solution to distance from goal to every point along drivable terrain (bottom left) [28]. This cost-to-go is represented in grayscale (lighter near the goal), with red pixels assigned to untraversable regions. To simulate partial map observability, dozens of observation masks (center) are applied to the full images to produce the training set.

accommodate the various typical raw data formats, this work uses semantic gridmaps as a common representation format. A set of 41 (25 train, 16 test) semantic maps was manually created using typical suburban house layouts, but recent breakthroughs in image segmentation algorithms [29] suggest semantic map creation could be automated from, for example, a set of satellite images. Each map yields $\sim$ 200 training pairs, due to the various masks applied (explained below), reducing the number of semantic maps that are needed. An example semantic map of a suburban front yard is shown in the top left corner of Fig. 3: red shows the road, blue is the driveway, magenta is the house, green is grass, and the small yellow square is the delivery destination (front door).

Algorithm 1 describes the process of automatically generating training data from a set of semantic maps, $\mathbf{S}$. A semantic map, $\mathcal{S} \in \mathbf{S}$, is first separated into traversable (roads, driveways) and non-traversable (grass, house) regions, represented as a binary array, $\mathcal{S}_{tr}$ (Line 4). Then, the shortest path length between each traversable point in $\mathcal{S}_{tr}$ and the goal is computed with Dijkstra's algorithm [28] (Line 5). The result, $\mathcal{S}_{c2g}$, is stored in grayscale (Fig. 3 bottom left: darker is further from goal). Non-traversable regions are red in $\mathcal{S}_{c2g}$.

This work assumes the robot starts with no knowledge of a particular environment's semantic map; it observes (uncovers) new areas of the map as it explores the environment. To approximate the partial maps that the robot will have at each planning step, full training maps undergo various masks to occlude certain regions. For some binary observation mask, $\mathcal{M}$: the full map, $\mathcal{S}$, and full cost-to-go, $\mathcal{S}_{c2g}$, are masked with element-wise multiplication as $\mathcal{S}^M = \mathcal{S} \circ \mathcal{M}$ and $\mathcal{S}^M_{c2g} = \mathcal{S}_{c2g} \circ \mathcal{M}$ (Lines 7 and 8). The (input, target output) pairs used to train the image-to-image translator are the set of $(\mathcal{S}^M, \mathcal{S}^M_{c2g})$ (Line 9).

Algorithm 1: Automated creation of GAN training data

1 **Input:** set of semantic maps $\mathbf{S}$
2 **Output:** set of training image pairs $\mathbf{T}$
3 **foreach** $\mathcal{S} \in \mathbf{S}$ **do**
4 $\mathcal{S}_{tr} \leftarrow$ Find traversable regions in $\mathcal{S}$
5 $\mathcal{S}_{c2g} \leftarrow$ Compute cost-to-go to goal of all pts in $\mathcal{S}_{tr}$
6 **foreach** $\mathcal{M} \in \mathbf{M}$ **do**
7 $\mathcal{S}^M_{c2g} \leftarrow$ Apply observation mask $\mathcal{M}$ to $\mathcal{S}_{c2g}$
8 $\mathcal{S}^M \leftarrow$ Apply observation mask $\mathcal{M}$ to $\mathcal{S}$
9 $\mathbf{T} \leftarrow \{(S^M, S^M_{c2g})\} \cup \mathbf{T}$

B. Image-to-Image Translation

The motivation for using image-to-image translation is that a) a robot's sensor data history can be compressed into an image (semantic gridmap), and b) an estimate of cost-to-go at every point in the map (thus, an image) enables efficient use of receding-horizon planning algorithms, given only a high-level goal ("front door"). Although the task of learning to predict just the goal location is easier, a cost-to-go estimate implicitly estimates the goal location and then provides substantially more information about how to best reach it. A partial map could be associated with multiple plausible cost-to-gos (depending on the unobserved parts of the map), so the generator predicts the most likely cost-to-go map, with respect to the common structures seen in training. Learning about common structures is a key insight that informs goal-directed exploration better than geometric measures of information gain (e.g. [2], [30]).

The image-to-image translator used in this work is based on [20], implemented in TensorFlow by [27]. The translator (generator) is a standard encoder-decoder network with skip connections between certain encoder and decoder layers ("U-Net") [31]. The objective is to supply a 256x256 RGB image[1] (semantic map, $\mathcal{S}$) as input to the encoder, and for the final layer of the decoder to output a 256x256 RGB image (estimated cost-to-go map, $\hat{\mathcal{S}}_{c2g}$).

Standard strategies for training the translator differ in their loss functions: supervised learning via L_2 often leads to blurry outputs, whereas L_1 produces low-frequency outputs [20]. GAN architectures, on the other hand, train a discriminator to identify whether a given image came from a dataset or from the generator; the generator is simultaneously trained to generate images that the discriminator cannot distinguish from the dataset's images. As discovered in [20], the GAN architecture enforces that high-frequency signals (sharp edges) exist in the generated images, otherwise a simple frequency-based discriminator would never be tricked. The target images in this work's domain are extremely smooth (low frequency) by definition: neighboring pixels have similar

[1]Modification of network to accept the domain's 32×32 semantic maps performed worse than scaling the grids to 256×256, possibly because fewer convolutional layers could be used in the encoder-decoder network.

Algorithm 2: DC2G (Deep Cost-to-Go) Planner

1 **Input:** current partial semantic map $\mathcal{S}$, pose (p_x, p_y, θ)
2 **Output:** action $\mathbf{u}_t$
3 $\mathcal{S}_{tr} \leftarrow$ Find traversable cells in $\mathcal{S}$
4 $\mathcal{S}_r \leftarrow$ Find reachable cells in $\mathcal{S}_{tr}$ from (p_x, p_y) w/ BFS
5 **if** *goal* $\notin \mathcal{S}_r$ **then**
6 $\quad \mathcal{F}_r \leftarrow$ Find reachable frontier cells in $\mathcal{S}_r$
7 $\quad \hat{\mathcal{S}}_{c2g} \leftarrow$ Query generator network with input $\mathcal{S}$
8 $\quad \mathcal{C} \leftarrow$ Filter and down-sample $\hat{\mathcal{S}}_{c2g}$
9 $\quad (f_x, f_y) \leftarrow$ Find cell in $\mathcal{F}_r$ with max value $\mathcal{C}$
10 $\quad \mathbf{u}_{t:\infty} \leftarrow$ Backtrack from (f_x, f_y) to (p_x, p_y) w/ BFS
11 **else**
12 $\quad \mathbf{u}_{t:\infty} \leftarrow$ Shortest path to goal via BFS

grayscale intensities because, from the planner's perspective, they are only 1 step apart in Euclidean distance. Although the translation results shown in Section IV use a combination of GAN and L_1 loss as recommended in [20], a simple L_1 loss was also able to learn this relatively low-frequency image translation task.

C. Environment

The custom simulation environment is based on a 32×32-cell gridworld [32] to approximate a real robotic vehicle operating in a delivery context. Each grid cell is assigned a static class (house, driveway, etc.); this terrain information is useful both as context to find the destination, but also to enforce the real-world constraint that robots should not drive across houses' front lawns. The agent can read the type of any grid cell within its sensor FOV (to approximate a common RGB-D sensor: 120° horizontal, 8-cell radial range), which would be done with a terrain classifier on a real robot (e.g., [33]). To approximate a SLAM system, the agent remembers all cells it has seen since the beginning of the episode. At each step, the agent sends an observation to the planner containing an image of the agent's semantic map knowledge, and the agent's position and heading. The planner selects one of three actions: go forward, or turn $\pm 90°$.

We chose to evaluate in a gridworld because it enables the thorough analysis provided in Section IV, yet is sufficiently complex to demonstrate the fundamental limitations of a baseline algorithm. Moreover, the environment and proposed planning algorithm could be extended to 3D (as in [34]). To scale to larger domains, a multi-scale approach might be necessary (as in [18]). Extension to a real-world robot will involve integration between the proposed algorithm and state-of-the-art perception and SLAM software [33], [35].

D. Planner

This work's planner is based on the idea of frontier exploration [1], where a frontier is defined as a cell in the map that is observed and traversable, but whose neighbor has not yet been observed. Given a set of frontier cells, the key challenge is in choosing *which* frontier cell to explore next. Existing algorithms often use geometry (e.g., frontier proximity, expected information gain based on frontier size/layout); we instead use context to select frontier cells that are expected to lead toward the destination.

The planning algorithm, called Deep Cost-to-Go (DC2G), is described in Algorithm 2. Given the current partial semantic map, $\mathcal{S}$, the subset of observed cells that are also traversable (road/driveway) is $\mathcal{S}_{tr}$ (Line 3). The subset of cells in $\mathcal{S}_{tr}$ that are also reachable, meaning a path exists from the current position, through observed, traversable cells, is $\mathcal{S}_r$ (Line 4).

The planner opts to explore if the goal cell is not yet reachable (Line 6). The current partial semantic map, $\mathcal{S}$, is passed into the image-to-image translator, which produces a 256x256 RGB image of the estimated cost-to-go, $\hat{\mathcal{S}}_{c2g}$ (Line 8). The raw output from the decoder network is converted to HSV-space and pixels with high value (not grayscale ⇒ estimated not traversable) are filtered out. The remaining grayscale image is scaled down to match the gridmap's 32×32 size with a nearest-neighbor interpolation. The value of every grid cell in the map is assigned to be the saturation of that pixel in the translated image (high saturation ⇒ "whiter" in grayscale ⇒ closer to goal).

To enforce exploration, only frontier cells, $\mathcal{F}_r$, (reachable, traversable, and with an unobserved neighbor) are considered as possible subgoals. The cell in $\mathcal{F}_r$ with highest estimated value is selected as the subgoal (Line 9). Since the graph of reachable cells was already searched, the shortest path from the selected frontier cell to the current cell is available by backtracking through the search tree (Line 10). This backtracking procedure produces the list of actions, $\mathbf{u}_{t:\infty}$ that leads to the selected frontier cell. The first action, $\mathbf{u}_t$ is implemented and the agent takes a step, updates its map with new sensor data, and the sense-plan-act cycle repeats. If the goal is deemed reachable, exploration halts and the shortest path to the goal is implemented (Line 12). However, if the goal has been observed, but a traversable path to it does not yet exist in the map, exploration continues in the hope of finding a path to the goal.

The DC2G algorithm as described above requires knowledge of the environment's dimensions (i.e., 32×32), in order to produce partial semantic maps with the uncovered regions in the proper map location. This information might not be available on a real robot; therefore we evaluated a variant of our algorithm, DC2G-Rescale, in which the largest square containing observed cells is up-scaled to the full map size (32×32), and this up-scaled map is used as the $\mathcal{S}$ input to the generator. Because this variant lacks some information in the vanilla DC2G algorithm, performance is expected to be worse. However, in practice, a maximum search area would likely already be defined relative to the robot's starting position (e.g., to ensure operation time less than battery life). This search boundary could be used as prior knowledge for the planner, enabling the use of vanilla DC2G.

A key benefit of the DC2G planning algorithm is that it can be used alongside a local collision avoidance algorithm, which is critical for domains with dynamic obstacles (e.g., pedestrians on sidewalks). This flexibility contrasts end-to-end learning approaches where collision avoidance either must be part of the objective during learning (further increasing training complexity) or must be somehow combined with the policy in a way that differs from the trained policy.

Moreover, a benefit of map creation during exploration is the possibility for the algorithm to confidently "give up" if it fully explored the environment without finding the goal.

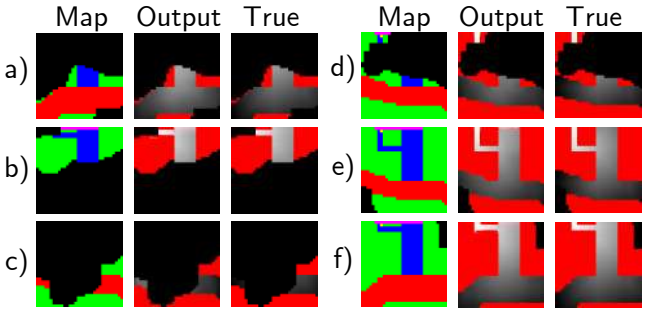


Fig. 4: Trained Generator vs. Truth. Generator output on 6 unseen partial semantic maps is quite similar to ground truth. Outputs correctly assign red to untraversable regions, black to unseen regions, and grayscale with intensity corresponding to distance from goal. Some undesirable features exist, such as blurring near goal in f). These generated outputs enable planning without knowledge of the goal's location.

This idea would be difficult to implement as part of a non-mapping exploration algorithm, and could address ill-posed exploration problems that exist in real environments (e.g., if no path exists to destination).

IV. Results

A. Image-to-Image Translation

Training the cost-to-go estimator took about 2 hours on a GTX 1060, though this is highly dependent on the number of training images (6,400). Generated outputs resemble the analytical cost-to-gos within a few minutes of training, but images look sharper/more accurate as training time continues.

Fig. 4 shows the generator's output on 6 partial semantic maps that were made with worlds and observation masks that were not seen during training. The similarity to the ground truth values qualitatively suggests that the generator successfully learned that: non-traversable regions should be colored red, unseen areas should be colored black, and cells in traversable regions should be grayscale with saturation corresponding to goal proximity. Some undesirable features exist, like false lightness close to edges. In Fig. 4 c), only road and grass cells have been observed (very little context); the output is accordingly inaccurate, with the road on the middle-left colored much lighter in the output versus the true target.

In general, measuring the performance of image-to-image translators is difficult [36]. Common approaches use humans or pass the output into a segmentation algorithm that was trained on real images [36]; but, the first approach is not scalable and the second does not apply here. Unique to this paper's domain, for fully-observed maps, ground truth and generated images can be directly compared, since there is an analytical solution to each pixel's target intensity. However, it is unclear how to compare penalties on undesirable features: is a too-light pixel worse than a too-dark one? Furthermore, partially-observed maps lack a ground truth image because there could be many different map layouts that lead to the same partial map. An alternative evaluation metric is the performance of a planning algorithm that uses a particular generator since this more directly answers the paper's main question: how to extract context to improve planning?

It turns out the generators trained with each loss function (L_1 and GAN+L_1) produce similar planning performance, suggesting the discriminator architecture is not necessarily needed for these low-dimensional environments.

B. Planner Scenario

Next, we tested the proposed DC2G algorithm against Frontier, a standard algorithm that always plans to the nearest frontier cell (pure exploration) [1]. Both algorithms start in the same position in a world that was not seen during training; several steps are shown in Fig. 5. The top row shows the partial semantic map at that timestep (observation), the middle row is the generator's cost-to-go estimate, and the bottom shows the agent's trajectory in the whole, unobservable map. Both algorithms begin with little context since most of the map is unobserved. By step 20, DC2G (green box on left) has found the intersection between road (red) and driveway (blue): this context causes it to turn up the driveway. By step 51, DC2G has observed the goal cell, so it simply plans the shortest path to it with BFS, finishing in 59 steps.

Conversely, Frontier (pink box on right) takes much longer (139 steps) to reach the goal, because it does not consider terrain context. Although the two algorithms have similar observations at one point (step 20 of DC2G, step 43 of Frontier), Frontier turns away from the driveway and continues exploring road regions.

The use of exploration is critical to maintain robustness to local minima in the cost-to-go estimate. A simpler planning algorithm, in which a set of motion primitives within the current FOV are ranked by the lowest estimated cost-to-go, was observed to often become stuck at local minima. Local minima in the cost-to-go estimates arise from many sources, including incomplete context (e.g., only have seen one type of terrain) and noise in the network approximation. Rather than try to balance exploration and exploitation by, say, exploiting until reaching a local minimum, then exploring for some time period, the DC2G planner explores constantly.

C. Planner Performance

30 trials are run in each test set. The first test set has 6 world layouts: each differs from the training set but maintain many common structures. In each trial, a world layout and a random initial position are selected, and each algorithm is tested on these same conditions. The optimal path is computed by an oracle with full knowledge of the map ahead of time; each algorithm's performance is therefore presented as *extra* time to the goal, $t^e_{goal} = t^{alg}_{goal} - t^{oracle}_{goal} \geq 0$, beyond the oracle's path.

The results of each algorithm are aggregated in Fig. 6, with first (most similar to training) test set as the leftmost group. Vanilla DC2G (green, left) on average takes only 7.8 ± 7.8 extra steps, whereas the pure frontier exploration (red, right) takes 75.9 ± 47.2 extra steps than the oracle. DC2G-Rescale (dark green, middle), the DC2G variant that lacks knowledge of the partial map's location within the true semantic map, takes 17.1 ± 15.7; it performs slightly worse than DC2G but still much better than the frontier-based planner. To put the number of steps in perspective, the oracle averaged a total path length of 40.0 ± 13.3 steps. Therefore, DC2G's trajectories were 63% shorter than Frontier's.

This result demonstrates a substantial increase in efficiency by using the DC2G planner's context-guided exploration in

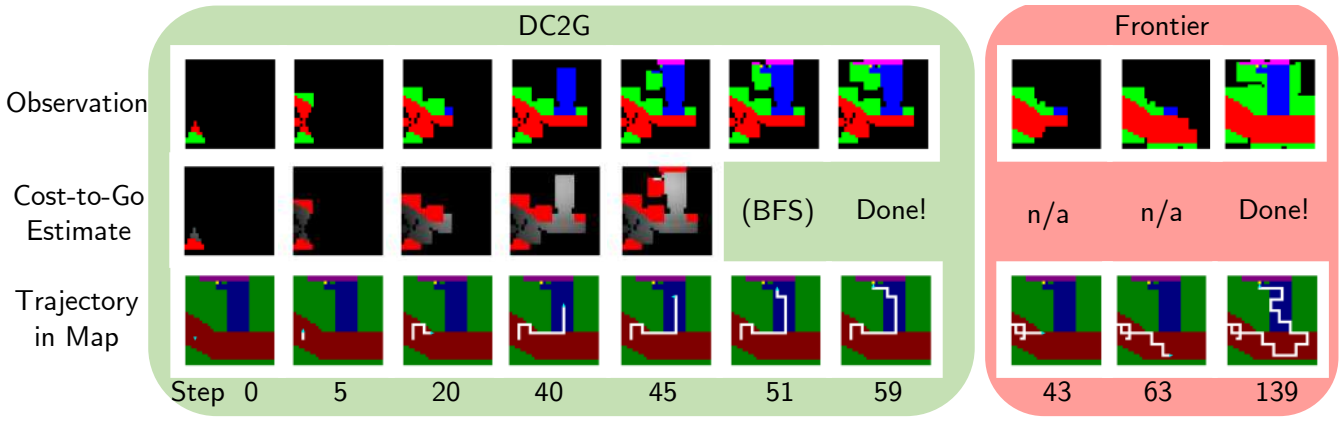
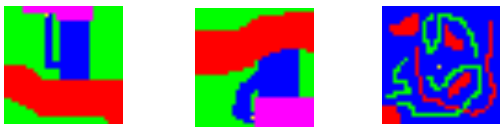
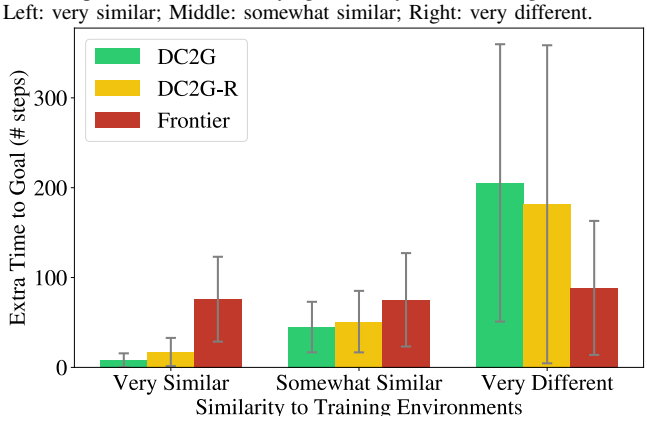


Fig. 5: Search experiment with DC2G planner. The top row is the agent's observed semantic map (generator input); the middle is its current estimate of the cost-to-go (generator output); the bottom is the trajectory so far. The DC2G agent reaches the goal much faster (59 vs. 139 steps) by using learned context. At DC2G step 20 and Frontier step 43, the observations are almost identical; yet DC2G chooses to turn up the driveway, whereas Frontier explores the roads.



(a) Examples of cases with varying similarity to the training environment. Left: very similar; Middle: somewhat similar; Right: very different.



(b) Performance

Fig. 6: Comparison of Planning Algorithms. Three planning algorithms are tested in three types of environments. For test environments with similar layouts to the training worlds (leftmost group), DC2G takes near-optimal paths to the goal, whereas the frontier planner wastes time exploring. DC2G generalizes to the somewhat similar environments (middle group), but performance declines as the environment diverges from the training worlds; the frontier planner is essentially environment agnostic.

structured environments, as opposed to generic exploration. The frontier-based exploration algorithm spends time exploring regions that are not useful for finding the goal since its sole objective is to push its frontier forward and lacks context-awareness.

D. Generalizability

A common criticism of learning-based algorithms is their lack of ability to generalize to unseen data. This paper measures the DC2G algorithm's ability to generalize by considering performance in various classes of environments, aggregated in Fig. 6. Above each of the three environment class groups, an example map is given to convey what is meant by similarity.

As the test environments diverge from the training set (middle group), the performance of DC2G (and DC2G-Rescale) declines: (DC2G: 44.9 ± 28.1, DC2G-Rescale: 50.9 ± 34.2, Frontier: 75.2 ± 52.0 extra steps). This is expected, since the cost-to-go estimate will be less accurate, and therefore the learned context is less informative. Importantly, the DC2G planners still outperform the frontier-based planner in the middle group. This result suggests the DC2G planner learned to extract context sufficiently well, such that even in strange layouts, it could still recognize some context clues that guide the exploration.

In the extreme case of generalizability, consider the test set with worlds that do not have anything (except terrain types) in common with the training set. The performance is shown in the rightmost group of Fig. 6: the DC2G planners fail miserably as expected, as many of these worlds are simply random scribbles. The frontier cell with lowest estimated cost varies widely each time a new cell is uncovered since the estimation is mostly noise on untrained inputs. Therefore, the agent wastes time driving through mapped regions (uncovering nothing new) from one frontier boundary of its partial map to another. Interestingly, the frontier-based planner performs consistently across test cases, indicative of its total lack of context-awareness.

The overall trend of Fig. 6 demonstrates that DC2G can provide substantial performance improvement in the environments that are most similar to its training environments. In practice, the training environments would be chosen such that test environments fall in the leftmost category almost all of the time, with the understanding that even in the occasional middle category, the algorithm will still perform well.

V. Conclusion

This work presented an algorithm for learning to utilize context in a structured environment in order to inform an exploration-based planner. The new approach, called Deep Cost-to-Go (DC2G), represents scene context in a semantic gridmap, learns to estimate which areas are beneficial to explore to quickly reach the goal, and then plans toward promising regions in the map. The efficient training algorithm requires zero training in simulation: a context extraction model is trained on a static dataset, and the creation of the dataset is highly automated. The algorithm outperforms pure frontier exploration by 63% in structured environments that differ from the training worlds in specific layout.

Acknowledgment

This work is supported by the Ford Motor Company.

REFERENCES

[1] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.

[2] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters." in *Robotics: Science and Systems*, vol. 2, 2005, pp. 65–72.

[3] D. Joho, M. Senk, and W. Burgard, "Learning search heuristics for finding objects in structured environments," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 319–328, 2011.

[4] M. Samadi, T. Kollar, and M. M. Veloso, "Using the web to interactively learn to find objects." in *AAAI*, 2012, pp. 2074–2080.

[5] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2168–2173.

[6] L. Kunze, C. Burbridge, and N. Hawes, "Bootstrapping probabilistic models of qualitative spatial relations for active visual object search," in *AAAI Spring Symposium*, 2014, pp. 24–26.

[7] L. Kunze, K. K. Doreswamy, and N. Hawes, "Using qualitative spatial relations for indirect object search," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 163–168.

[8] M. Lorbach, S. Höfer, and O. Brock, "Prior-assisted propagation of spatial information for object search," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 2904–2909.

[9] P. Vernaza and A. Stentz, "Learning to locate from demonstrated searches." in *Robotics: Science and Systems*, 2014.

[10] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.

[11] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, *et al.*, "Robot task planning and explanation in open and uncertain worlds," *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[12] S. Brahmbhatt and J. Hays, "Deepnav: Learning to navigate large cities," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3087–3096.

[13] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *CoRR*, vol. abs/1801.02209, 2018. [Online]. Available: http://arxiv.org/abs/1801.02209

[14] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi, "Following high-level navigation instructions on a simulated quadcopter with imitation learning," *arXiv preprint arXiv:1806.00047*, 2018.

[15] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning," in *IEEE International Conference on Robotics and Automation*, 2017.

[16] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual semantic planning using deep successor representations," *arXiv preprint ArXiv:1705.08080*, pp. 1–13, 2017.

[17] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: visual question answering in interactive environments," *CoRR*, vol. abs/1712.03316, 2017. [Online]. Available: http://arxiv.org/abs/1712.03316

[18] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *arXiv preprint arXiv:1702.03920*, vol. 3, 2017.

[19] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied Question Answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[21] A. Pronobis and R. P. Rao, "Learning deep generative spatial models for mobile robots," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 755–762.

[22] C. Richter, J. Ware, and N. Roy, "High-speed autonomous navigation of unknown environments using learned probabilities of collision," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6114–6121.

[23] C. Richter and N. Roy, "Learning to plan for visibility in navigation of unknown environments," in *2016 International Symposium on Experimental Robotics*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Cham: Springer International Publishing, 2017, pp. 387–398.

[24] C. Richter, W. Vega-Brown, and N. Roy, *Bayesian Learning for Safe High-Speed Navigation in Unknown Environments*. Cham: Springer International Publishing, 2018, pp. 325–341. [Online]. Available: https://doi.org/10.1007/978-3-319-60916-4_19

[25] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation," *arXiv preprint arXiv:1707.03034*, 2017.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[27] C. Hesse, "Image-to-image translation in tensorflow," https://affinelayer.com/pix2pix/, accessed: 2018-08-28.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[30] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1462–1468.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[32] L. W. Maxime Chevalier-Boisvert, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.

[33] A. Valada, G. L. Oliveira, T. Brox, and W. Burgard, "Deep multispectral semantic scene understanding of forested environments using multimodal fusion," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 465–477.

[34] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "MarrNet: 3D Shape Reconstruction via 2.5D Sketches," in *Advances In Neural Information Processing Systems*, 2017.

[35] X. Zhang, "Semantic slam," https://github.com/floatlazer/semantic_slam, 2018.

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.