Title

Learning robotic navigation from experience: principles, methods and recent results

Permalink
https://escholarship.org/uc/item/9zq0h6tf

Journal
Philosophical Transactions of the Royal Society B Biological Sciences, 378(1869)

ISSN
0962-8436

Authors
Levine, Sergey
Shah, Dhruv

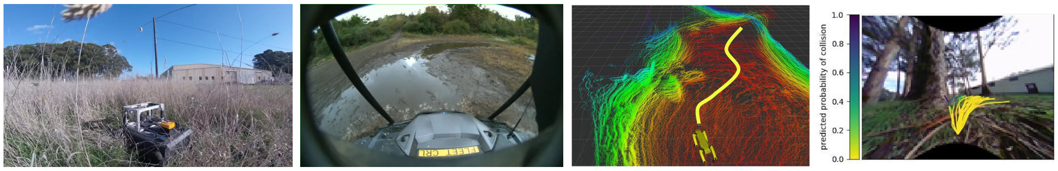
Publication Date
2023-01-30

DOI
10.1098/rstb.2021.0447

Learning Robotic Navigation from Experience: Principles, Methods, and Recent Results

Sergey Levine, Dhruv Shah
UC Berkeley
{svlevine, shah}@eecs.berkeley.edu

Abstract: Navigation is one of the most heavily studied problems in robotics, and is conventionally approached as a geometric mapping and planning problem. However, real-world navigation presents a complex set of physical challenges that defies simple geometric abstractions. Machine learning offers a promising way to go beyond geometry and conventional planning, allowing for navigational systems that make decisions based on actual prior experience. Such systems can reason about traversability in ways that go beyond geometry, accounting for the physical outcomes of their actions and exploiting patterns in real-world environments. They can also improve as more data is collected, potentially providing a powerful network effect. In this article, we present a general toolkit for experiential learning of robotic navigation skills that unifies several recent approaches, describe the underlying design principles, summarize experimental results from several of our recent papers, and discuss open problems and directions for future work.



(a) (b) (c) (d)

Figure 1: **Learning-based methods can handle situations that violate the assumptions of geometric methods**: sometimes obstacles that geometrically appear to block the robot's path, such as tall grass, are actually traversible (a), and sometimes seemingly solid ground is actually not traversible, as in the case of mud or sand traps (b). Unlike geometry-based methods [1], which plan through 3D reconstructions of the environment (c), experiential learning methods [2] learn to determine from raw sensory observations which features are traversible and which aren't (d). This, together with their ability to improve as more data is collected, makes such techniques a powerful choice for real-world navigation.

1 Introduction

Navigation represents one of the most heavily studied topics in robotics [3]. It is often approached in terms of *mapping* and *planning*: constructing a geometric representation of the world from observations, then planning through this model using motion planning algorithms [4–6]. However, such geometric approaches abstract away significant physical and semantic aspects of the navigation problem that in practice leave a range of real-world situations difficult to handle (see Figure 1). These challenges require special handling, resulting in complex systems with many components. Some works have sought to incorporate machine learning techniques to either learn navigational skills from simulation or to learn perception systems for navigation for human-provided labels. In this article, we instead argue that learned navigational models, trained directly on real-world experience rather than human-provided labels or simulators, provide the most promising long-term direction for a general solution to navigation. We refer to such learning approaches as *experiential learning*, because they learn directly from past experience of performing real-world navigation. As we will discuss in Section 2, such methods relate closely to reinforcement learning.

Geometry-based methods for navigation, based on mapping and planning, are appealing in large part because they *simplify* the navigation problem into a concise geometric abstraction: if the 3D shape of the environment can be inferred from observations, this can be used to construct an accurate geometric model, a path to the destination can be planned within this model, and that path can then be executed in the real world. However, although some idealized environments fit neatly into this geometric abstraction, real-world settings have a tendency to confound it. Obstacles are not always rigid impassable barriers (e.g., tall grass), and areas that appear geometrically passable might not be (e.g., mud, foliage, etc.). Real-world environments also exhibit patterns that are not used by purely geometric approaches: roads often (but not always) intersect at right angles, city blocks tend to be of equal size, and buildings are often rectangular. Such patterns can lead to convenient shortcuts and intuitive behaviors that are often exploited by humans.

Machine learning can offer an appealing toolkit for addressing these complex situations and exploiting such patterns, but the many different ways of utilizing machine learning for navigation come with very different tradeoffs. In this article, we will focus specifically on *experiential* learning, where a robot learns how to navigate directly from real-world navigation data. We can contrast this with four other types of approaches: (1) methods that utilize learning to handle *semantic* aspects of navigation, typically based on computer vision with human-provided labels [7–10]; (2) methods that utilize learning to assist in 3D mapping, which is then integrated into standard geometric pipelines [11–18]; (3) methods that utilize reinforcement learning in simulated environments and then employ transfer learning or domain adaptation [19–22]; (4) methods that use human-provided demonstrations to learn navigational policies [23–28].

Methods that utilize learning only for handling semantic (1) or geometric (2) perception do not address the limitations of geometry-based methods in terms of failing to understand the *physical* meaning of traversibility and navigational affordances detailed above, though they can significantly improve the performance of geometric methods and address their limitations in regard to semantics. Such techniques can help to make conventional mapping and planning pipelines more effective by endowing them with semantics or more accurate 3D reconstruction. However, like conventional mapping techniques, they do not attempt to directly predict the physical outcome of a robot's actions. This stands in contrast to experiential learning methods that directly learn which observations correspond to traversible or untraversible terrains or obstacles, though a number of works in robotic perception have incorporated elements of experiential learning, for example for learning to classify traversability [29–33].

Methods based on simulation (3) are limited in that they rely on the fidelity of the simulator to learn about the situations a robot might encounter in the real world. Although simulation methods can significantly simplify the *engineering* of navigational systems, in the end they kick the can down the road: instead of manually adding special cases (tall grass, mud, etc.) into the standard mapping pipeline, we must instead model all such possible conditions in the simulator. Sometimes it might be easier to simulate some phenomenon and then learn how to handle it than to design a controller for it directly. However, human insight is still needed to identify the phenomena to simulate, and human engineering is needed to build such simulations, in contrast to methods that learn from real data and therefore learn about how the world *actually* works. Indeed, in other domains where machine learning methods have been successfully deployed in real-world products and applications – computer vision, NLP, speech recognition, etc., [34] – such methods utilize *real* data precisely because such data provides the best final performance in the real world with the least amount of effort.

Methods based on human-provided demonstrations (4), which have a long history in robotic navigation [23–28, 35, 36], have the benefit of learning about the world as it really is, but carry a heavy price: the performance of the system is entirely limited by the number of demonstrations that are provided and does not improve with more use. In contrast, experiential learning methods [2, 37–40], which may also utilize demonstration data in combination with the robot's own experience and, crucially, do not make the assumption that all of the provided data is *good* (i.e., it should not be imitated blindly) offer the most appealing combination of benefits. Such methods handle the world the way it really is, learning traversability and navigational affordances directly from experience, improving as more data is collected and do not require an expert human engineer to model the long tail of scenarios and special conditions that a robot might encounter in the real world.

Algorithms that learn robotic policies from experience often employ "end-to-end" learning methods [41, 42]. This can either mean that the robot learns the task directly from final task outcome feedback, or that it learns directly from raw sensory perception. Both have appealing benefits, but particularly the former is a critical strength of experiential learning: only by associating actual real-world trajectories with actual real-world outcomes can a robot acquire navigational skills that are not vulnerable to the "leaky abstractions" that afflict other manually designed techniques. For example, the abstraction of geometry doesn't capture that tall grass is traversable. The abstraction of a simulator that doesn't model wheel slip doesn't capture that wheels can become stuck in mud. By learning about real outcomes from real data, such issues can be eliminated.

At the same time, as we will discuss in Section 4, learned navigation systems can (and should) still employ modularity and compositionality to solve temporally extended tasks. Indeed, we will argue that effective learning systems, like conventional mapping and planning methods, should still be divided into two parts: a *memory* or "mental map" of their environment, and a high-level *planning* algorithm that uses this mental map to choose a route. Conventional methods simply choose specific abstractions, such as meshes or points in Cartesian space, to represent this map, whereas learning-based methods *learn* a suitable abstraction from data. These learned abstractions are grounded in the things that are actually important for real-world traversability, and they improve as the robot gathers more and more experience in the environment.

The goal of this article is to provide a high-level tutorial on how navigational systems can be trained on real-world data, provide pointers to relevant recent works, and present the overall architecture that a navigational system learned from experience should have. The remainder of this article will focus on providing a high-level summary of navigation via experiential learning, algorithms for learning low-level navigational skills from data, algorithms for composing these skills to solve temporally extended navigation problems, and a brief discussion of several of our recent works that provide experimental evidence for the viability of these approaches.

2 An Overview of Experiential Learning for Navigation

The central principle behind experiential learning is to learn from actual experience of attempting (and succeeding or failing) to perform a given task, as opposed to learning from human-provided labels, such as semantic labels provided by humans (e.g., road vs. not road), or demonstrations. Perhaps the best known framework for experiential learning is reinforcement learning (RL) [43], which formulates the problem in terms of learning to maximize reward signals through active online exploration. However, we will make a distinction between the principle of experiential learning – learning how to perform a task using experience – and the *methodology* prescribed by RL. This is because the primary benefits really come from the use of experience, rather than the specific choice of algorithm (RL or otherwise). The particular methods in the case studies in Section 5 use simple supervised learning methods, though they can be seen as a particularly naïve version of offline RL [44] and could likely utilize more advanced and modern offline RL methods as well.

We can use $\mathbf{o}_t$ to denote the robot's observation at time t, $\mathbf{a}_t$ to denote its commanded action (e.g., steering and throttle commands), and $\tau = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_H, \mathbf{a}_H\}$ to denote a trajectory (i.e., a trial obtained by running the robot). The algorithm is provided with a dataset of trajectories $\mathcal{D} = \{\tau_i\}$, which it uses to learn. This can be done either *offline*, where a static dataset consisting of previously collected data is provided and the algorithm learns entirely from this dataset, or it can be done online, where the policy explores the environment, appends the resulting experience to $\mathcal{D}$, and periodically retrains the policy. The critical ingredient is the use of real trial data, *not* whether or not this data is collected online. The power of experiential learning comes from using real experience to understand which trajectories are possible, and which aren't. For example, if $\mathcal{D}$ contains a trajectory that successfully drives through tall grass, the robot can learn that tall grass is traversable. Traversals that are not seen in the data (e.g., there is no trajectory where the robot drives through a wall) should be assumed to be impossible. Of course, this presumes a high degree of coverage in the dataset, and additional online exploration can be helpful here.

It is likely that ultimately the full benefit of experiential learning will be unlocked by *combining* offline and online training, as they offer complementary benefits. The central benefit of offline training is the ability to reuse large and diverse navigational datasets. In the same way that state-of-the-art models in computer vision [45] and NLP [46] achieve remarkable generalization by training on huge datasets, effective navigational systems will work best when trained on large previously collected datasets, which would be impractical to recollect online for every experiment. At the same time, a major strength of such methods is to continue to improve as more data is collected, particularly for real-world deployments where such methods can benefit from a network effect: as more robots are deployed, more data is collected, the robots become more capable, and it becomes possible to deploy more of them in more settings.

To define the task, we can assume that something in $\mathbf{o}_t$ indicates task completion. For example, the task might be defined by a *goal* $\mathbf{o}_g$, or by a *goal location*, where the location is part of $\mathbf{o}_t$. More generally, it can be defined by some reward function $r(\mathbf{o}_t)$ or goal set $\mathbf{o}_g \in \mathcal{G}$. We will assume for now that it is defined by a single goal $\mathbf{o}_g$, though this requirement can be relaxed. The specific question that the learned model must be able to answer then becomes: given the current observation $\mathbf{o}_t$ and some goal $\mathbf{o}_g$, which action $\mathbf{a}_t$ should the robot take to eventually reach $\mathbf{o}_g$?

RL [43] and imitation learning [23, 35] offer viable solutions to this problem by learning policies of the form $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$, as we will discuss in the next section. However, it is difficult to directly learn fully reactive policies that can reach very distant goals. Instead, we can decompose the navigation problem hierarchically: the robot should build some sort of "mental map" of its surroundings, plan through this mental map, and utilize low-level navigational skills to execute this plan. Such skills might, for example, know how to navigate around a muddy puddle, cut across a grassy field, or go through a doorway in a building. But they do not reason about the longer-horizon structure of the plan, and therefore do not require memory. The role of $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ is to represent such skills and, as we will discuss in the next two sections, also to provide *abstractions* that can be used to build the higher level "mental map." This higher level, discussed in Section 4, can either be an explicit search algorithm, or can be defined implicitly as part of a memory-based (e.g., recurrent) neural network model [17, 47–52]. This hierarchy is also present in the standard mapping and planning approach, where the geometric map represents the robot's "memory," but the abstraction (3D points) is chosen manually. Viewed in this way, a central benefit of the experiential learning approach is to learn low-level skills $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ that represent navigational affordances, and then *build up its higher level mapping and planning mechanisms in terms of the capabilities of these skills*.

3 Learning Policies From Data

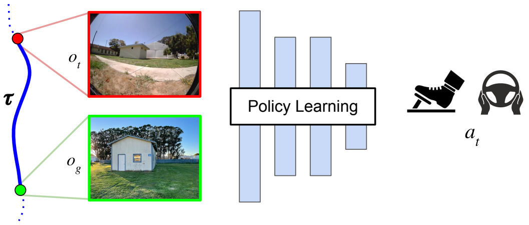


Figure 2: **Low-level navigational policies can be trained from data** by extracting tuples $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g)$, where $\mathbf{o}_t$ is an observation along a trajectory, $\mathbf{a}_t$ is the corresponding action, and $\mathbf{o}_g$ is an eventual goal that can be reached successfully after taking $\mathbf{a}_t$ in $\mathbf{o}_t$. Both supervised learning and RL-based techniques can do this by using some sort of relabeling function $g(\tau, t)$ to select $\mathbf{o}_g$ during training from the remainder of the trajectory τ from which $\mathbf{o}_t$ was taken.

Training $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ can be framed either as maximizing the probability that π reaches $\mathbf{o}_g$, minimizing the time it takes to reach $\mathbf{o}_g$, or in terms of some other metrics. In practice, methods for $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ include goal-conditioned imitation [53–57] and RL [58–68] which, though seemingly different conceptually, can be cast into the same framework. An algorithm for training $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ must provide an objective function $J_\mathcal{D}(\pi)$, which factorizes over the dataset:

$$J_\mathcal{D}(\pi) = \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{H} J_{\mathbf{o}_t, \mathbf{a}_t, \tau}(\pi).$$

We slightly abuse notation to index time steps in τ as $\mathbf{o}_t, \mathbf{a}_t$. In the case of supervised learning, $J_{\mathbf{o}_t, \mathbf{a}_t, \tau}$ is given by $J^{\text{ML}}_{\mathbf{o}_t, \mathbf{a}_t, \tau}$:

$$J^{\text{ML}}_{\mathbf{o}_t, \mathbf{a}_t, \tau}(\pi) = E_{\mathbf{o}_g \sim g(\tau, t)}[\log \pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)],$$

where $g(\tau, t)$ is a relabeling distribution that selects future observations in τ as possible goals. For example, $g(\tau, t)$ might uniformly sample all $\mathbf{o}_{t'}$ where $t' > t$, or select $\mathbf{o}_H$ or $\mathbf{o}_{t+K}$ (see Figure 2). The general idea is to train the policy to imitate the actions in the trajectory when conditioned on the current observation and *future* observations in that same trajectory. Reinforcement learning algorithms typically use either an expected Q-value objective or a weighted likelihood objective, given by

$$J^{\text{Q}}_{\mathbf{o}_t, \mathbf{a}_t, \tau}(\pi) = E_{\mathbf{o}_g \sim g(\tau, t), \mathbf{a} \sim \pi(\mathbf{a}|\mathbf{o}_t, \mathbf{o}_g)}[Q^\pi(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g)]$$
$$J^{\text{W}}_{\mathbf{o}_t, \mathbf{a}_t, \tau}(\pi) = E_{\mathbf{o}_g \sim g(\tau, t)}[w(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g) \log \pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)],$$

respectively. In the case of RL, $g(\tau, t)$ can select as goals future time steps in τ, as in the case of supervised learning ("positives"), but can also mix in observations sampled from other trajectories ("negatives") that are less likely to be reached, since the Q-function or weight will tell the policy that these "negative" goals have low values. Prior works have discussed a wide range of different relabeling strategies and their tradeoffs [58, 59, 65, 68]. The expected Q objective J^{Q} is typically used by standard actor-critic methods such as DDPG and SAC [69, 70], as well as offline RL methods such as CQL [71]. The Q-function in this case is trained via Bellman error minimization on the same data, with offline RL methods typically including some explicit regularization to avoid out-of-distribution actions. The weighted likelihood objective J^{W} is used by a number of offline RL methods, such as AWR, AWAC, and CRR [72–74], which utilize it to avoid out-of-distribution action queries. Typically, the weight $w(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g)$ is chosen to be larger for actions with large Q-values. For example, AWAC uses the weight $w(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g) = \exp(Q^\pi(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g) - V^\pi(\mathbf{o}_t, \mathbf{o}_g))$. Further technical details can be found in prior work on goal-conditioned imitation [53–57], standard online RL [59–63, 65, 69, 70], and offline RL [68, 71–74]. For the purpose of this article, note that all three loss functions have a similar structure: they all involve selecting goals $\mathbf{o}_g$ using some relabeling function $g(\tau, t)$, and they all involve somehow training $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ to favor those actions that reach $\mathbf{o}_g$, either directly using the actions that actually led to $\mathbf{o}_g$ in the data in the case of J^{ML}, or actions that have a high value for $\mathbf{o}_g$ according to a separate learned Q-function.

As discussed in the previous section, $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ by itself will not necessarily be effective at reaching distant goals, and perhaps more importantly, it does not maintain a memory of the environment, does not attempt to map it and remember the locations of landmarks, and does not perform explicit planning (though the process of training the Q-function arguably performs amortized planning via dynamic programming during training). Therefore, it is generally only effective for *short-horizon* goals. In the case of navigation tasks studied in prior work with such approaches, this typically means goals that are within line of sight of the robot, or within a few tens of meters of its present location [2, 38–40, 64, 75], though some works have explored extensions to enable significantly longer-range control in some settings, including through the use of memory and recurrence [48, 51]. As a side note, $\mathbf{o}_t$ in general may not represent a Markovian state of the system, but only an observation. The use of recurrence mitigates this issue [76], but if we only require the policies to represent short-range skills, this issue often does not cause severe problems.

In the next section, we will discuss how planning and memory can be incorporated into a complete navigational method that uses the policies $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ as *local* controllers. This will require an additional object besides the policy itself: an evaluation or distance function $D(\mathbf{o}_t, \mathbf{o}_g)$ that additionally predicts *how long* $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ will actually take to reach $\mathbf{o}_g$ from $\mathbf{o}_t$ (and if it will succeed at all). As discussed in prior work [58], this distance function can be extracted from a value function learned with RL. If we choose the reward to be -1 for all time steps when the goal is not reached (i.e., $r(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_g) = -\delta(\mathbf{o}_t \neq \mathbf{o}_g)$) and $\gamma = 1$, we have $D(\mathbf{o}_t, \mathbf{o}_g) = -V^\pi(\mathbf{o}_t, \mathbf{o}_g)$, though in practice it is convenient to use $\gamma < 1$. With supervised learning, this quantity can be learned by regressing onto the distances in the dataset, using the loss $E_{\mathbf{o}_g \sim g(\tau,t)}[(D(\mathbf{o}_t, \mathbf{o}_g) - (t' - t))^2]$, where t' is the time step in τ corresponding to $\mathbf{o}_g$.

4 Planning and High-Level Decision Making

Navigation is not just a reactive process, where a robot observes a snapshot of its environment and chooses an action. While the explicit process of exact geometric reconstruction in classic navigational methods may be obviated by learning from data, any effective navigational method likely must still retain, either explicitly or implicitly, a similar overall structure: it should acquire and remember the overall shape of the environment (though perhaps not in terms of precise geometrical detail), and it must plan through this environment to reach the final destination, while reasoning about parts of the environment that are not currently visible but were observed before and stored in memory. Indeed, it has been extensively verified experimentally that humans and animals maintain "mental maps" of environments that they visit frequently [77], and these mental maps are more likely topological rather than precise geometric reconstructions. Crucially, such mental maps depend on *abstractions* of the environment. Precise geometric maps use coordinates of vertices or points as abstractions, but these abstractions are more detailed than necessary for high-level navigation, where we would like to make decisions like "turn left at the light," and allow our low-level skills (as described in the previous section) to take care of carrying out such decisions. Thus, the problem of building effective mental maps hinges on acquiring effective abstractions.

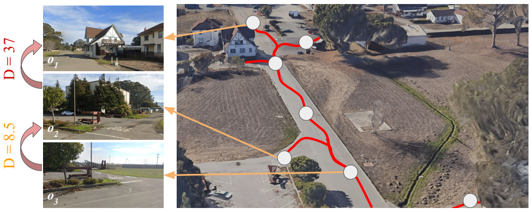


Figure 3: **Planning with learned policies.** Learned distance functions, which correspond to value functions for a low-level policy, describe the connectivity structure in the environment. This provides an *abstraction* for high-level planning informed by the capabilities of low-level skills.

A powerful idea in learning-based navigation is that the low-level policies $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ and their distance functions $D(\mathbf{o}_t, \mathbf{o}_g)$ can provide us with such abstractions. The basic principle is that $D(\mathbf{o}_t, \mathbf{o}_g)$ can describe the *connectivity* between different observations in the environment. Given input observations of two landmarks, $D(\mathbf{o}_t, \mathbf{o}_g)$ can tell us if the robot's low-level policy $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ can travel between them, which induces a graph that describes the connectivity of previously observed landmarks, as illustrated in Figure 3. Thus, storing a subset of previously seen observations represents the robot's *memory* of its environment, and the graph induced by edge weights obtained from $D(\mathbf{o}_i, \mathbf{o}_j)$ for each pair $(\mathbf{o}_i, \mathbf{o}_j)$ of stored observations then represents a kind of "mental map" – an abstract counterpart to the *geometric* map constructed by conventional SLAM algorithms. Searching through this graph can reveal efficient paths between any pair of landmarks. Critically, this mental map is not based on the geometric shape of the environment, but rather its *connectivity* according to the robot's current navigational capabilities, as described by $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ and $D(\mathbf{o}_t, \mathbf{o}_g)$. Particularly in the RL setting, where $D(\mathbf{o}_t, \mathbf{o}_g)$ corresponds to the value function of $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$, this makes it clear that the robot's low-level capabilities effectively inform its high-level abstraction, defining both the representation of its memory (i.e., the graph) and its mechanism for high-level planning (i.e., search on this graph).

Using policies and their value functions as abstractions for search and planning has been explored in a number of prior works, both in the case of RL (where distances are value functions) [64, 78] and in the case of supervised learning (where distances are learned with regression) [38, 75, 79–81], and we refer the reader to these prior works for technical details. However, the important ingredient is not *necessarily* the use of graph search, but rather the use of low-level skills to form abstractions for high-level skills. For example, it is entirely possible to dispense with the graph entirely and instead optimize over the goals using a high-level model via trajectory optimization or tree search [78, 81]. Indeed, it is entirely possible that *amortized* higher-level planning methods (i.e., higher-level RL or other learned models) might in the long run prove more effective than classic graph search, or the two paradigms might be combined, for example by using differentiable search methods as in the case of (a hierarchical variant of) value iteration networks [47] or other related methods [48, 51]. The key point is that the higher-level mapping and planning process, whether learned or not, should operate on abstractions that are informed by the (learned) capabilities of the robot.

The graph described in Figure 3 can be utilized for planning in a number of different ways. In the simplest case, the current observation $\mathbf{o}_t$ and the final desired goal $\mathbf{o}_g$ are simply connected to the graph by using $D(\mathbf{o}_i, \mathbf{o}_j)$, a graph search algorithm determines the next waypoint along the shortest path $\mathbf{o}_w$, and then $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_w)$ is used to select the action [64]. However, real-world navigation problems often require more sophisticated approaches because (1) the environment might not have been previously explored, and therefore requires simultaneously constructing the "mental map" and planning paths that move the robot toward the goal; (2) the goal might be specified with other information besides the target observation. In general, observations in a new environment might be added to the robot's memory ("mental map"), connecting them to a growing graph, and each time the robot might replan a path toward the goal. When the path to the goal cannot be determined because the environment has not been explored sufficiently, the robot might choose to explore a new location [39], or might use some sort of heuristic informed by side information, such as the spatial coordinate of the target, or even an overhead map [40]. The latter also provides a natural avenue for introducing other goal specification modalities: while the mental map is built in terms of the robot's observations, the final goal can be specified in terms of any function of this observation, including potentially its GPS coordinates [40].

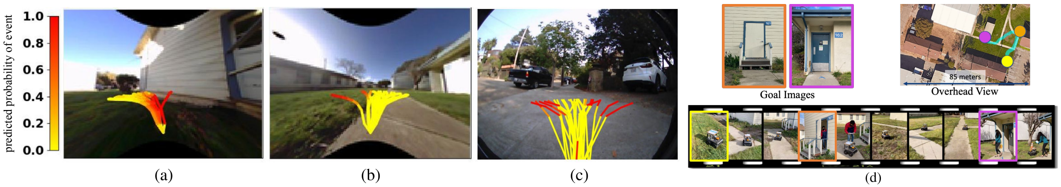


Figure 4: **Learning outdoor navigation.** BADGR [2] learns short-horizon skills from randomly collected data to minimize the risk of collision (a) or choose paths with minimum bumpiness (to stay on paved trails) (b). LaND [37] extends this system to also learn from human-provided disengagement commands, thus learning the semantics of driving on sidewalks (c) without explicit labels or rules. ViNG [38] utilizes goal-conditioned skills and combines them into long-horizon plans by building "mental maps" from prior experience in a given environment to reach a series of visually indicated goals for the task of autonomous mail delivery (d).

5 Experimental Case Studies

In this section, we will discuss selected recent works that develop experiential learning systems for robotic navigation, shown in Figure 4 and Figure 5: BADGR [2], which learns short-horizon navigational skills from autonomous exploration data, LaND [37], which extends BADGR to incorporate semantics in order to navigate sidewalks, ViNG [38], which incorporates topological "mental maps" as described in the previous section, and its two extensions: RECON [39] and ViKiNG [40], which incorporate the ability to explore new environments and utilize overhead maps, respectively.

5.1 Learning Low-Level Navigational Skills

Low-level navigational skills can be learned using model-free RL, model-based RL, or supervised learning. We illustrate a few variations in Figure 4. All of these methods only use forward-facing monocular cameras, without depth sensing, GPS, or LIDAR. BADGR [2] employs a partially model-based method for training the low-level skill, predicting various task-specific metrics based on an observation and a candidate sequence of actions. Data for this method is collected by using a randomized policy. The model is trained from this autonomously collected data, analogously to the models in Section 5.1, and can predict the probability that actions will lead to collision (visualized in Figure 4 (a)), the expected bumpiness of the terrain (Figure 4 (b)), and the location that the robot will reach, which is used to navigate to goals. LaND [37], shown in Figure 4 (c), further extends this method to also predict disengagements from a human safety monitor, with data collected by attempting to navigate real-world sidewalks. By taking actions that *avoid* expected future disengagements, the robot implicitly learns social conventions and rules, such as staying on sidewalks and avoiding driveways, which enables the LaND system to effectively navigate real-world sidewalks in the city of Berkeley, California. These case studies illustrate how experiential learning can enable robotic navigation with a variety of objectives that allow accommodating user preferences and semantic or social rules. The training labels for these objective terms are provided during data collection either automatically via on-board sensors (with collision and bumpiness) or from human interventions that happen naturally during execution (in the case of LaND).

The above methods focus on low-level skills. Figure 4 (d) illustrates ViNG [38], which integrates low-level skills, in this case represented by a goal conditioned policy $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ and distance model $D(\mathbf{o}_t, \mathbf{o}_g)$ that are trained following using the supervised learning loss in Section 5.1, with a high-level navigation strategy that builds a "mental map" graph over previously seen observations in the current environment, as detailed in Section 4. Note that this "map" does not use any explicit localization, it is constructed entirely from images previously observed in the environment. The visualization in Figure 4 (d) uses GPS tags, but these are not available to the algorithm and only used for illustration. ViNG uses goals specified by the user as images (i.e., photographs of the desired destination), and requires the robot to have previously driven through the environment to collect images that can be used to building the mental map that the system plans over. Note, however, that the low-level model $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ is trained on data from many different environments, constituting about 40 hours of total experience, while the experience needed to build the map in each environment is comparatively more modest (comprising tens of minutes). In the next two sections, we will describe how this can be extended to also handle novel environments.

5.2 Searching in Novel Environments

The methods discussed above don't give us a way to reach goals in previously unseen environments—this would require the robot to *physically search* a novel environment for the desired goal. In conventional navigation systems, this is done by simultaneously mapping the environment and updating the plan on the fly. Experiential learning systems can also do this, building up their "mental map" as they explore the new environment.

As with ViNG, we can first train the low-level policy $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{o}_g)$ and distance model $D(\mathbf{o}_t, \mathbf{o}_g)$ on data from many different environments (RECON [39], described here, uses the same dataset). However, exploring a new environment requires being able to propose *new* feasible goals that have not yet been visited, rather than simply planning over a set of previously observed landmarks. This requires the learned low-level models to support an additional operation: sampling a new *feasible* subgoal that can be reached from the current observation. Sampling entire images, though feasible with a generative model, is technically complex. Instead, RECON employs a low-dimensional latent representation of feasible subgoals, learned via a variational information bottleneck (VIB) [82]. Specifically, a latent goal embedding is computed according to a conditional encoder $q(\mathbf{z}_g|\mathbf{o}_g, \mathbf{o}_t)$, where conditioning on *both* $\mathbf{o}_g$ and $\mathbf{o}_t$ causes $\mathbf{z}_g$ to represent a kind (latent) change in state. The VIB formulation provides us with both a trained encoder $q(\mathbf{z}_g|\mathbf{o}_g, \mathbf{o}_t)$, which we can use to then train $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{z}_g)$ and $D(\mathbf{o}_t, \mathbf{z}_g)$, and a prior distribution $p_0(\mathbf{z}_g)$ that can be used to sample random latent goals. VIB training optimizes for random samples $\mathbf{z}_g \sim p_0(\mathbf{z}_g)$ to correspond to feasible random goals – essentially random nearby locations that are *reachable* from $\mathbf{o}_t$.

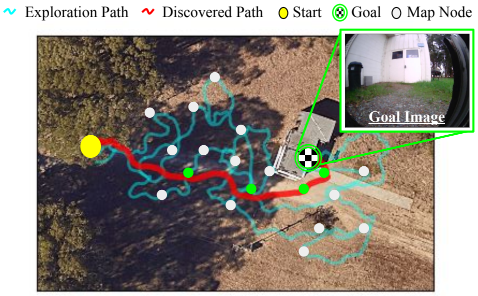


Figure 5: **Searching a novel environment** for a user-specified goal image (inset), RECON [39] incrementally builds a topological "mental map" of landmarks (white) by sampling *latent* subgoals and navigating to them (blue path). Subsequent traversals use this mental to reach the goal quickly (red).

The ability to sample random subgoals $\mathbf{z}_g$ is used by RECON in combination with a fringe exploration algorithm, which serves as the high-level planner. RECON keeps track of how often the vicinity each landmark in the graph has been visited and, if the robot cannot plan a path directly to the final goal, it plans to reach the "fringe" of the current graph, defined as landmarks with low visitation counts, and from there sample random goals $\mathbf{z}_g \sim p_0(\mathbf{z}_g)$. This causes the robot to seek out rarely visited locations and explore further from there, as illustrated in Figure 5 (blue path). After searching through the environment once, the robot can then reuse the mental map to reach the same or other goals much more quickly.

5.3 Learning to Navigate with Side Information

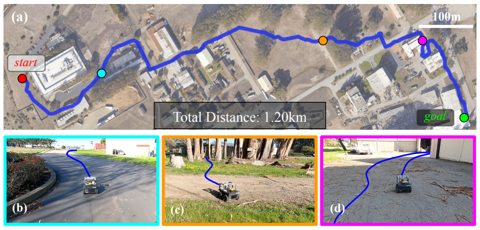


Figure 6: **Kilometer-scale navigation using geographic hints.** ViKiNG [40] can use a satellite image to perform informed search in large real-world environments (a), involving navigating on paved roads (b), through a dense patch of trees (c), while showing complex behavior like backtracking on encountering a dead-end (d).

Specifying the goal solely using an image can be limiting for more complex navigational tasks, where the robot must drive a considerable distance and simply searching the entire environment is impractical. To extend experiential learning to such settings, ViKiNG [40] further incorporates the ability to use GPS and overhead maps (e.g., satellite images or road schematics) as "heuristics" (in the A* sense) into the high-level planning process. This can be seen as somewhat analogous to how humans navigate new environments by using both geographic knowledge (e.g., from a paper map) and first-person observations, combined with patterns learned from their experience [83].

ViKiNG trains an additional *heuristic* model that receives as input the image of the overhead map, the approximate goal location (obtained via a noisy GPS measurement), and a query location, and predicts a heuristic estimate of the feasibility of reaching the goal from that location. This estimate is learned from data via a variant of contrastive learning [84]. It is then included in the search process as a heuristic, analogously to how heuristics are used in A* search, though with a modification to account for the fact that the robot is carrying out a *physical* search of the environment, and therefore should also take into consideration the time it would take for it to travel to the best current graph node from its *current* location.

In an experimental evaluation, ViKiNG is able to extract useful heuristics from satellite images and road schematics, and can navigate to destinations that are up to 2 kilometers away from the starting location in new, previously unseen environments, using low-level policies and heuristic models trained on data from other environments. Evaluated environments include hiking trails, city roads in Berkeley and Richmond in California, suburban neighborhoods, and office parks. Figure 6 shows one such experiment, where the robot successfully uses satellite image hints to navigate to a goal 1.2km away without any human interventions.

Note that the information from GPS and overhead maps is used merely as heuristics in the high-level planning algorithm, and not directly incorporated in the observation space for the low-level navigational skills. This illustrates an important principle of the low-level vs. high-level decomposition for such learning-based methods: both the low-level and high-level components can utilize learning and benefit from patterns in the environment, but they serve inherently different purposes. The low level deals with local traversibility, while the high level aims to determine which paths are more likely to lead to the destination. Note also that the approach for learning the heuristic model is fairly general, and could potentially be extended in future work to incorporate other types of hints, such as textual directions.

6 Prospects for the Future and Concluding Remarks

We discussed how experiential learning can be used to address robotic navigation problems by learning how to traverse real-world environments from real-world data. In contrast to conventional methods based on mapping and planning, methods that learn from experience can learn about how the robot *actually* interacts with the world, directly inferring which terrain features and obstacles are traversible and which ones aren't, and developing a grounded representation of the navigational affordances of the current robot in the real world. However, much like how conventional mapping and planning methods build an internal abstract model of the world and then use it for planning, learning-based methods *also*, implicitly or explicitly, construct such a model out of their experience in each environment. However, as we discuss in Section 4, in contrast to the hand-designed abstractions in geometric methods (e.g., 3D points or vertices), learning-based methods acquire these abstractions based on the capabilities of the learned skills. Thus, robots with different capabilities will end up using different abstractions, and the representations of the "mental maps" that result from such abstractions are not geometric, but rather describe the connectivity of the environment in terms of the robot's capabilities.

Such methods for robotic navigation have a number of key advantages. Besides grounding the robot's inferences about traversibility in actual experience, they can benefit from large and diverse datasets collected over the entirety of the robot's lifetime. In fact, they can in principle even incorporate data from other robots to further improve generalization [85]. Furthermore, and perhaps most importantly, such methods can continue to improve as more data is collected. In contrast to learning-based methods that utilize human-provided labels, such as imitation learning [23] and many computer vision approaches [7–10], experiential learning methods do not require any additional manual effort to be able to include more experience in the training process, so every single trajectory executed by the robot can be used for further finetuning its learned models. Therefore, such approaches will benefit richly from scale: the more robots are out there navigating in real-world environments, the more data will be gathered, and the more powerful their navigational capabilities will become. In the long run, this might become one of the largest benefits of such methods.

Of course, such approaches are not without their limitations. A major benefit of hand-designed abstractions, such as those used by geometric methods, is that the designer has a good understanding of what goes on *inside* the abstracted model. It is easy to examine a geometric reconstruction to determine if it is good, and it is comparatively easy to design an effective planning algorithm if it only needs to plan through geometric maps constructed by a given mapping algorithm (rather than a real and unpredictable environment). But such abstractions suffer considerable error when applied to real-world settings that violate their assumptions. Learning-based methods, in contrast, are much more firmly grounded in the real world, but because of this, their representations are as messy as the real world itself, making the learned representations difficult to interpret and debug. The dependence of these representations on the data also makes the construction and curation of the dataset a critical part of the design process. While workflows for evaluating, debugging, and troubleshooting supervised learning methods are mature and generally quite usable, learning-based control methods are still difficult to troubleshoot. For example, there is no equivalent to a "validation set" in learning-based control, because a learned policy will encounter a different data distribution when it is executed in the environment than it saw during training. While some recent works have sought to develop workflows, for instance, for offline RL methods [86], such research is still in its infancy, and more robust and reliable standards and workflows are needed.

Safety and robustness are also major challenges. In some sense these challenges follow immediately from the previously mentioned difficulties in regard to interpretability and troubleshooting: ultimately, a method that always works is always safe, but a method that sometimes fails can be unsafe *if it is unclear when such failures will occur*, which makes it difficult to implement mitigating measures. Therefore, approaches that improve validation of learning-based methods will likely also improve their safety. Non-learning-based methods often have more clearly defined assumptions. This can make enforcing safety constraints easier in environments where those assumptions are not violated, or where it is easy to detect such violations. However, this can present a significant barrier to real-world applications: a SLAM method that assumes static scenes can work well for indoor navigation, but is not viable for example for autonomous driving. The most challenging open-world settings could violate *all* simplifying assumptions, which might simply leave no other choice except for learning-based methods. This makes it all the more important to develop effective techniques for uncertainty estimation, out-of-distribution robustness, and intelligent control under uncertainty, which are all currently active areas of research with many open problems [87–89].

In the end, learning-based methods for robotic navigation offer a set of features that are very difficult to obtain in any other way: they provide for navigational systems that are grounded in the real-world capabilities of the robot, make it possible to utilize raw sensory inputs, improve as more data is gathered, and can accomplish all this with systems that, in terms of overall engineering, are often simpler and more compact than hand-designed mapping and planning approaches, once we account for the additional features and extensions that the latter require to handle all the edge cases that violate their assumptions. Methods based on experiential learning are still in their early days: although the basic techniques are decades old, their real-world applicability has only become feasible in recent years with the advent of effective deep neural network models. However, their benefits may make such approaches the standard for robotic navigation in the future.

Acknowledgements and Funding

The research discussed in this article was partially supported by ARL DCIST CRA W911NF-17-2-0181, the DARPA Assured Autonomy Program, and the DARPA RACER Program, as well as Berkeley AI Research and Berkeley DeepDrive.

References

[1] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, M. F. Ginting, K. Ebadi, M. Anderson, T. Pailevanian, E. Terry, M. Wolf, A. Tagliabue, T. S. Vaquero, M. Palieri, S. Tepsuporn, Y. Chang, A. Kalantari, F. Chavez, B. Lopez, N. Funabiki, G. Miles, T. Touma, A. Buscicchio, J. Tordesillas, N. Alatur, J. Nash, W. Walsh, S. Jung, H. Lee, C. Kanellakis, J. Mayo, S. Harper, M. Kaufmann, A. Dixit, G. Correa, C. Lee, J. Gao, G. Merewether, J. Maldonado-Contreras, G. Salhotra, M. S. Da Silva, B. Ramtoula, Y. Kubo, S. Fakoorian, A. Hatteland, T. Kim, T. Bartlett, A. Stephens, L. Kim, C. Bergh, E. Heiden, T. Lew, A. Cauligi, T. Heywood, A. Kramer, H. A. Leopold, C. Choi, S. Daftry, O. Toupet, I. Wee, A. Thakur, M. Feras, G. Beltrame, G. Nikolakopoulos, D. Shim, L. Carlone, and J. Burdick, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *CoRR*, 2021.

[2] G. Kahn, P. Abbeel, and S. Levine, “Badgr: An autonomous self-supervised learning-based navigation system,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[3] B. Siciliano, O. Khatib, and T. Kröger, *Springer Handbook of Robotics*. Springer, 2008, vol. 200.

[4] S. Thrun, “Simultaneous localization and mapping,” in *Robotics and cognitive approaches to spatial mapping*. Springer, 2007, pp. 13–41.

[5] 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.

[6] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, “Simultaneous localization and mapping: A survey of current trends in autonomous driving,” *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[7] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2722–2730.

[8] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, “3d semantic parsing of large-scale indoor spaces,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1534–1543.

[9] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, “Computer vision for autonomous vehicles: Problems, datasets and state of the art,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[10] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[11] F. Liu, C. Shen, G. Lin, and I. Reid, “Learning depth from single monocular images using deep convolutional neural fields,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2024–2039, 2015.

[12] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” in *European conference on computer vision*. Springer, 2016, pp. 740–756.

[13] K. Tateno, F. Tombari, I. Laina, and N. Navab, “Cnn-slam: Real-time dense monocular slam with learned depth prediction,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.

[14] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Toward geometric deep slam,” *arXiv preprint arXiv:1707.07410*, 2017.

[15] N. Yang, R. Wang, J. Stuckler, and D. Cremers, “Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833.

[16] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[17] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, “Neural topological slam for visual navigation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 875–12 884.

[18] J. Krishna Murthy, S. Saryazdi, G. Iyer, and L. Paull, “gradslam: Dense slam meets automatic differentiation,” in *arXiv*, 2020.

[19] F. Sadeghi and S. Levine, “Cad2rl: Real single-image flight without a single real image,” *arXiv preprint arXiv:1611.04201*, 2016.

[20] X. Pan, Y. You, Z. Wang, and C. Lu, “Virtual to real reinforcement learning for autonomous driving,” *arXiv preprint arXiv:1704.03952*, 2017.

[21] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[22] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9068–9079.

[23] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[24] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565–1592, 2010.

[25] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4693–4700.

[26] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[27] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 237–252.

[28] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.

[29] N. Hirose, A. Sadeghian, P. Goebel, and S. Savarese, "To go or not to go? a near unsupervised learning approach for robot navigation," *arXiv preprint arXiv:1709.05439*, 2017.

[30] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5129–5136.

[31] L. Wellhausen, R. Ranftl, and M. Hutter, "Safe robot navigation via multi-modal anomaly detection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1326–1333, 2020.

[32] S. Palazzo, D. C. Guastella, L. Cantelli, P. Spadaro, F. Rundo, G. Muscato, D. Giordano, and C. Spampinato, "Domain adaptation for outdoor robot traversability estimation from rgb data with safety-preserving loss," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 014–10 021.

[33] H. Lee and W. Chung, "A self-training approach-based traversability analysis for mobile robots in urban environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3389–3394.

[34] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[35] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[36] J. A. Bagnell, "An invitation to imitation," Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, Tech. Rep., 2015.

[37] G. Kahn, P. Abbeel, and S. Levine, "Land: Learning to navigate from disengagements," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1872–1879, 2021.

[38] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "ViNG: Learning Open-World Navigation with Visual Goals," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[39] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid exploration for open-world navigation with latent goal models," in *5th Annual Conference on Robot Learning*, 2021.

[40] D. Shah and S. Levine, "ViKiNG: Vision-based kilometer-scale navigation with geographic hints," *arXiv preprint arXiv:2202.11271*, 2022.

[41] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[42] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[43] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[44] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[47] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *Advances in neural information processing systems*, vol. 29, 2016.

[48] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2616–2625.

[49] J. Zhang, L. Tai, M. Liu, J. Boedecker, and W. Burgard, "Neural slam: Learning to explore with external memory," *arXiv preprint arXiv:1706.09520*, 2017.

[50] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[51] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell *et al.*, "Learning to navigate in cities without a map," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[52] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," *arXiv preprint arXiv:2004.05155*, 2020.

[53] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," *arXiv preprint arXiv:1912.06088*, 2019.

[54] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*. PMLR, 2020, pp. 1113–1132.

[55] S. Dasari and A. Gupta, "Transformers for one-shot visual imitation," *arXiv preprint arXiv:2011.05970*, 2020.

[56] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "The essential elements of offline rl via supervised learning," in *International Conference on Learning Representations*, 2021.

[57] R. Yang, Y. Lu, W. Li, H. Sun, M. Fang, Y. Du, X. Li, L. Han, and C. Zhang, "Rethinking goal-conditioned supervised learning and its connection to offline rl," *arXiv preprint arXiv:2202.04478*, 2022.

[58] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*, 1993, pp. 1094–1099.

[59] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[60] V. Veeriah, J. Oh, and S. Singh, "Many-goals reinforcement learning," *arXiv preprint arXiv:1806.09605*, 2018.

[61] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.

[62] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," *arXiv preprint arXiv:1811.11359*, 2018.

[63] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," *arXiv preprint arXiv:1903.03698*, 2019.

[64] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[65] B. Eysenbach, R. Salakhutdinov, and S. Levine, "C-learning: Learning to achieve goals via recursive classification," *arXiv preprint arXiv:2011.08909*, 2020.

[66] C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer, "Intrinsically motivated goal-conditioned reinforcement learning: a short survey," *arXiv preprint arXiv:2012.09830*, 2020.

[67] E. Chane-Sane, C. Schmid, and I. Laptev, "Goal-conditioned reinforcement learning with imagined subgoals," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1430–1440.

[68] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," *arXiv preprint arXiv:2104.07749*, 2021.

[69] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[70] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[71] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *arXiv preprint arXiv:2006.04779*, 2020.

[72] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[73] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[74] Z. Wang, A. Novikov, K. Zolna, J. S. Merel, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess *et al.*, "Critic regularized regression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7768–7778, 2020.

[75] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[76] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[77] P. Gould and R. White, *Mental maps*. Routledge, 2012.

[78] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[79] S. Emmons, A. Jain, M. Laskin, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse graphical memory for robust planning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5251–5262, 2020.

[80] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf, "Learning to plan with uncertain topological maps," in *European Conference on Computer Vision*. Springer, 2020, pp. 473–490.

[81] B. Ichter, P. Sermanet, and C. Lynch, "Broadly-exploring, local-policy trees for long-horizon task planning," *arXiv preprint arXiv:2010.06491*, 2020.

[82] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[83] J. M. Wiener, S. J. Büchner, and C. Hölscher, "Taxonomy of human wayfinding tasks: A knowledge-based approach," *Spatial Cognition & Computation*, 2009.

[84] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2019.

[85] K. Kang, G. Kahn, and S. Levine, "Hierarchically integrated models: Learning to navigate from heterogeneous robots," in *5th Annual Conference on Robot Learning*, 2021.

[86] A. Kumar, A. Singh, S. Tian, C. Finn, and S. Levine, "A workflow for offline model-free robotic reinforcement learning," *arXiv preprint arXiv:2109.10813*, 2021.

[87] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[88] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[89] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *Advances in neural information processing systems*, vol. 30, 2017.