Topometric Imitation Learning For Route Following Under Appearance Change

Shaojun Cai
UISEE Technology Inc.
85 Hongan Road, Fangshan District, Beijing, China
shaojun.cai@uisee.com

Yingjia Wan
Key Laboratory of Behavioral Sciences, Institute of Psychology, Chinese Academy of Sciences
16 Lincui Road, Chaoyang District, Beijing, China
wanyj@psych.ac.cn

Abstract

Traditional navigation models in autonomous driving rely heavily on metric maps, which severely limits their application in large scale environments. In this paper, we introduce a two-level navigation architecture that contains a topological-metric memory structure and a deep image-based controller. The hybrid memory extracts visual features at each location point with a deep convolutional neural network, and stores information about local driving commands at each location point based on metric information estimated from ego-motion information. The topological-metric memory is seamlessly integrated with a conditional imitation learning controller through the navigational commands that drives the vehicle between different vertices without collision. We test the whole system in teach-and-repeat experiments in an urban driving simulator. Results show that after being trained in a separate environment, the system could quickly adapt to novel environments with a single teach trial and follow route successively under various illumination and weather conditions.

1. Introduction

In recent years, spatial navigation has received widespread interest from the areas of artificial intelligence and cognitive neuroscience. Inspired by the cognitive models of animals and humans in navigation tasks, researchers have proposed a number of navigation models for autonomous agents, and the performance of many nicely replicated the results of some classic behavioral experiments in maze environments [2]. However, navigation remains a critical challenge for real-world robotic applications. The complex mechanical control and the rapidly changing environments make it difficult and costly for autonomous agents to navigate in real-world contexts.

In the area of autonomous driving, the majority of existing navigation models rely heavily on a metric map for localization and path planning. Most of the traditional visual navigation systems first build a geometric map of the environment with SLAM techniques and then perform feature matching with the map to compute the camera pose. After that, the system calculates a feasible path accordingly and execute the plan with control modules. However, SLAM suffers from several problems. Firstly, visual SLAM is based on local feature matching which tends to fail due to weather and illumination changes commonly found in outdoor spaces [20] (Figure 1 and Figure 2). Secondly, precise reconstruction of every geometric element could result in an overwhelmingly large map that contains much redundant information about task-irrelevant elements, such as details of stores on the sides of the roads. Lastly, the heavy reliance on a precise geometric map makes it difficult to apply SLAM-based systems to large-scale environments, as the amount of calculation would be proportionate to the area of the environment.

Compared to the commonly used navigation systems in autonomous driving which rely on a detailed metric map and a planner, mammal's navigation system is more similar to a combination of a topological map for global planning and a stimulus-response system for local control [23]. Instead of having an accurate map of the entire environment, mammals store information about key location points, such as certain landmarks and intersections. When traveling in the environment, they match their perceptions to these key points to localize themselves and plan paths toward goals, and then travel from one point to the next by simply following the planned path. Similarly, navigation systems for autonomous driving could build a topological map to represent the environment with vertices – nodes that correspond

(a) Clear Noon (b) After rain noon (c) Cloudy after rain (d) Heavy rain (e) Clear sunset (f) Soft rain sunset

Figure 1: Sample images under various weather and illumination conditions in town01 and town02. The first row represents images in town01, and the second row represents images in town02. Each column shows a different weather condition.

(a) Clear sunset (b) Cloudy sunset (c) Hard rain noon (d) Mid rain sunset

Figure 2: Sample images in town03 and town05. The first row represents images in town03, and the second row represents images in town05. Both environments are more complex than town01 and town02, with multi-way lanes and irregular intersections. It can also be spotted that there are roundabouts and slopes in town03.

to locations in the environment, use feature matching to locate the vehicle, and travel from one vertex to the next with local control modules.

Recent work started developing navigation systems using topological representation [24, 21, 6]. However, this line of work targets only the high-level navigation problem, rather than issuing steering commands to the low-level controller. Inspired by the development in deep learning, recent work [4] proposed to directly map the visual perception to a controller output with an end-to-end neural network. In an effort to combine high-level command and local control, [15, 9] proposed to use an additional navigational command alongside the image to compute the final control. However, this system requires a separate external mapping and planning module to generate the navigational command. In other words, a precise map is still required, and the limitations of building a metric map are still unsolved.

In order to develop a complete visual navigation system, a map representation that is suitable for downstream control task is required so that the high-level command and the low-level control can be integrated seamlessly. In this paper, we present a two-level navigation architecture for autonomous driving that uses topological-metric memory representation for high-level commands and conditional imitation learning for local control [15]. The map stores high-level navigational commands at the graph vertices, which are then

passed to the local controller. The mapping system first constructs a topological graph G with vertices that correspond with the places in the mapping process, and then uses the convolutional layer output of ResNet50 [17] as the embedded scene signature for each place. The system also stores approximate distances between different vertices based on the ego-motion information, and generates the navigational command which is either a discrete value or a top-down view of the local path at each vertex. After constructing the map, the navigation system uses place recognition technique to localize the vertex and retrieves the navigational command stored in that vertex. Since the place recognition result could produce error due to repetitive appearances of the environment, and the navigation trial might not overlap completely with the mapping trial, the learning-based local controller is designed to handle the uncertainty inherent to the navigational command. Although conditional imitation learning has certain degree of resistance to inaccurate navigational command, we further account for the errors in local path by adding permutations to the data in the training process.

We run all the experiments in an autonomous driving simulator Carla [12] that provides realistic urban driving environments. We first train the retrieval and controller networks in a separate training environment, and evaluate the mapping and navigation process using teach-and-repeat paradigm in a new environment with notably different road structures and features. During the evaluation phase, the vehicle is spawned in a random position and given one demonstration trial with the images, controls, and ego-motions recorded. Then the vehicle is reinitialized at the same starting point and asked to follow the same route to the endpoint under a variety of weather and illumination conditions. Preliminary results show that this model performs well in complex urban environments despite the appearance differences between the teaching trial and repeating trials.

2. Related Works

Traditional navigation methods in autonomous driving generally rely on metric map representation for both high-level planning and low-level control. In particular, SLAM-based methods are gaining increasing popularity in recent years [7, 22]. Feature-based SLAM method performs feature matching to construct the sparse 3D point cloud as map representation. Direct SLAM [13] performs direct pixel-level matching and builds a semi-dense representation of the environment. Both types of SLAM methods build a detailed metric map of the environment, and the map construction process is prone to errors in feature matching caused by weather and illumination changes.

In order to reduce reliance on precise mapping and localizaton systems, recent works [4, 18, 8] have started to utilize information perceived locally from sensors such as cameras to compute controller outputs such as steer, gas, and brake. A particularly popular approach is imitation learning [4], in which the system learns from expert demonstrations. Imitation learning is generally considered model-free, as it directly maps the visual perception to a controller output with an end-to-end neural network rather than conducting planning or model-based reasoning. [9] proposed to augment the purely reactive imitation learning method with a high-level navigational command. [15] changes the conditional command to a more general top-down view representation. Rather than learning control from first-person images, [3, 28] proposes to learn the control from a bird-eye view of an abstract scene representation. However, the above methods either rely on a metric map or leave open the issue of map representation.

Unlike SLAM-based method, humans and animals do not rely on metric representations of the environment for global navigation and local control [16, 26, 14]. While mammals also use a cognitive map for high-level planning[25], the map contains relatively abstract knowledge of the spatial layout. While the exact navigation strategy of mammals is still under debate, it appears to rely on information such as landmarks and connections between different locations [14], suggesting that their representation of the environment is mostly topological.

Topological maps have been applied to navigation models for decades, especially in 2D environments [5, 11, 20]. Recently, [24] designed a topological memory model that uses deep neural network to retrieve information about vertices based on observations. However, their work focuses on the indoor environments rather than the large-scale outdoor scenarios used in autonomous driving setting. [6] performs topological map construct in a large environment with a single traversal, but it mainly focuses on high-level navigation task and does not address the control issue. To successfully apply topological maps, the robot has to associate the abstract places and paths with physical places and paths perceived locally [20]. In contrast, metric map stores rich geometric and navigational information in the map itself, and thus lowers the burden for the perception module. However, building a globally consistent metric map is difficult and only feasible in small areas. For this reason, a combination of topological and metric map, or topometric map [10, 1], is a good balance. Topometric maps only contains locally consistent metric information and is thus more applicable.

3. Methodology

Our method can be separated into two modules: topological localization and local control. The topological localization module is trained with a metric learning paradigm, and the local control is trained with conditional imitation learning paradigm. Both modules are trained in a separate en-

vironment different from the test environment, and without any human labeling effort.

We test our method in a teach-and-repeat paradigm. In the testing environment, we first conduct a teach trial that navigates from a specified start point to an endpoint with ground truth localization and a planning system. After that, we construct the topology graph and save the images and navigational commands at each topological node. There are two types of navigational commands, the Discrete Local Move (**DLM**) and Local Path (**LP**), and the type of command stored in the topological map is different according to the chosen controller model that are described. The details of the two types of navigational commands are described in the following sections. In the repeat trial, the agent conducts topology localization by matching the convolutional feature of the current image and a subset of vertices in the graph. After that, the agent retrieves the navigational command from the matched vertex and performs conditional imitation learning to drive the vehicle forward.

3.1. Discrete Local Move Command

We trained the model using conditional imitation learning – a variant of imitation learning that allows providing high-level commands to a learning-based controller model. When coupled with a high-level topological planner, the method can scale to complex navigation tasks such as driving in an urban environment. We briefly review the approach here and refer the readers to [9, 15] for further details. [15] proposes two kinds of representations of the high-level commands. The DLM-net represents the intention with four discrete values: follow lane, left, right, straight.

3.2. Local Path Command

The above discrete commands only apply to simple intersections. In order to handle more complicated intersections, e.g. roundabout, we adopt a more general command representation as in [15]. Specifically, we first compute the ego-motion of the vehicle using the provided velocity and timestamps from the simulator. Afterward, we draw the coordinates of the positions in a local window of 224x224 pixels in the local coordinate system. Then the image of the local path is concatenated with the first-person image and passed further through two fully-connected layers to generate the final control command.

We made two modifications to the original implementation in [15]. First, the original local path representation is drawn on top of a local environment sketch. Since we do not assume any form of prior map of the environment , we draw the local path on a blank background instead. Second, the original network is implemented in a siamese network style. However, in the experiment we find that the non-siamese network produces better results in our setting. A possible explanation is that although the siamese network tries to embed the two inputs into a shared space, it also forces the rich first-person image to lose a certain amount of information which might be essential to the control output.

In the experiment, we also find that simply training the network in a non-siamese fashion could lead to severe overfitting to the local path image. In the extreme case, the network would just completely ignore the visual input and simply follow the direction in the Local Path. Therefore, it is important to account for the possible localization error in the retrieval process. We augment the training local paths by randomly selecting a path centered around a perturbated location with proximity to the true vehicle position. In addition, we add a small amount of random rotation to each local path to simulate the possible deviation from the path in the repeat process.

3.3. Visual Place Recognition

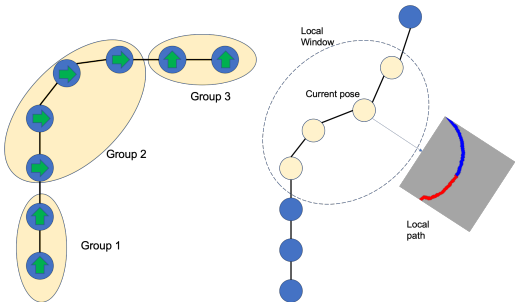
Visual Place Recognition [20], or topological localization, determines the most likely place of the vehicle based on the current camera input. Unlike metric localization that computes the metric location in a global coordinate system, place recognition aims at locating the most likely vertex in the topological map. Compared to the local feature matching and geometric pose estimation in SLAM systems, place recognition has a larger localization error as it does not compute the accurate pose. However, global features used in the place recognition process are more robust to changing lighting conditions than local features used in the SLAM method [20], and therefore has a higher overall success rate.

We implement the place recognition algorithm with the siamese network [27]. The network aims at projecting the input images into a common feature space, where images spatially close to each other in the map are also close to each other in the embedding space. To achieve this goal, the network extracts visual features from a pair of images using shared weights, and features are further processed by fully-connected layers to determine whether they are close or not.

Specifically, we employ the ResNet50 [17] as our base network and adopts the 2048 dimensional feature of the average pooling layer as the embedded feature vector. The feature vectors of the two images are concatenated and further processed by two fully-connected layers and a softmax function to produce the final output. The network is trained with cross-entropy loss.

3.4. Topological Map Construction

Upon finishing the teaching trial, we construct a topological node at each position on the path. We store the approximate positions of each vertex based on the estimation using path integration on the velocity input from the simulator with additional Gaussian noise. The ego-motion could



(a) Topo-DLM map (b) Topo-LP Map

Figure 3: We group the vertices in Topo-DLM Map according to their values. Every time, the agent only needs to locate itself to the correct group, rather than to the exact vertex. The Topo-LP map maintains a local window around its current position and searches in the window.

also be calculated from the visual odometry, but we leave this for future work. After determining the places of the vertices, we store the visual features and the corresponding navigational commands along with the vertices.

3.5. Visual Navigation Pipeline

After finishing the teaching trial and having constructed the topological graph, in the repeat trial the system first searches for the vertex that matches the current image the most and retrieves the stored navigational command. After that, the system performs the conditional imitation learning pipeline according to the stored command.

We perform a different type of topological localization strategy for each navigational command. For the DLM command, we group the topological nodes belonging to the same command, as shown in Figure 3a. Upon localized to a certain node, we use the retrieved discrete command for further processing in the DLM-net. Otherwise, we simply set the conditional command as 2 (Go Straight).

For the LP command, we perform a more sophisticated localization procedure. We conduct image retrieval with a fixed frequency in the neighborhood of the last estimated pose (shown in Figure 3b, and the retrieval is deemed successful only if the score predicted by the retrieval network on the current image and the retrieved image is higher than 0.8. If the retrieval is not performed or unsuccessful at the current timestamp, we use path integration from the velocity to compute the estimated position on the topometric graph. Due to the noise in the velocity, the integrated position could drift, and we set the estimated pose to the retrieved pose if successful. If there is no successful retrieval in the past 30 timestamps, we perform global relocalization that searches all the vertices to relocalize the vehicle. During the relocalization procedure, we consider the recent five images and compare them independently against all the candidate images. The result positions of the images should not be too far away from each other, so we only consider the localization successful if the max and min of the result positions are within 30 vertices on the topometric graph. During the global relocalization procedure, we deliberately slow down the agent as the global search takes more time than a window-based search. The full navigation pipelines of the two methods are shown in Figure 4 and Figure 5.

4. Experiments

We perform experiments using Carla [12], a driving simulator that renders realistic outdoor urban environments. Unlike the indoor environments used in other works [24], this simulated outdoor environment is much larger in scale and highly structured. In addition, it contains various weather and illumination conditions.

We use two versions of Carla simulation environment, Carla 0.8.2 and Carla 0.9.7. Carla 0.8.2 contains two town environments. Carla 0.9.7 contains seven environments, two of which are the same as the two towns in Carla 0.8.2. Carla 0.8.2 contains a planner that provides high-level discrete commands at intersections, which is suitable to test the DLM agents.

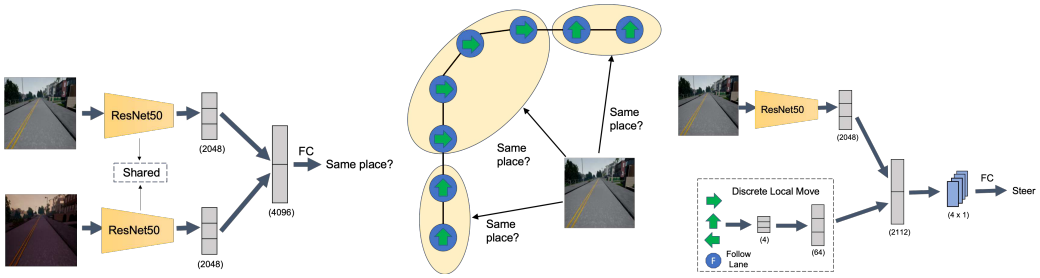
Town01 and *town02* are regular urban environments, with two-way lanes and buildings on the sides, as shown in Figure 1. *town03* and *town05* are more complex urban environment with irregular intersections and multi-way lanes, as shown in Figure 2. *town03* contains roundabouts and tunnels which are not present in *town05* We deliberately pick environment *town03* to examine the performance of LP command when navigating through complex structures.

We conduct two sets of experiments. In the first set, we train all the networks of the model separately in *town01* and test the full model in *town02*. In the second set, we train all the networks in *town05* and test the model in *town03*. The two environments in each train-test pair are different in appearance but contain similar structures.

For different experiments, we pick different combinations of weather conditions. For town01 and town02, we choose six weather conditions, including rain, sunset, etc. For town03 and town05, we choose five different weather conditions.

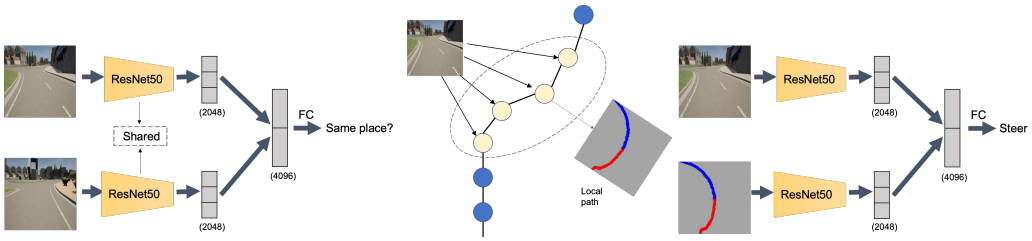
We evaluate the performance with a teach-and-repeat paradigm. Although teach-and-repeat is a special case of navigation, it can properly demonstrate the essential abilities of our system, such as topological-metric map construction, self-localization, etc. To extend the experiment to a more general goal-based navigation, we adopt the place recognition network to associate the goal image with a vertex, and conduct A* search from the start to the goal.

To demonstrate the generalizability of our method, we train the place recognition network, intersection classification network, and the controller network only in *town01* and



(a) Similarity Comparison (b) Localization (c) DLM controller

Figure 4: The visual navigation pipeline for Topo-DLM model.



(a) Similarity Comparison (b) Localization (c) LP controller

Figure 5: The visual navigation pipeline for Topo-LP model.

town05, and perform route following in *town02* and *town03*.

(a) Camera image (b) Local path image

Figure 6: Sample image and generated local path in town03 at the middle of the roundabout. Local path representation is flexible and able to represent more general road structure such as roundabout, whereas discrete command could only represent regular intersection.

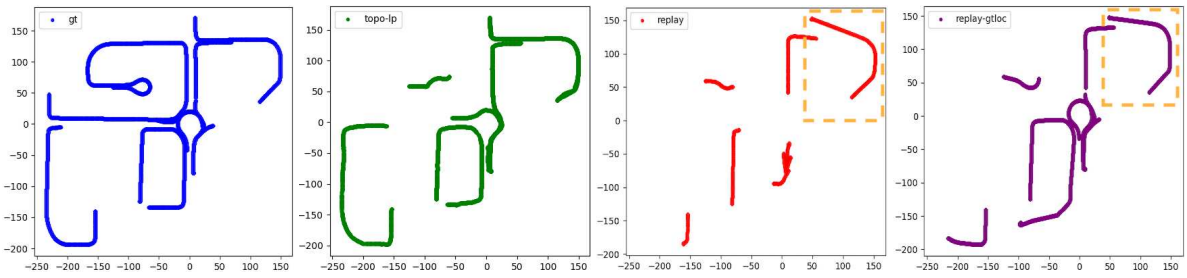
4.1. Training Details

We train the control network and the retrieval network in *town01* and *town05*. In *town01* we use the dataset provided by [9]. The dataset contains two hours of human driving data in *town01*, 10% of which contains injected noise (i.e., intentionally deviates from the optimal path). The label of each image contains the steering angle, throttle and brake value, positions, rotations, and the high-level command. In *town05*, we collect the training data with the planning-based controller provided by the simulator. As in the dataset from *town01*, we also inject noise in about 10% of the data. We record the images along with steering angles, locations, rotations. It is important to note that for the noisy data points, we record the steering angle computed by the controller that brings the vehicle to its correct path. The noise injection is critical to train a stable controller. The steering angle is within the range [-1, 1].

In *town01*, we use the discrete commands provided by the dataset. In *town03* we generate the local path in a local window around the current position of the robot. We convert the coordinates in the local window to the local coordinate of the current position, and draw the local coordinates in the background, with the current position at the center. Sample local path images are shown in Figure 6. To avoid overfitting to the local path, we add random noise to the robot position and generate the local path at the perturbated position.

To train the retrieval network, we sample image pairs with a distance less than 2.0 meters as positive pair, and sample image pairs with a distance greater than 2.0 meters as negative pairs. To make the network robust for images across different weather and illumination conditions, we sample the positive pairs across various weather and illumination conditions in the training data.

We use a batch size of 32 in the training process for both networks. We perform gradient update using RAdam [19]. The learning rate is initially set to 0.0001 and dynamically



(a) Demonstraion (b) Topo-LP (c) Replay (d) Replay(gt loc)

Figure 7: Display of all nine trajectories of different methods under the weather of clear sunset in town03 (units of the axes are meters). Topo-LP completes more trajectories than replay agents with or without ground truth locations. It is especially noteworthy that both the replay and replay(gt loc) agents drift away due to open loop control, as shown in the orange rectangle at the top right corner.

decreases as the training proceeds.

4.2. Measurements

We measure the performance of each method by its success rate in the repeat trials. A trial is considered successful when the vehicle reaches the goal within a certain distance. We calculates the average success rate for all the lighting conditions, as well as the success rate under each individual lighting condition.

Carla simulator provides a benchmark system that could test the agent navigation performance in a set of pre-defined routes and weather conditions. After finishing all the routes, the benchmark system measures performance results with a set of criteria.

4.3. Baselines

We compare both the Topo-DLM and Topo-LP methods to the baseline action replay agent. Upon reaching a vertex, the agent reads the target speed and target steer stored in the vertex. It then maintains the speed as close to the target speed as possible, and executes the target steer accordingly. We test two variants: one uses image retrieval to get the vertex, and the other uses ground truth localization to get the vertex. The velocity of the agent has a gaussian distribution with mean = 0.0 and deviation = 0.1.

5. Results

Results are shown in Table 1 and Table 2. As the table shows, simple action replay has the worst performance. In *town02*, the replay agent, with or without ground truth, has no successful trials, while in *town03* the replay agent with ground truth location achieves a success rate of 0.5 and the replay agent without ground truth location achieves a success rate of 0.25. The reason for the especially poor performance in *town02* could be that the narrow roads in town02 lead to more crashes when the vehicle drifts. As shown in Figure 2, the roads in town03 are much wider, which reduces the chances for the agent to collide to the road structure on the sides. The generally poor performance of the replay agent, with or without ground truth location, is because it is open-loop without any correction, so any localization or control error could lead to the failure of the entire navigational process.

Both the Topo-DLM and the Topo-LP Agent outperforms the replay agent by a large margin. There are several reasons to explain the robustness of our method. Firstly, the network aims at computing an action conditioned on the discrete local move or the local path, rather than completely replay the path. Secondly, in the training process, we inject noise that takes into account the possible pattern of localization error, so that the network could learn to balance the information between the navigational command and the camera image. Thirdly, we adaptively adjust the speed according to the localization accuracy. When the vehicle is lost and conducting global relocalization, it slows down until relocalization is successful.

We also show detailed performance under different lighting conditions for each method. We can see that Topo-LP and Topo-DLM consistently outperform replay in all cases, and perform slightly worse than Topo-LP and Topo-DLM with ground truth location in most cases. The example trajectories of different methods under the clear sunset weather in town03 are presented in Figure 7.

It is worth noting that in relatively regular environments *town02*, Topo-DLM agent works better than Topo-LP agent. However, in the more complex roundabout environment in *town03*, discrete values are not sufficient to describe the navigational command, and therefore Topo-LP is more appropriate and appears to perform well.

The result suggests a trade-off here. On the one end, if the structure of the environment is simple enough, such as a straight road, no navigational command is needed for the vehicle to drive forward. On the other end, if the envi-

	Mean	Clear Noon	After rain noon	After rain cloudy	Heavy rain	Clear sunset	Soft rain sunset
Replay (gt loc)	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Topo-DLM	0.65	0.8	0.3	0.9	0.6	0.8	0.5
Topo-DLM (gt loc)	0.667	1.0	0.3	0.8	0.7	0.8	0.4
Topo-LP	0.567	1.0	0.6	0.5	0.6	0.4	0.3
Topo-LP (gt loc)	0.817	1.0	1.0	1.0	0.9	0.3	0.7

Table 1: Results of town02

	Mean	Clear sunset	Cloudy sunset	Hard rain noon	Mid rain sunset
Topo-LP	0.722	0.778	0.667	0.667	0.778
Topo-LP (gt loc)	0.806	0.556	0.889	0.889	0.889
Replay	0.25	0.222	0.444	0.222	0.111
Replay (gt loc)	0.5	0.444	0.667	0.444	0.444

Table 2: Results of town03

ronment is complex, such as a roundabout, a sophisticated navigational command is essential.

6. Conclusion

This paper proposes a two-level navigation method that integrates a model-based planning module and a model-free local control module. The planning module builds a topological representation of the environment by storing information about location and direction in vertices. The control module uses imitation learning to train the vehicle to travel between vertices without collision.

This method can be generalized to novel environments and perform robustly in various weathers and illuminations. Results show that the proposed combination of topological representation and conditional imitation learning could allow vehicles to navigate in complex, novel urban environments without a metric map.

Currently, our method focuses only on static environments without considering other active agents on the road that could potentially affect both place recognition and local control performance. In addition, the need to avoid other agents might lead to inconsistency between local control and global navigational command, which would be an issue worth addressing in future research.

References

[1] Hernán Badino, Daniel Huber, and Takeo Kanade. Real-time topometric localization. In *2012 IEEE International Conference on Robotics and Automation*, pages 1635–1642. IEEE, 2012. 3

[2] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018. 1

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 3

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2, 3

[5] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263, 2008. 3

[6] Jake Bruce, Niko Sünderhauf, Piotr Mirowski, Raia Hadsell, and Michael Milford. Learning deployable navigation policies at kilometer scale from a single traversal. *arXiv preprint arXiv:1807.05211*, 2018. 2, 3

[7] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 3

[8] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015. 3

[9] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 2, 3, 4, 6

[10] Feras Dayoub, Timothy Morris, Ben Upcroft, and Peter Corke. Vision-only autonomous navigation using topometric

maps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1923–1929. IEEE, 2013. 3

[11] Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002. 3

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. 3, 5

[13] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017. 3

[14] Patrick Foo, William H Warren, Andrew Duchon, and Michael J Tarr. Do humans integrate routes into a cognitive map? map-versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(2):195, 2005. 3

[15] Wei Gao, David Hsu, Wee Sun Lee, Shengmei Shen, and Karthikk Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. *arXiv preprint arXiv:1710.05627*, 2017. 2, 3, 4

[16] Sabine Gillner and Hanspeter A Mallot. Navigation and acquisition of spatial knowledge in a virtual maze. *Journal of cognitive neuroscience*, 10(4):445–463, 1998. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 4

[18] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 3

[19] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 6

[20] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2015. 1, 3, 4

[21] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, pages 2419–2430, 2018. 2

[22] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 3

[23] Mark G. Packard and James L. McGaugh. Inactivation of hippocampus or caudate nucleus with lidocaine differentially affects expression of place and response learning. *Neurobiology of Learning and Memory*, 65(1):65 – 72, 1996. 1

[24] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018. 2, 3, 5

[25] Edward C Tolman. Cognitive maps in rats and men. *Image and environment: cognitive mapping and spatial behavior*, 1948:27–50, 1973. 3

[26] Ranxiao Frances Wang and Elizabeth S Spelke. Human spatial representation: Insights from animals. *Trends in cognitive sciences*, 6(9):376–382, 2002. 3

[27] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2015. 4

[28] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 3