APPEARED IN
Annual Conference on Robot Learning (CORL), 2017

Intention-Net: Integrating Planning and Deep Learning for Goal-Directed Autonomous Navigation

Wei Gao **David Hsu** **Wee Sun Lee**
School of Computing
National University of Singapore
{gaowei90, dyhsu, leews}@comp.nus.edu.sg

Shengmei Shen **Karthikk Subramanian**
Panasonic R&D Center Singapore
{shengmei.shen, karthikk.subramanian}@sg.panasonic.com

Abstract: How can a delivery robot navigate reliably to a destination in a new office building, with minimal prior information? To tackle this challenge, this paper introduces a two-level hierarchical approach, which integrates model-free deep learning and model-based path planning. At the low level, a neural-network motion controller, called the *intention-net*, is trained end-to-end to provide robust local navigation. The intention-net maps images from a single monocular camera and "intentions" directly to robot controls. At the high level, a path planner uses a crude map, *e.g.*, a 2-D floor plan, to compute a path from the robot's current location to the goal. The planned path provides intentions to the intention-net. Preliminary experiments suggest that the learned motion controller is robust against perceptual uncertainty and by integrating with a path planner, it generalizes effectively to new environments and goals.

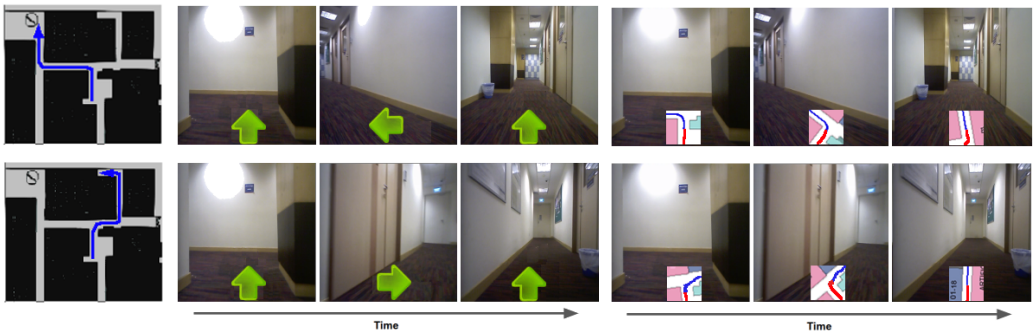
Keywords: Visual navigation, deep Learning, imitation learning, path planning

1 Introduction

Goal-directed, collision-free global navigation is an essential capability of autonomous robots. However, robots still do not quite match humans in such navigation tasks, despite decades of research and a vast literature [1, 2]. For example, upon arriving at a new shopping complex, a human can follow an abstract 2-D floor plan, devoid of most geometric and visual details of the 3-D world, and reach any location in the building, unimpeded by railings, glass walls, ... that often cause failures in robot navigation systems. Similarly, a human can drive to any destination in a new city by following a roadmap or GPS based navigation directions. There are two key elements to humans' performance: the ability to plan a path using a simplified abstract model of the 3-D world and more importantly, the ability to execute the planned path robustly using local visual information. To achieve comparable performance in robot navigation, we propose to integrate model-free deep learning for local collision avoidance and model-based path planning for goal-directed global navigation.

The idea of combining global path planning and local motion control is well studied in collision-free robot navigation [3, 4]. There are many path planning methods [5]. There are many motion control methods as well, *e.g.*, potential-field [6], visual servoing [7], ..., but they are usually manually designed. In recent years, deep learning has achieved extraordinary success in many domains, from image classification, speech recognition to game playing [8, 9, 10]. Deep learning has also found application in navigation tasks. Earlier work, however, usually trains the agent to move along a fixed path [11], with some notable exceptions [12].

Our proposed approach consists of a path planner at the high level and a motion controller, trained via deep learning, at the low level (Figure 1). The path planner computes a collision-free global path according to a crude input map, *e.g.*, a 2-D floor plan. The motion controller tracks this path, using images from a single monocular camera onboard the robot. It handles all local environment



(a) (b) (c)

Figure 1: A two-level navigation hierarchy, with a path planner at the top and the *intention-net*, a learned local motion controller, at the bottom. (a) Two paths planned for different goals in the same environment. (b) The intention as a discretized local move. (c) The intention as a local path and environment.

dynamics, including obstruction from static and dynamic obstacles not in the input map—such as furniture and people—and social conventions for navigation. The local motion controller is trained end-to-end via imitation learning. After training, the robot navigates to arbitrary goals in previously *unseen* environments. The proposed two-level architecture is more robust against perceptual uncertainty than traditional navigation systems, as the motion controller is learned from data. Further, by integrating with a path planner, it the learned motion controller generalizes to new environments and goals effectively.

We represent the local motion controller as a deep neural network F. Previous work trains F via imitation learning to follow roads and avoid obstacles, with impressive performance [11]. However, it implicitly assumes F to be a mapping directly from perceptual inputs to controls. The robot is thus not steerable according to high-level goals. For example, at an intersection, the decision to go straight or make a turn depends on not only the local perceptual input, but also the navigation goal. Mimicking the expert in the training dataset does not lead to the right action, as the goals may be different. Instead, we train a neural-network controller, called the *intention-net*, conditioned on both perceptual inputs and *intentions*. In our two-level architecture, the planned path at the top provides the intention for the motion controller at the bottom. The controller chooses the robot control conditioned on both the intention and the local perceptual input. Specifically, the *intention* may represent a subgoal along the path and the local environment at the robot's current location.

This work introduces a two-level hierarchical approach for robot navigation, with a path planner at the top and the intention-net, a motion controller learned from data, at the bottom (Section 3). The intention is the key ingredient that binds together the two levels of the hierarchy. Preliminary results in simulation and real-robot experiments suggest that the new approach enables a mobile robot to navigate reliably in a new environment with only a crude map (Section 4).

2 Related Work

Deep learning has been immensely successful in many domains [11, 12, 13, 9, 14, 15, 10]. In robot navigation, one use of deep learning is to learn a flight controller that maps perceptual inputs directly to control for local collision-free maneuver of a drone [15]. It addresses the issue of local collision avoidance, but not that of goal-directed global navigation. Another use is to train a system end-to-end for autonomous driving, using monocular camera images [11], but the system drives along a fixed route and cannot reach arbitrary goals. Some recent work explores model-free end-to-end learning for goal-directed navigation by incorporating the goal as part of the perceptual inputs [16, 17]. One may improve the learning efficiency and navigation performance by adding auxiliary objectives, such as local depth prediction and loop closure classification [18]. Without a model, these approaches cannot exploit the sequential decision nature of global navigation effectively and have difficulty in generalizing to complex new environments. To tackle this challenge, our proposed two-level architecture integrates model-free deep learning for local collision avoidance and model-based global path planning, using a crude map.

Hierarchies are crucial for coping with computational complexity, both in learning and in planning. Combining learning and planning in the same hierarchy is, however, less common. Kaelbling *et al.*proposed to learn composable models of parameterized skills with pre- and post-conditions so that a high-level planner can compose these skills to complete complex tasks [19]. Our hierarchical method shares similar thinking, but specializes to navigation tasks. It uses intention instead of general pre- and post-conditions to bind together hierarchical layers and achieves great efficiency.

Navigation is one of the most important robotic tasks. One classic approach consists of three steps: build a map of the environment through, *e.g.*, SLAM [20], plan a path using the map, and finally execute the path. High-fidelity geometric maps make it easier for path planning and execution. However, building such maps is time-consuming. Further, even small environmental changes may render them partially invalid. Alternatively, the optical flow approach does not use maps at all and relies on the visual perception of the local environment for navigation [21]. While this avoids the difficulty of building and maintaining accurate geometric maps, it is difficult to achieve effective goal-directed global navigation in complex geometric environments without maps. Our approach uses crude maps, *e.g.*, 2-D floor plans, and sits between the two extremes. Floor plans are widely available for many indoor environments. One may also sketch them by hand.

3 Integrated Planning and Learning with Intention-Net

Our proposed hierarchical method performs closed-loop planning and control. At each time step, the path planner at the high level replans a path from the robot's current position to the goal, using a crude floor-plan map. The path is processed to generate the "intention" for the intention-net motion controller. Given the intention and an image from a single monocular camera, the low-level controller computes the desired speed and steering angle for the robot to execute. The process then repeats. In this section, we first present the intention-net motion controller (Section 3.1) and the learning method (Section 3.2). We then briefly describe the path planner (Section 3.3).

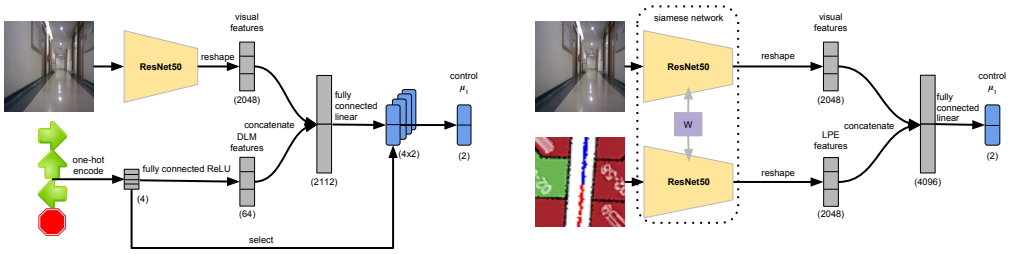
3.1 Intention-Net

Intention binds together global path planning and local motion control. Intuitively, intention captures what the path planner expects the motion controller to do at the robot's current position. We propose two definitions of intention. The simpler one mimics the instructions usually given for navigation: go straight, turn left, *etc.*. We call it *discretized local move* (DLM). DLM may take four discrete values: TURNLEFT, TURNRIGHT, GOFORWARD, and STOP. Given a path σ from the robot's current position x to the goal, we compute DLM by estimating the local signed curvature of σ at x. If the absolute value of the curvature is smaller than a chosen threshold, then the DLM is GOFORWARD. Otherwise, the DLM is either TURNLEFT or TURNRIGHT according to the sign of the curvature. The DLM is STOP if the robot reaches the goal. DLM is intuitive, but restricts the intention to four discrete values. This is clearly inadequate, for example, at an intersection with five or more branches. Our second definition, *local path and environment* (LPE), is richer. It is represented as a 224×224 image, containing both the path and the environment within a local window of the current robot position (Figure 2). See Figure 1c for additional examples.

Figure 2: The LPE intention represented as a 224×224 image. It captures information on the path that the robot has recently traversed (in red), the path ahead (in blue), the robot's current position (where the red and the blue paths meet), and the local geometric environment according to the map.

We represent the intention-net is a deep multivariate regression network F. At time step t, it takes as input a camera image X_t and the intention I_t, which is obtained from the path computed by the high-level planner. It outputs the robot control $\mu_t = (v_t, \theta_t)$, where v_t and θ_t are the robot speed and steering angle, respectively. Formally, $\mu_t = (v_t, \theta_t) = F(X_t, I_t)$.

Figure 3 provides a sketch of the network architectures for the DLM-net and the LPE-net. The DLM-net takes as input a camera image of size 224×224 and the DLM intention with four discrete values. It extracts from the image a feature vector of length 2048, using ResNet50 without the last fully connected layer. It encodes the intention first as a binary vector of length 4 and then expands it to a vector of length 64 in order to balance the influence of the visual and the intention feature



(a) (b)

Figure 3: Intention-net architectures. (a) DLM-net. The DLM intention takes four discrete values: TURN-RIGHT, GOFORWARD, TURNLEFT, and STOP. (b) LPE-net. The LPE intention is represented as a 224×224 image. Both networks also take a 224×224 camera image as the input.

vectors. The visual and the intention feature vectors are concatenated together as a single vector, which is then processed by a fully connected linear layer to generate the controls for all intentions. The DLM-net finally selects the control using the input intention.

The architecture for the LPE-net is even simpler conceptually. It takes as input a camera image and an intention image, both of size 224×224, and process them through a two-stream siamese network with shared weights. The intention image represents the environment according to the map and the robot path within a local window of the robot's current position. The camera image captures visual appearance of the 3-D world. The siamese network tries to reasoning about the spatial relationships between the two and embeds them in a space that preserves their spatial relationships. The siamese network outputs two feature vectors of length 2048. They are concatenated and processed by a fully connected linear layer to generate the final robot control.

3.2 Data Collection and Training

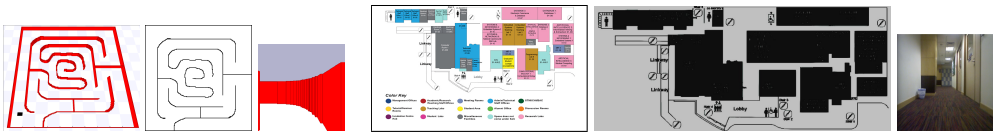
We collect data from demonstrations in the form of tuples (X_t, I_t, μ_t) for $t = 1, 2, \ldots$. For the simulation experiments, we run the path planner to generate the intention I_t and use the dynamic window approach [22] as the expert to generate the robot control μ_t. For the real-robot experiments, we collect data in a similar way, but use human experts. We run the path planner to generate the intention I_t and visualize it. The human expert controls the robot with a joystick according to the visualized intention. The collected dataset is split randomly into 4:1 ratio for training and evaluation.

We learn the desired velocity v_t and steering angle θ_t jointly and reweight their values so that $v_t, \theta_t \in [-1, 1]$, for all t. We use rmsprop optimizer with initial learning rate $1e-4$ and L2 weight regularization $1e-4$ to training our network. We anneal the learning rate over time by $\alpha = \alpha_0/(1+k)$, where k is the number of iterations and α_0 is the initial learning rate. We observe that training with smaller batch size dramatically increases the performance of the final regression. For practical use, we use a batch size of 8 in our experiments. We also find that using large training samples per epoch reduces the validation loss significantly in training. Specifically, we maintain a dataset pool that can be resampled at any time. We use $200,000$ uniformly sampled samples from the dataset pool in each epoch. Maintaining the dataset pool also makes it possible to create a large number of samples with a relatively small dataset.

3.3 Path Planning

The path planner replans a path from the robot's current position to the goal at every time step. It represents a crude map as an occupancy grid and performs adaptive Monte Carlo localization (AMCL) [23] to localize the robot and determine its current position with respect to the map. The planner searches the occupancy grid for the path using the hybrid A^* algorithm [24], which can accommodate robot kinematic and dynamic constraints.

The crude map may cause inaccuracies in both localization and path planning. Our hierarchical approach addresses the issue in two ways. First, the planned path is not executed directly. Its primary



(a) (b)

Figure 4: Experimental setup in simulation and real-robot experiments. (a) A simulated environment, the associated 2D map, and a camera view. (b) The floor plan for a robot experiment environment, the occupancy grid map computed from the floor plan, and a camera view of the environment.

purpose is to generate the intention. The intention-net motion controller combines the intention with robot's local perceptual input to generate the controls for execution. By training on suitable datasets, intent-net learns to be robust against some of these inaccuracies. Second, replanning at each time step further improves robustness.

4 Experiments

We evaluated our approach in both simulation and real-robot experiments. For simulation, we used the Stage simulator [25] as the experimental testbed. We hand-sketched a set of 2-D maps and used Stage to generate the corresponding 3-D environments for training and testing the intention-net. See Figure 4a for an example. For the real-robot experiment, we trained the robot on the first floor of our office building and tested it on both the first floor and the second floor, which differs significantly from the first floor geometrically and visually (Figure 7). We digitized the visitor floor plans of our building to create crude occupancy-grid maps (Figure 4b). Both the simulation and real-robot experiments used a Pioneer 3-DX robot. In real-robot experiments, the robot is equipped with a webcam that has 70° field of view. It is also equipped with a Sick LMS-200 LIDAR, used for localization only.

For comparison, we consider three alternative methods: Path Tracker, Non-Intention Net, and Dynamic Window [22]. The first two are ablated versions of our approach. Path Tracker replaces the intention-net with a standard motion controller and follows the planned path without visual feedback from the camera. In contrast, Non-Intention Net removes the intention as an input to the neural-network controller and relies on visual feedback only. It removes the path planner as well. Dynamic Window is a well-established successful method for robot navigation.

4.1 Results from Simulation Experiments

In simulation experiments, our main objective is to compare the two intention-net methods, DLM-Net and LPE-Net, with the three alternative methods(Table 1), according to four measures: success in task completion, the number of human interventions, total task completion time, and robot motion smoothness. *Smoothness* is defined as the average *jerk*, *i.e.*, the change in acceleration, of the path that the robot traverses.

We consider five tasks (Table 1 and Figure 5). Tasks A–C use environments present in the training dataset and test for generalization to new goals. Tasks D–E test for generalization to new environments unseen before. The geometric complexity of the environment increases from task A to E, roughly. Task A has a simple environment with multi-way junctions. The environment for task E is a maze.

Overall LPE-Net performs better than DLM-Net, especially in task D and E, which involve generalization to new environments. Both substantially outperform the alternatives. They complete all tasks with no human interventions. Path Tracker completes the task faster than others in the simple task, task A. However, it generally produces robot motions that are jerky and not smooth. This is quite visible in Figure 5. It also requires a large number of human interventions in the most difficult task, task E. Non-Intention Net completes task A, but fails in all others. This is unsurprising. By removing the intention input, Non-Intention Net has no notion of global navigation goals. It blindly mimics

Table 1: Performance comparison on five navigation tasks in simulation. Each task requires the robot to navigate through a sequence of goals. See Figure 5.

Task	Method	Success	Intervention	Time (sec.)	Smoothness
	DLM-Net	Y	0	113	0.0053
	LPE-Net	Y	0	114	0.0053
A	Path Tracker	Y	0	**105**	0.0040
	Non-Intention Net	Y	0	112	0.0053
	Dynamic Window	Y	0	109	**0.0039**
	DLM-Net	Y	0	**118**	0.0074
	LPE-Net	Y	0	128	**0.0068**
B	Path Tracker	Y	3	155	0.0170
	Non-Intention Net	N	–	–	–
	Dynamic Window	Y	0	126	0.0100
	DLM-Net	Y	0	**559**	0.0074
	LPE-Net	Y	0	561	**0.0072**
C	Path Tracker	Y	16	640	0.0150
	Non-Intention Net	N	–	–	–
	Dynamic Window	Y	0	565	0.0094
	DLM-Net	Y	0	237	0.0085
	LPE-Net	Y	0	**226**	**0.0066**
D	Path Tracker	Y	5	240	0.0120
	Non-Intention Net	N	–	–	–
	Dynamic Window	Y	0	238	0.0095
	DLM-Net	Y	0	545	0.0080
	LPE-Net	Y	0	**531**	**0.0075**
E	Path Tracker	Y	21	546	0.0089
	Non-Intention Net	N	–	–	–
	Dynamic Window	Y	0	551	0.0084

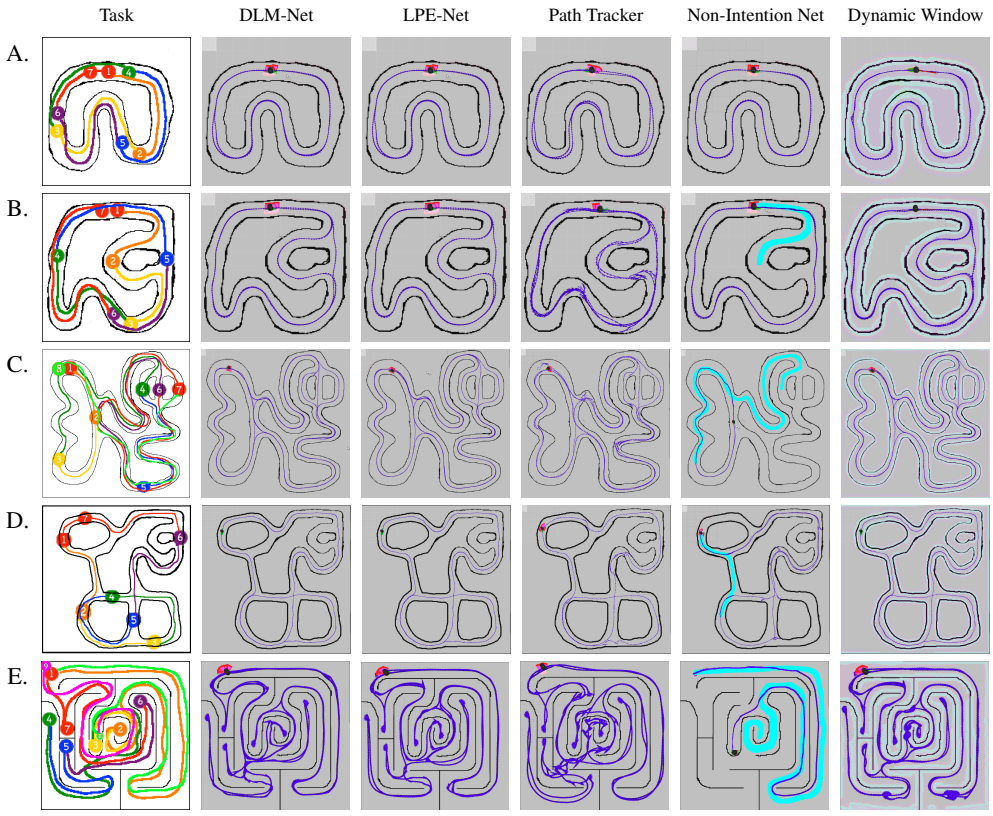


Figure 5: Example runs of five navigation tasks in simulation. Each task requires the robot to navigate through a sequence of goals $1, 2, \ldots$. The first column shows the task and the planned path. The remaining columns show the execution traces (thin blue lines) of five methods. For Non-Intention Net, we overlay the planned paths (thick light-blue lines) for the failed tasks. For tasks B, D, and E, it fails when traveling from goal 1 to 2. For task C, it fails when traveling from goal 3 to 4.

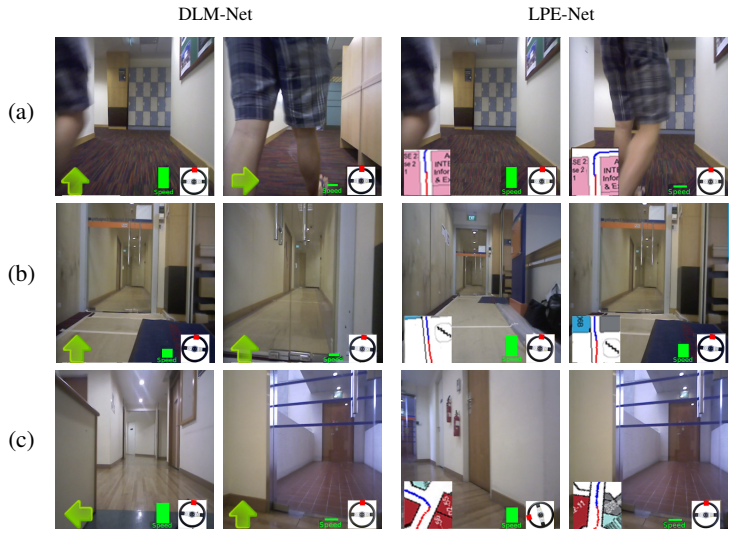


Figure 6: Examples from real-robot experiments. (a) The robots slow down to let a human blocking its path pass by. (b) The robot encounters a glass door in the training dataset. (c) The robot encounters a different glass door not in the training dataset. The robot detects the glass door and slows down to a stop, as indicated by the speedometer at the lower right corner of each image.

the expert, without recognizing that given the same visual feedback, the expert may have different goals in the training dataset. It succeeds in task A, because this task has a very simple environment, with no junction to separate paths for different goals. LPE-Net and DLM-Net also outperform Dynamic Window in terms of task completion time and robot motion smoothness, especially in the more complex environments (tasks C–E).

4.2 Results from Real-Robot Experiments

Next, we evaluate the performance of our approach, when faced with the complexity of the real world. We highlight several interesting scenarios below:

- *Moving people.* The crude floor-plan maps obviously do not contain information on dynamic obstacles, such as moving people. Nevertheless, our approach can handle people obstructing the robot path, thanks to the intention-net learned from data. The robot recognizes people, men or women dressed in different types of clothing. If a person moves in the same direction ahead of the robot, the robot usually slows down and follows the people. If a person moves in the opposite direction, the robot stops and waits for the people to pass by. See Figure 6a for an example.
- *Glass doors.* Transparent obstacles such as glass doors present a well-known difficulty for navigation systems that rely on LIDAR. Using images from a single monocular webcam, the intention-net enables our robot to handle glass doors. Figure 6b shows the glass door in the training dataset. Figure 6c shows the robot behavior when it encounters a different glass door not present in the dataset. Even more interestingly, the floor-plan map is inaccurate and does not contain the glass door. So the high-level path planner is not aware of it. The intention-net nevertheless detects the glass door and responds to it correctly by slowing down the robot to a full stop.
- *New environments.* The robot is trained on data from the first door of our office building only. It navigates very capable on the second floor, which differs significantly from the first floor in geometric arrangement and visual appearance (Figure 7), though more thorough evaluation is required to quantify the performance.

(a) (b)

Figure 7: Generalization to a new environment. (a) The first floor of the building in robot experiments. The robot is trained on data from the first floor only. (b) The second floor, used for testing the robot, has a significantly different geometric arrangement and visual appearance.

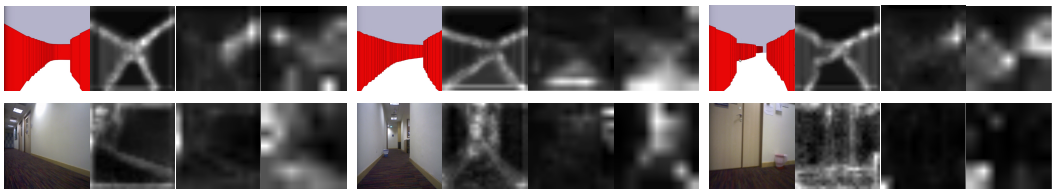


Figure 8: Visualization of the learned feature in simulation and real-robot experiments. Each block shows the camera view and the activated features from the 9th, 18th, and 50th layers of ResNet50.

Both DLM-Net and LPE-Net are evaluated in our experiments. Overall they both perform well, but LPE-Net usually outperforms DLM-Net when difficult maneuvers, such as sharp turns, are required. DLM-Net restricts the intention to four discrete values based on a manually chosen threshold. While it is possible to learn the threshold from data, we believe that LPE-Net, which uses a richer intention representation, is a better and more systematic way to proceed.

The most common failure of our approach occurs when the planned path generates the wrong intention as a result of inaccurate localization, especially with DLM-Net. For example, the robot approaches the end of a hallway and must turn left. Intuitively, the TURNLEFT intention is required. Instead, the GOFORWARD is generated because of the localization error. The robot may then crash against the wall. This happens because the simple A^* path planner does not account for localization errors and relies solely on replanning to close the loop. A more sophisticated planner, which hedges against uncertainty in localization, would alleviate this difficulty.

4.3 Visualization of Learned Features

To gain some understanding of how the intention-net processes the visual feedback, we visualize the learned features by projecting all activated features of a layer on an image. Figure 8 shows some examples. The lower-layer features are more concrete and capture the lines where the wall meets the floor or the ceiling. The is clear, especially in the simulation setting. The higher-layer features are more abstract and difficult to interpret. Sometimes they capture the free space for navigation, *e.g.*, the lower-left block of Figure 8.

5 Conclusion

We propose a two-level hierarchical approach that integrates model-free deep learning and model-based path planning for reliable robot navigation. Learning allows us to train the intention-net, a neural-network motion controller, to be robust against robot perceptual noise and localization errors. Planning enables the learned controller to generalize to new environments and goals effectively. Together they enable our robot to navigate almost immediately in a new indoor environment, with a very crude map, such as a digitized visitor floor plan.

One main limitation of our current approach is robot localization. It treats localization as a black box and segregates it from the planner. Integrating localization with planning and learning will be the next challenge in our work.

Acknowledgments

This research is supported in part by Singapore Ministry of Education AcRF grant MOE2016-T2-2-068 and Singapore-MIT Alliance for Research & Technology IRG grant R-252-000-655-592. We thank Bin Zhou from Panasonic R&D Center Singapore for helping with some experiments.

References

[1] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *J. Intelligent & Robotic Systems*, 53(3):263, 2008.

[2] N. Fallah, I. Apostolopoulos, K. Bekris, and E. Folmer. Indoor human navigation systems: A survey. *Interacting with Computers*, 25(1):21–33, 2013.

[3] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2002.

[4] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 24(2):237–267, 2002.

[5] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[6] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 1991.

[7] N. J. Cowan, J. D. Weingarten, and D. E. Koditschek. Visual servoing via navigation functions. *IEEE Trans. on Robotics*, 18(4):521–533, 2002.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, 2016.

[9] G. Hinton et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] D. Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[11] M. Bojarski et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[12] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proc. IEEE Int. Conf. on Computer Vision*, 2015.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[14] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *J. Machine Learning Research*, 17(39):1–40, 2016.

[15] F. Sadeghi and S. Levine. (CAD) ^{2} RL: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[16] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. *arXiv preprint arXiv:1609.07910*, 2016.

[17] 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. *arXiv preprint arXiv:1609.05143*, 2016.

[18] P. Mirowski et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[19] L. P. Kaelbling and T. Lozano-Pérez. Learning composable models of parameterized skills. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2017.

[20] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó. The SLAM problem: A survey. *CCIA*, 184 (1):363–371, 2008.

[21] H. Chao, Y. Gu, and M. Napolitano. A survey of optical flow techniques for robotics navigation applications. *J. Intelligent & Robotic Systems*, 73(1-4):361, 2014.

[22] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[23] D. Fox. KLD-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems*, 2001.

[24] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robotics Research*, 29(5):485–501, 2010.

[25] R. T. Vaughan. Stage: A multiple robot simulator. Technical report, Technical Report IRIS-00-394, Institute for Robotics and Intelligent Systems, University of Southern California, 2000.