Combining Optimal Control and Learning for Visual Navigation in Novel Environments

Somil Bansal[*1] **Varun Tolani**[*1] **Saurabh Gupta**[2] **Jitendra Malik**[1,3] **Claire Tomlin**[1]

Abstract: Model-based control is a popular paradigm for robot navigation because it can leverage a known dynamics model to efficiently plan robust robot trajectories. However, it is challenging to use model-based methods in settings where the environment is *a priori* unknown and can only be observed partially through on-board sensors on the robot. In this work, we address this short-coming by coupling model-based control with learning-based perception. The learning-based perception module produces a series of *waypoints* that guide the robot to the goal via a collision-free path. These waypoints are used by a model-based planner to generate a smooth and dynamically feasible trajectory that is executed on the physical system using feedback control. Our experiments in simulated real-world cluttered environments and on an actual ground vehicle demonstrate that the proposed approach can reach goal locations more reliably and efficiently in novel environments as compared to purely geometric mapping-based or end-to-end learning-based alternatives. Our approach does not rely on detailed explicit 3D maps of the environment, works well with low frame rates, and generalizes well from simulation to the real world. Videos describing our approach and experiments are available on the project website[4].

Keywords: Optimal Control, Learning for Visual Navigation, Learning for Control

1 Introduction

Developing a fully autonomous robot that can navigate in *a priori* unknown environments is difficult due to challenges that span dynamics modeling, on-board perception, localization and mapping, trajectory generation, and optimal control. One way to approach this problem is to generate a globally-consistent geometric map of the environment, and use it to compute a collision-free trajectory to the goal using optimal control and planning schemes. However, the real-time generation of a globally consistent map tends to be computationally expensive, and can be challenging in texture-less environments or in the presence of transparent, shiny objects, or strong ambient lighting [1]. Alternative approaches employ end-to-end learning to side-step this explicit map estimation step. However, such approaches tend to be extremely sample inefficient and highly specialized to the system they were trained on [2].

In this paper, we present a framework for autonomous, vision-based navigation in novel cluttered indoor environments under the assumption of perfect robot state measurement. We take a factorized approach to navigation that uses *learning* to make high-level navigation decisions in unknown environments and leverages *optimal control* to produce smooth trajectories and a robust tracking controller. In particular, we train a Convolutional Neural Network (CNN) that incrementally uses the current RGB image observations to produce a sequence of intermediate states or *waypoints*. These waypoints are produced to guide the robot to the desired target location via a collision-free path in previously unknown environments, and are used as targets for a model-based optimal controller to generate smooth, dynamically feasible control sequences to be executed on the robot. Our approach, *LB-WayPtNav* (Learning-Based WayPoint Navigation), is summarized in Fig. 1.

LB-WayPtNav benefits from the advantages of classical control and learning-based approaches in a way that addresses their individual limitations. Learning can leverage statistical regularities to make predictions about the environment from partial views (RGB images) of the environment, allowing

[*]The first two authors contributed equally to this paper.
[1] University of California, Berkeley. [2] University of Illinois, Urbana-Champaign. [3] Facebook AI Research.
[4] Videos and an extended version of this article are available at: https://vtolani95.github.io/WayPtNav/.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

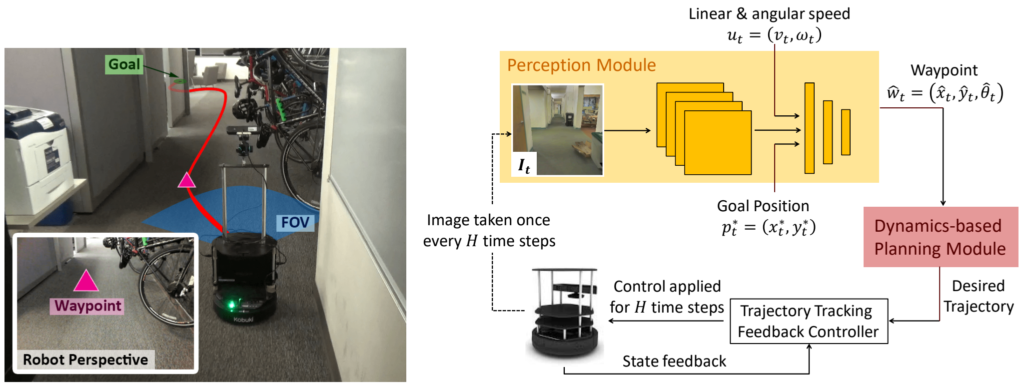


Figure 1. Overview: We consider the problem of navigation from a start position to a goal position. Our approach (LB-WayPtNav) consists of a learning-based perception module and a dynamics model-based planning module. The perception module predicts a waypoint based on the current first-person RGB image observation. This waypoint is used by the model-based planning module to design a controller that smoothly regulates the system to this waypoint. This process is repeated for the next image until the robot reaches the goal.

for generalization to unknown environments. Leveraging underlying dynamics and feedback-based control leads to smooth, continuous, and efficient trajectories that are naturally robust to variations in physical properties and noise in actuation, allowing us to deploy our framework directly from simulation to real-world. Furthermore, learning now does not need to spend interaction samples to learn about the dynamics of the underlying system, and can exclusively focus on dealing with generalization to unknown environments. To summarize, our key contributions are:

- an approach that combines learning and optimal control to robustly maneuver the robot in novel, cluttered environments using only a single on-board RGB camera,
- through simulations and experiments on a mobile robot, we demonstrate that our approach is *better* and more *efficient* at reaching the goals, results in *smoother* trajectories, as compared to End-to-End learning, and more *reliable* than geometric mapping-based approaches,
- we demonstrate that our approach can be *directly* transferred from simulation to unseen, real-world environments without any finetuning or data collection in the real-world,
- an optimal control method for generating optimal waypoints to support large-scale training of deep neural networks for autonomous navigation without requiring any human labeling.

2 Related Work

Classical Robot Navigation. Classical robotics has made significant progress by factorizing the problem of robot navigation into sub-problems of mapping and localization [3, 4], path planning [5], and trajectory tracking. Mapping estimates the 3D structure of the world (using RGB / RGB-D images / LiDAR scans), which is used by a planner to compute paths to goal. However, such purely geometric intermediate representations do not capture navigational affordances (such as: to go far away, one should step into a hallway, *etc.*). Furthermore, mapping is challenging with just RGB observations, and often unreliable even with active depth sensors (such as in presence of shiny or thin objects, or in presence of strong ambient light) [1]. This motivates approaches that leverage object and region semantics during mapping and planning [6, 7]; however, such semantics are often hand-coded. Our work is similarly motivated, but instead of using geometry-based reasoning or hand-crafted heuristics, we employ learning to directly predict good waypoints to convey the robot to desired target locations. This also side-steps the need for explicit map-building.

End-to-End (E2E) Learning for Navigation. There has been a recent interest in employing end-to-end learning for training goal-driven navigation policies [8, 9, 10, 11]. The motivation here is to incorporate semantics and common-sense reasoning into navigation policies. While Zhu *et al.* [8] learn policies that work well in training environments, Gupta *et al.* [9] and Khan *et al.* [10] design policies that generalize to previously unseen environments. Most such works abstract out dynamics and work with a set of macro-actions (going forward x cm, turning $\theta°$). Such ignorance of dynamics results in jerky and inefficient policies that exhibit stop-and-go behavior on real robots. Several works also use E2E learning for navigation using laser scans [12, 13, 14], or for training and combining a local planner with a higher level roadmap for long range navigation [15, 16]. Numerous other works have tackled navigation in synthetic game environments [17, 18, 19], largely ignoring considerations of real-world deployment (i.e. dynamics and state estimation). Researchers have also used learning to tackle locomotion problems [20, 21, 22, 23] for collision-avoidance. Kahn *et al.* [21] use motion primitives, while Gandhi *et al.* [20], and Sadeghi and Levine [22] use velocity control for locomotion. While all of these works implement policies for collision avoidance via low-level control, our work studies how policy learning itself should be modified for dynamically feasible low-level control for goal-driven behavior.

Combining Optimal Control and Learning. A number of papers seek to combine the best of learning with optimal control for high-speed navigation [24, 25, 26, 27, 28, 29]. Drews *et al.* [25, 30] learn a cost function from monocular images for aggressive race-track driving via Model Predictive Control (MPC). Kaufmann *et al.* [31, 26] use learning to predict waypoints that are used with model-based control for drone racing. The focus of these works is on aggressive control in *training race-track environments*, whereas we seek to learn *goal-driven* policies that work well in *completely novel, cluttered, real-world testing environments*. This renders their approach for waypoint generation for learning (that does not reason about obstacles explicitly) ineffective. In contrast, waypoints generated by our optimal control based method, are guaranteed to generate a collision free trajectory. Muller *et al.* [32] predict waypoints from semantically segmented images and a user provided command for outdoor navigation, and use a PID controller for control. However, they do not explicitly handle obstacles and focus primarily on lane keeping and making turns. Instead, we use a model-based planner to generate rich, agile, and explicitly dynamically feasible and collision-free control behaviors in cluttered real-world indoor environments. In a work parallel to ours, Meng *et al.* [33] combine Riemannian Motion Policy with learning for autonomous navigation, whereas we focus on dynamically feasible spline-based policies. Bansal *et al.* [29] focus on learning driving policies in outdoor navigation settings using top-views and pre-planned routes. In contrast, our models learn to discover and execute collision-free paths to goals in uninstrumented, novel indoor environments using just monocular RGB images and goal specification. Other works such as that from Levine *et al.* [34] and Pan *et al.* [35] combine optimal control and end-to-end learning, by training neural network policies to mimic the optimal control values based on the raw images. We explicitly compare to such an approach in this work.

3 Problem Setup

In this work, we study the problem of autonomous navigation of a ground vehicle in previously unknown indoor environments. We assume that odometry is perfect (*i.e.*, the exact vehicle state is available), and that the environment is static. Dealing with imperfect odometry and dynamic environments are problems in their own right, and we defer them to future work. We model our ground vehicle as a three-dimensional non-linear Dubins car system with dynamics:

$$\dot{x} = v\cos\phi, \quad \dot{y} = v\sin\phi, \quad \dot{\phi} = \omega\,, \tag{1}$$

where $z_t := (x_t, y_t, \phi_t)$ is the state of vehicle, $p_t = (x_t, y_t)$ is the position, ϕ_t is the heading, v_t is the speed, and ω_t is the angular speed at time t. The input (control) to the system is $u_t := (v_t, \omega_t)$. The inputs v_t and ω_t are bounded within $[0, \bar{v}]$ and $[-\bar{\omega}, \bar{\omega}]$ respectively. We use a discretized version of the dynamics in Eqn. (1) for all planning purposes. The robot is equipped with a forward-facing, monocular RGB camera mounted at a fixed height and oriented at a fixed pitch. The goal of this paper is to learn control policies for goal-oriented navigation tasks: the robot needs to go to a target position, $p^* = (x^*, y^*)$, specified in the robot's coordinate frame (*e.g.*, 11m forward, 5m left), without colliding with any obstacles. These tasks are to be performed in novel environments whose map or topology is not available to the robot. In particular, at a given time step t, the robot with state z_t receives as input an RGB image of the environment $\mathcal{E}$, $I_t = I(\mathcal{E}, z_t)$, and the target position $p^*_t = (x^*_t, y^*_t)$ expressed in the current coordinate frame of the robot. The objective is to obtain a control policy that uses these inputs to guide the robot to the target as quickly as possible.

4 Model-based Learning for Navigation

We use a learning-based waypoint approach to navigation (LB-WayPtNav). The LB-WayPtNav framework is demonstrated in Figure 1 and summarized in Algorithm 1. LB-WayPtNav makes use of two submodules: perception and planning.

Algorithm 1 Model-based Navigation via Learned Waypoint Prediction

Require: $p^* := (x^*, y^*)$ ▷ Goal location
1: **for** $t = 0$ to T **do**
2: $z_t := (x_t, y_t, \phi_t)$; $u_t := (v_t, \omega_t)$ ▷ Measured robot pose, and linear and angular speed
3: **Every** H steps **do** ▷ Replan every H steps
4: $p_t^* := (x_t^*, y_t^*)$ ▷ Goal location in the robot's coordinate frame
5: $\hat{w}_t = \psi(I_t, u_t, p_t^*)$ ▷ Predict next waypoint
6: $\{z^*, u^*\}_{t:t+H} = FitSpline(\hat{w}_t, u_t)$ ▷ Plan spline-based smooth trajectory
7: $\{k, K\}_{t:t+H} = LQR(z^*_{t:t+H}, u^*_{t:t+H})$ ▷ Tracking controller
8: $u_{t+1} = K_t(z_t - z_t^*) + k_t$ ▷ Apply control
9: **end for**

4.1 Perception Module

We implement the perception module using a CNN that takes as input a 224×224 pixel RGB image, I_t, captured from the onboard camera, the target position, p_t^*, specified in the vehicle's current coordinate frame, and vehicle's current linear and angular speed, u_t, and outputs the desired next state or a waypoint $\hat{w}_t := (\hat{x}_t, \hat{y}_t, \hat{\theta}_t) = \psi(I_t, u_t, p_t^*)$ (Line 5 in Algorithm 1). Intuitively, the network can use the presence of surfaces and furniture objects like floors, tables, and chairs in the scene, alongside the learned priors about their shapes to generate an estimate of the next waypoint, without explicitly building an accurate map of the environment. This allows for efficient exploration in novel environments that is guided by the robot's prior experience with similar scenes and objects.

4.2 Planning and Control Module

Given a waypoint $\hat{w}_t$, and the current linear and angular speed u_t, the planning module uses the system dynamics in Eqn. (1) to design a smooth trajectory, satisfying the dynamics and control constraints, from the current vehicle state to the waypoint. In this work, we represent the x and y trajectories using third-order splines, whose parameters can be obtained using $\hat{w}_t$ and u_t [36]. This corresponds to solving a set of linear equations, and thus, planning can be done efficiently onboard. Since the heading of the vehicle can be determined from the x and y trajectories, a spline-based planner ultimately provides the desired state and control trajectories, $\{z^*, u^*\}_{t:t+H} = FitSpline(\hat{w}_t, u_t)$, that the robot follows for the time horizon $[t, t+H]$ to reach the waypoint $\hat{w}_t$ (Line 6). Since the splines are third-order, the generated speed and acceleration trajectories are smooth. This is an important consideration for real robots, since jerky trajectories might lead to compounding sensor errors, poor tracking, or hardware damage [37]. While we use splines in this work for computational efficiency, other model-based planning schemes can also be used for trajectory planning.

To track the generated trajectory $\{z^*, u^*\}$, we design a LQR-based **linear feedback controller** [38], $\{k, K\}_{t:t+H} = LQR(z^*_{t:t+H}, u^*_{t:t+H})$ (Line 7). Here k and K represent the feed-forward and feedback terms respectively. The LQR controller is obtained using the dynamics in Eqn. (1), linearized around the trajectory $\{z^*, u^*\}$. LQR is a widely used feedback controller in robotic systems to make planning robust to external disturbances and mismatches between the dynamics model and the actual system [39]. This feedback controller allows us to deploy the proposed framework directly from simulation to a real robot (provided the real-world and simulation environments are visually similar), even though the model in Eqn. (1) may not capture the true physics of the robot.

The control commands generated by the LQR controller are executed on the system over a time horizon of H seconds (Line 8), and then a new image is obtained. Consequently, a new waypoint and plan are generated. This entire process is repeated until the robot reaches the goal position.

4.3 Training Details

LB-WayPtNav's perception module is trained via supervised learning in training environments where the underlying map is known. No such assumption is made at test time; the robot relies only on an RGB image and other on-board sensors. Knowledge of the map during training allows us to compute optimal waypoints (and trajectories) by formulating the navigation problem between randomly sampled pairs of start and goal locations as an optimal control problem, which can be solved using MPC (described in Section 8.2 in the extended version of this paper [40]). The proposed method

Table 1. Quantitative Comparisons in Simulation: Various metrics for different approaches across the test navigation tasks: success rate (higher is better), average time to reach goal, jerk and acceleration along the robot trajectory (lower is better) for successful episodes. LB-WayPtNav conveys the robot to the goal location more often, faster, and produces considerably less jerky trajectories than E2E learning approach. Since LB-WayPtNav only uses the current RGB image, whereas the geometric mapping and planning approach integrates information from perfect depth images, it outperforms LB-WayPtNav in simulation. However, performance is comparable when the mapping based approach only uses the current image (like LB-WayPtNav, but still depth *vs.* RGB).

Agent	**Input**	**Success (%)**	**Time taken (s)**	**Acceleration (m/s^2)**	**Jerk (m/s^3)**
Expert	Full map	100	10.78 ±2.64	0.11 ±0.03	0.36 ±0.14
LB-WayPtNav (our)	RGB	80.65	11.52 ±3.00	0.10 ±0.04	0.39 ±0.16
End To End	RGB	58.06	19.16 ±10.45	0.23 ±0.02	8.07 ±0.94
Mapping (memoryless)	Depth	86.56	10.96 ±2.74	0.11 ±0.03	0.36 ±0.14
Mapping	Depth + Spatial Memory	97.85	10.95 ±2.75	0.11 ±0.03	0.36 ±0.14

does not require any human labeling and can be used to generate supervision for a variety of ground and aerial vehicles. Given first-person images and relative goal coordinates as input, the perception module is trained to predict these optimal waypoints. Note that, unlike most navigational studies, we not only use smooth splines for test-time planning, but also for generating supervision for the network. This is crucial for performance as training and test time mismatch in trajectory generation can lead to undesirable behaviors (i.e. simply outputting a collision free point is not enough, we need to reason about how the robot will get there, in the same way as it would at test time).

5 Simulation Experiments

LB-WayPtNav is aimed at combining classical optimal control with learning for interpreting images. In this section, we present experiments in simulation, and compare to representative methods that only use E2E learning (by ignoring all knowledge about the known system), and that only use geometric mapping and path planning (and no learning).

Simulation Setup: Our simulation experiments are conducted in environments derived from scans of real world buildings (from the Stanford large-scale 3D Indoor Spaces dataset [41]). Scans from 2 buildings were used to generate training data to train LB-WayPtNav. 185 test episodes (start, goal position pairs) in a 3rd *held-out* building were used for testing the different methods. Test episodes are sampled to include scenarios such as: going around obstacles, going out of the room, going from one hallway to another. Though training and test environments consists of indoor offices and labs, their layouts and visual appearances are quite different (see Section 8.4 in [40] for some images).

Implementation Details: We used a pre-trained ResNet-50 [42] as the CNN backbone for the perception module, and finetuned it on 125K data points for waypoint prediction (control commands for E2E learning) with MSE loss using the Adam optimizer. We also perform data augmentation during training by applying a variety of random distortions to images which significantly improves the generalizability of our framework to unseen environments. More training details and the exact CNN architecture are provided in Section 8.1 in [40].

Comparisons: We compare to two alternative approaches. *E2E Learning:* This approach is trained to directly output velocity commands corresponding to the optimal trajectories produced by the spline-based planner (the same trajectories used to generate supervision for LB-WayPtNav). This is a purely learning based approach that does not explicitly use any system knowledge at test time. *Geometric Mapping and Planning:* This approach represents a learning-free, purely geometric approach. As inferring precise geometry from RGB images is challenging, we provide ideal depth images as input to this approach. These depth images are used to incrementally build up an occupancy map of the environment, that is used with the same spline-based planner (that was used to generate the expert supervision, see Sec. 8.2 in [40]), to output the velocity controls. Results reported here are with control horizon, $H = 1.5s$. We also tried $H = 0.5, 1.0$, but the results and trends were the same.

Metrics: We make comparisons via the following metrics: success rate (if the robot reaches within $0.3m$ of the goal position without any collisions), the average time to reach the goal (for the successful episodes), and the average acceleration and jerk along the robot trajectory. The latter metrics measure execution smoothness and efficiency with respect to time and power.

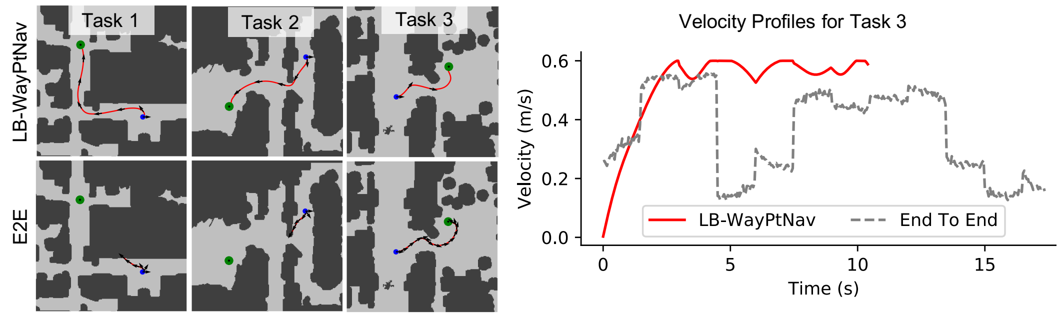


Figure 2. Trajectory Visualization: We visualize the trajectories produced by the model-based planning approach (top row) and the end-to-end (E2E) learning approach (bottom row) for sample test tasks. The E2E learning approach struggles to navigate around the tight corners or narrow hallways, whereas LB-WayPtNav is able to produce a smooth, collision-free trajectory to reach the target position. Even though both approaches are able to reach the target position for task 3, LB-WayPtNav takes only 10s to reach the target whereas the E2E learning approach takes about 17s. Moreover, the control profile produced by the E2E learning approach is significantly more jerky than LB-WayPtNav, which is a concern for the real robot. Jerky control profiles are power inefficient, can lead to significant errors in sensors and cause hardware damage.

5.1 Results

Comparison with the End-to-End learning approach. Table 1 presents quantitative comparisons. We note that LB-WayPtNav conveys the robot to the goal location more often (22% higher success rate), much faster (40% less time to reach the goal), and with less power consumption (50% less acceleration). Figure 2(left) shows the top view visualization of trajectories executed by the two methods. Top view maps are only used for visualization, both methods operate purely based on first-person RGB image inputs. As LB-WayPtNav uses a model-based planner to compute exact controls, it only has to learn "where to go" next, as opposed to the E2E method that also needs to learn "how to go" there. Consequently, LB-WayPtNav is able to successfully navigate through narrow hallways, and make tight turns around obstacles and corners, while E2E method struggles. This is further substantiated by the velocity control profiles in Figure 2(right). Even though the E2E method was trained to predict smooth control profiles (as generated by the expert policy), the control profiles at test time are still discontinuous and jerky. We also experimented with adding a smoothing loss while training the E2E policy; though it helped reduce the average jerk, there was also a significant decline in the success rate. This indicates that learning both an accurate and a smooth control profile can be a hard learning problem. In contrast, as LB-WayPtNav uses model-based control for computing control commands, it achieves average acceleration and jerk that is as low as that of an expert.[1] This distinction has a significant implication for actual robots since the consumed power is directly proportional to the average acceleration. Hence, for the same battery capacity, LB-WayPtNav will drive the robot twice as far as compared to E2E learning.

Comparison with Geometric Mapping and Planning Approach. We note that an online geometric mapping and planning approach, when used with ideal depth image observations, achieves near-expert performance. This is not surprising as perfect depth and egomotion satisfies the exact assumptions made by such an approach. Since LB-WayPtNav is a reactive planning framework, we also compare to a memory-less version that uses a map derived from *only* the current depth image. Even though the performance of memory-less planner is comparable to LB-WayPtNav, it still outperforms slightly due to the perfect depth estimation in simulation. However, since real-world depth sensors are neither perfect nor have an unlimited range, we see a noticeable drop in the performance of mapping-based planners in real-world experiments as discussed in Section 6.

Visualization of Learned Navigation Affordances. We conducted analysis to understand what cues LB-WayPtNav leverages in order to solve navigation tasks. Figure 3 shows two related navigation tasks where the robot is initialized in the same state, but is tasked to either go inside a close by room (Case A), or to a room that is further down the hallway (Case B). LB-WayPtNav correctly picks appropriate waypoints, and is able to reason that a goal in a room further away is better achieved by going down the hallway as opposed to entering the room currently in front of the robot.

[1]For a fair comparison, we report these metrics over only those test tasks at which all approaches succeed.

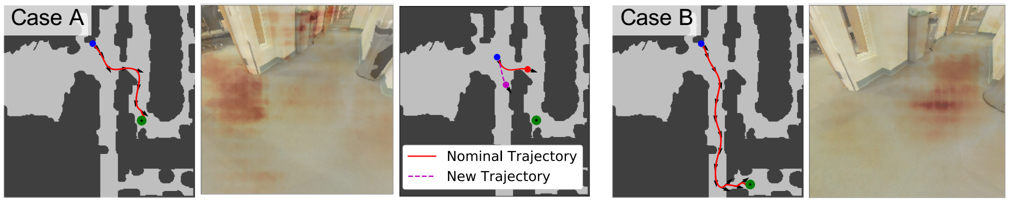
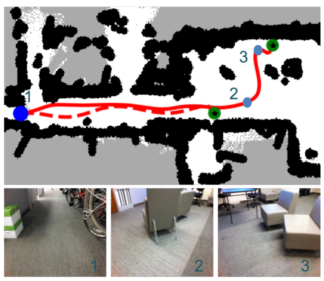
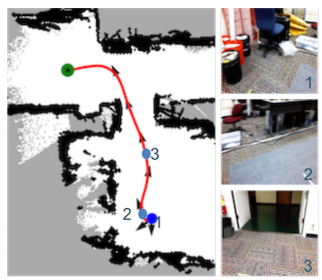
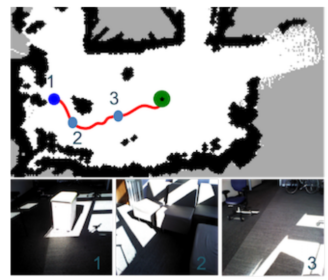


Figure 3. LB-WayPtNav is able to learn the appropriate navigation cues, such as entering the room through the doorway for a goal inside the room, continuing down the hallway for a farther goal. Such cues enable the robot to navigate efficiently in novel environments.

Table 2. Experiment setups, with top-views (obtained offline only for visualization), and sample images. Robot starts at the blue dot, and has to arrive at the green dot. Path taken by LB-WayPtNav is shown in red.

Experiment 1 and 2	**Experiment 3**	**Experiment 4**
		
Navigation through cluttered environments: This tests if the robot can skillfully pass through clutter in the real world: a narrow hallway with bikes on a bike-rack on one side, and an open space with chairs and sofas.	**Leveraging navigation affordances:** This tests use of semantic cues for effective navigation. Robot starts inside a room facing a wall. Robot needs to realize it must exit the room through the doorway in order to reach the target location.	**Robustness to lighting conditions:** Experiment area is similar to that used for experiment 1, but experiment is performed during the day when sunlight comes from the windows. Robot needs to avoid objects to get to the goal.

We also conduct an occlusion sensitivity analysis [43], where we measure the change in the predicted waypoint as a rectangular patch is zeroed out at different locations in the input image. We overlay the magnitude of this change in prediction on the input image in Red. LB-WayPtNav focuses on the walls, doorways, hallways and obstacles such as trash cans as it predicts the next waypoint, and what the network attends to depends on where the robot is trying to go. Furthermore, for Case A, we also show the changed waypoint (in pink) as we zero out the wall pixels. This corresponds to a shorter path to the goal in the absence of the wall. More such examples in Section 8.5 in [40].

Failure Modes. LB-WayPtNav can navigate in novel environments, but it can only do local reasoning (there is no memory in the network). The most prominent failure modes are: a) when the robot is too close to an obstacle, and b) situations that require 'backtracking' from an earlier planned path.

6 Hardware Experiments

We next test LB-WayPtNav on a TurtleBot 2 hardware testbed.[2] We use the network trained in simulation, as described in Section 5, and deploy it directly on the TurtleBot without any additional training or finetuning. We tested the robot in two different buildings, neither of which is in the training dataset (in fact, not even in the S3DIS dataset).[3] For state measurement, we use the on-board odometry sensors on the TurtleBot. Test environments for the experiments are described in Table 2.

We repeat each experiment for our method and the three baselines: E2E learning, mapping-based planner, and a memoryless mapping-based planner, for 5 trials each. Results across all 20 trials are summarized in Table 3, where we report success rate, time to reach the goal, acceleration and jerk.

Comparison to E2E learning are consistent with our conclusions from simulation experiments. LB-WayPtNav results in more reliable, faster, and smoother robot trajectories.

[2]More details about the hardware testbed in Sec. 8.6 in [40].

[3]Representative images of our experiment environments are shown in Figure 8 in Section 8.4 in [40].

Table 3. Quantitative Comparisons for Hardware Experiments: We deploy LB-WayPtNav and baselines on a TurtleBot 2 hardware testbed for four navigation tasks for 5 trials per task. We report the success rate (higher is better), average time to reach goal, jerk and acceleration along the robot trajectory (lower is better).

Agent	Input	Success (%)	Time taken (s)	Acceleration (m/s^2)	Jerk (m/s^3)
LB-WayPtNav (our)	RGB	95	22.93 ±2.38	0.09 ±0.01	3.01 ±0.38
End To End	RGB	50	33.88 ±3.01	0.19 ±0.01	6.12 ±0.18
Mapping (memoryless)	RGB-D	0	N/A	N/A	N/A
Mapping	RGB-D + Spatial Memory	40	22.13 ±0.54	0.11 ±0.01	3.44 ±0.21

Comparison to Geometric Mapping and Planning. Geometric mapping and planning is implemented using the RTAB-Map package [44]. RTAB-Map uses RGB-D images as captured by an on-board camera to output an occupancy map that is used with our spline-based planner to output motor commands. As our approach only uses the current image, we also report performance of a memory-less variant of this baseline where occupancy information is derived only from the current observation. While LB-WayPtNav is able to solve 95% of the trials, this memory-less baseline completely fails. It tends to convey the robot too close to obstacles, and fails to recover. In comparison, the map building scheme performs better, with a 40% success rate. This is significantly lower than performance of our method (95%), and its near perfect performance in simulation. We found that this poor performance is largely due to imperfections in depth measurements in the real world. The depth sensor fails to image shiny, matte, and small, detailed objects such as bikes, computer monitors, and thin chair legs. These systematically missing obstacles cause the robot to collide in experiment 1 and 2. Map quality also substantially deteriorates in the presence of strong infrared light (e.g. strong sunlight in experiment 4 - see videos and visualizations in [40]). These are known fundamental issues with depth sensors that limit the performance of classical navigation stacks that rely on them.

Performance of LB-WayPtNav: In contrast, our proposed learning-based scheme performs much better without need for extra instrumentation in the form of depth sensors, and without building explicit maps, for the short-horizon tasks that we considered. LB-WayPtNav precisely controls the robot through narrow hallways with obstacles (as in experiment 1 and 2) while maintaining a smooth trajectory. This is particularly striking, as the dynamics model used in simulation is only a crude approximation of the physics of a real robot (it does not include any mass and inertia effects, for example). The LQR feedback controller compensates for these approximations, and enables the robot to closely track the desired trajectory (to an accuracy of $4cm$ in experiment 1). LB-WayPtNav also successfully leverages navigation cues (in experiment 3 when it exits the room through a doorway), even when such a behavior was never hard-coded. Furthermore, thanks to the aggressive data augmentation, LB-WayPtNav performs well even under extreme lighting conditions in Experiment 4.

Furthermore, LB-WayPtNav is agile and can adapt to changes in the environment. In an additional experiment (Figure 4), we change the environment as the robot executes its policy. The robot's goal is to go straight $6m$. As the policy is executed, we move the chair to repeatedly block the robot's path (we show the chair locations in brown, blue, and purple, and mark the corresponding positions of the robot at which the chair was moved by same colors). We decrease the control horizon to $0.5s$ for this experiment for faster visual feedback. The robot successfully avoids the moving obstacle and reaches the target without colliding.

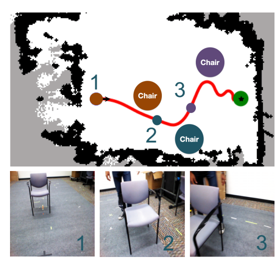


Figure 4. LB-WayPtNav can adapt to dynamic environments.

7 Conclusion and Future Work

We propose LB-WayPtNav, a navigation framework that combines learning and model-based control for goal-driven navigation in novel indoor environments. LB-WayPtNav is better and more reliable at reaching unseen goals compared to an End-to-End learning or a geometric mapping-based approach. Use of a model-based feedback controller allows LB-WayPtNav to successfully generalize from simulation to physical robots. In the future, it would be interesting to incorporate long-term spatial memory in LB-WayPtNav and to extend it to dynamic environments.

Acknowledgments. This research is supported in part by the DARPA Assured Autonomy program under agreement number FA8750-18-C-0101, by NSF under the CPS Frontier project VeHICaL project (1545126), by SRC under the CONIX Center, and by Berkeley Deep Drive.

References

[1] F. Alhwarin, A. Ferrein, and I. Scholl. IR stereo kinect: improving depth images by combining structured light with IR stereo. In *PRICAI*, 2014.

[2] B. Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[3] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[4] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 2015.

[5] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[6] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic slam. In *ICRA*, 2017.

[7] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 1991.

[8] 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 *ICRA*, 2017.

[9] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017.

[10] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee. Memory augmented control networks. In *ICLR*, 2018.

[11] D. K. Kim and T. Chen. Deep neural network for real-time autonomous indoor navigation. *arXiv preprint arXiv:1511.04668*, 2015.

[12] H. L. Chiang, A. Faust, M. Fiser, and A. Francis. Learning navigation behaviors end-to-end with AutoRL. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, 2019.

[13] L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *IROS*, 2017.

[14] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019.

[15] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. In *CoRL*, 2017.

[16] A. Faust, K. Oslund, O. Ramirez, A. Francis, et al. PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *ICRA*, 2018.

[17] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017.

[18] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018.

[19] E. Parisotto and R. Salakhutdinov. Neural Map: Structured memory for deep reinforcement learning. In *ICLR*, 2018.

[20] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *IROS*, 2017.

[21] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[22] F. Sadeghi and S. Levine. $(\text{CAD})^2$RL: Real single-image flight without a single real image. In *RSS*, 2017.

[23] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. *arXiv preprint arXiv:1902.03701*, 2019.

[24] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Robotics Research*, pages 325–341. Springer, 2018.

[25] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *CoRL*, 2017.

[26] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In *CoRL*, 2018.

[27] S. Jung, S. Hwang, H. Shin, and D. H. Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 2018.

[28] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[29] 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.

[30] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg. Vision-based high-speed driving with a deep dynamic observer. *IEEE Robotics and Automation Letters*, 2019.

[31] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *ICRA*, 2019.

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

[33] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Neural autonomous navigation with Riemannian motion policy. *arXiv preprint arXiv:1904.01762*, 2019.

[34] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

[35] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile off-road autonomous driving using end-to-end deep imitation learning. In *RSS*, 2018.

[36] R. Walambe, N. Agarwal, S. Kale, and V. Joshi. Optimal trajectory generation for car-type mobile robot using spline interpolation. *IFAC*, 49(1):601 – 606, 2016.

[37] D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Trans. Intelligent Transportation Systems*, 2016.

[38] D. J. Bender and A. J. Laub. The linear-quadratic optimal regulator for descriptor systems: discrete-time case. *Automatica*, 23(1):71–85, 1987.

[39] J. Van Den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *IJRR*, 2011.

[40] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. *arXiv preprint arXiv:1903.02531*, 2019.

[41] 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 *CVPR*, 2016.

[42] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[43] M. Zeiler, R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[44] M. Labbé and F. Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *JFR*, 2019.