Hierarchical end-to-end autonomous navigation through few-shot waypoint detection

Amin Ghafourian[*]
University of California, Davis
`aghafourian@ucdavis.edu`

Zhongying CuiZhu[*]
Ford Motor Company
`cuizy3@gmail.com`

Debo Shi
University of California, Davis
`deshi@ucdavis.edu`

Ian Chuang
University of California, Davis
`itchuang@ucdavis.edu`

Francois Charette
Ford Motor Company
`charette.francois@outlook.com`

Rithik Sachdeva
University of California, Davis
`rithiksachdeva@gmail.com`

Iman Soltani[†]
University of California, Davis
`isoltani@ucdavis.edu`

Abstract

Human navigation is facilitated through the association of actions with landmarks, tapping into our ability to recognize salient features in our environment. Consequently, navigational instructions for humans can be extremely concise even easily distilled into linguistic content, indicating a small memory requirement and no reliance on complex and overly accurate navigation tools. Conversely, current autonomous navigation schemes rely on accurate positioning devices and algorithms as well as extensive streams of sensory data collected from the environment. Inspired by this human capability and motivated by the associated technological gap, in this work we propose a hierarchical end-to-end meta-learning scheme that enables a mobile robot to navigate in a previously unknown environment upon presentation of only a few sample images of a set of landmarks along with their corresponding high-level navigation actions. This dramatically simplifies the wayfinding process and enables easy adoption to new environments. For few-shot waypoint detection, we implement a metric-based few-shot learning technique through distribution embedding. Waypoint detection triggers the multi-task low-level maneuver controller module to execute the corresponding high-level navigation action. We demonstrate the effectiveness of the scheme using a small-scale autonomous vehicle on novel indoor navigation tasks in several previously unseen environments. The code, dataset, and example test recordings are made available online.[1]

1 Introduction

It has been widely accepted that some form of accurate positioning scheme such as visual or Lidar SLAM or GPS is an indispensable part of the sensory stack in a mobile robot [8, 2]. However, access

[*]Equal contribution.
[†]Corresponding author.
[1]This work was supported by Ford Motor Company.
Code: `https://github.com/Soltanilara/DNS.git`.
Dataset: `https://ucdavis.box.com/s/9tt9usez75uj5l551sgtghgpbgsuc955`.
Sample test recordings: `https://youtu.be/7gZOqxzsSpI`.

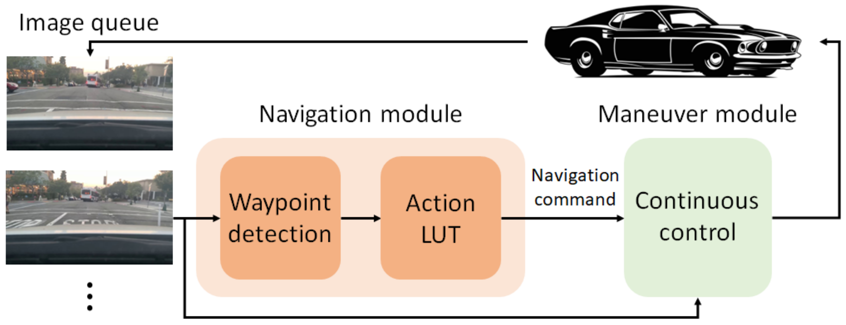


Figure 1: Proposed workflow. One or a few example images are used by the mobile robot to detect predefined landmarks in the environment. Upon landmark detection, the corresponding high-level discrete navigation action (e.g. turn right, turn left, etc.) is retrieved from a lookup table and passed to a continuous maneuver controller. Continuous control executes the resulting navigation command while avoiding obstacles and maintaining the vehicle on a drivable path.

to such sensory data and the associated processing poses challenges in terms of computational load and real-time processing limitations, hardware requirements, sensor fusion, and data integration [27, 35], adding to the cost and complexity of the system. In addition, some sensory modalities such as Global Positioning Systems (GPS), may not be robustly available on factory floors or in dense urban areas featuring high-rises, multi-level bridges, tunnels, and underground passageways due to signal blockage or multipath fading. Even assuming a future with extremely low-cost sensing and unbounded computational and power resources, redundancy is key for the widespread adoption of mobile robots in general and autonomous vehicles in particular, necessitating the development of novel navigation tools that can complement existing technologies.

In this research, we present a different perspective on vehicle trajectory planning and control which enables an autonomous vehicle to drive in new environments without reliance on computationally intensive localization or associated sensors. This approach is inspired by how humans are capable of following linguistic address instructions to traverse through an unseen route solely based on descriptions of a limited set of visual cues paired with a corresponding high-level navigation action ("Turn left at the block with red building," for instance). In this approach, it is assumed that the mobile robot is capable of low-level maneuvers while executing high-level navigational decisions such as turning right/left, making U-turns, etc. In such a case, to successfully complete a route, the mobile robot only needs to reliably recognize where along the path new high-level navigation decisions should be made. With this perspective, in our approach, hereafter referred to as DNS (Description-based Navigation System), path planning and execution are decomposed into a two-level hierarchy of maneuvering and navigation [5]. The former is concerned with vehicle maneuvers with the aim of maintaining the vehicle on a drivable passageway while avoiding obstacles. The latter provides higher-level planning in which the vehicle is guided toward a final destination by associating a limited number of images of distinct waypoints with their corresponding high-level actions.

Figure 1 demonstrates a workflow for the proposed hierarchical approach. Separate modules are assigned to the low-level maneuver and high-level navigation aspects of the methodology, issuing continuous control commands and discrete high-level decisions, respectively. To prepare a vehicle for an upcoming route, firstly a set of waypoints are identified. These waypoints are locations in which the vehicle has to execute a new high-level action. The vehicle is then provided with a description of the route, comprising the visuals of the waypoints along with their assigned navigation actions compiled into a look-up table (LUT). Upon recognizing a waypoint through visual cues, the vehicle then refers to the corresponding action to be executed by the maneuver control module which generates continuous commands to accelerate/decelerate and steer the vehicle.

In this work, we assume that we have access to at least one or a few example images of each waypoint location. This is a reasonable assumption given the availability of street view data e.g. through Google Street View for road navigation, and the low cost of collecting a limited set of such images for indoor applications. We then formulate the waypoint detection process through a few-shot learning scheme. The incorporated few-shot scheme is based on distribution embedding of visual scenes to allow for more expressive representations and improve the robustness of the method given the limited teaching samples. The encoding is paired with an enhanced latent space metric for comparing memory content and online data stream.

The contributions of this work can be enumerated as follows: 1) We present a simplified description-based, end-to-end navigation scheme in which low-level maneuvering and high-level navigation are split into separate modules. The route is described to the vehicle via a LUT constituting one or a few example images of key waypoints along with their corresponding high-level navigation actions, which are then used to condition the maneuvering control upon waypoint detection. 2) We present an efficient scheme for metric-based few-shot learning for robust identification of waypoints. 3) Finally, via experiments in multiple environments we demonstrate how well the proposed waypoint detection and high-level navigation model integrates with a conditional network for low-level maneuver control of the autonomous vehicle.

The paper hereafter is organized as follows: in Section 2 we review some of the prominent and recent approaches to various aspects of autonomous navigation including localization, path planning, and motion control. Section 3 discusses the proposed description-based modular navigation, and in Section 4 the proposed few-shot technique and its deployment in the context of DNS is explained in detail. In Section 5 we demonstrate the performance results for offline as well as real-time implementation of DNS and present the associated ablation study. Section 6 concludes the paper and presents future opportunities.

2 Related Works

In recent years, localization, path planning, and motion control for autonomous navigation have been active areas of research. Various sensors including cameras, GNSS (Global Navigation Satellite System) receivers, IMU (Inertial Measurement Unit) devices, visual and thermal cameras, and Lidar sensors have been adopted to develop autonomous functions using various AI technologies as well as classical control paradigms. In this section, we review some of the recent works in this domain.

[23] and [10] propose CNN-based outdoor and indoor localization techniques in a previously seen environment based on landmark detection and image classification. In [3], GNNS/INS (Inertial Navigation System) highway localization is further refined against detected signs and road facilities. In [24, 25], the need for HD prior maps for localization is relaxed by using Standard Definition (SD) maps paired with onboard perception for inferring the HD map online. Further, precise localization is obtained using a Localizing Ground Penetrating Radar (LGPR) to retrieve stable underground features that are robust to weather changes. [17] combines OpenStreetMap road network and local perception information obtained by 3D-Lidar and CCD camera sensor fusion to refine the global localization and path planning. [31] combines the point-to-point local planner with Ultra-wideband (UWB) localization technology.

For path planning and motion control, [28] uses semantic line detection and segmentation to identify traversable lanes for agricultural robots and vehicles. [22] also uses image segmentation paired with topological maps for road following and control. [20] leverages image segmentation as well as birds-eye view (BEV) perspective transformation to obtain a cost map and through A* algorithm carries out path planning and obstacle avoidance. In [6] lane detection and discrete optimization among considered paths, generated with cubic spline interpolation using GPS data, are used for path planning, and fuzzy sliding mode control is used for steering. [38] incorporates a motion planning system based on polynomial parameterization to compute the path, as well as a model-predictive-control-based system for trajectory tracking. [7] emphasizes route repeating capability through teach and repeat that utilizes odometry information and generates correction signals through lightweight processing of the visual input. For improving autonomous motion control, [19] proposes incorporating prediction of road drivers' intentions through hidden Markov models. Some techniques are also specifically designed to perform specialized maneuvers such as lane change [36], overtaking [26], and roundabout maneuvers [12].

Imitation learning [21, 4, 16] and reinforcement learning [31, 12, 9, 34, 33, 37] have also been attractive domains in mobile robotics research. [21] incorporates graph convolutional neural networks (GCNNs) for the perception of global and geometric features of the environment during complex navigation situations such as unsignaled intersections. Navigation policy is learned through conditional imitation learning given the high-level navigation command. [4] generates a BEV representation through adversarial training that also denotes the desired route given camera input and high-level navigation command obtained by GPS route planning. Given the BEV prediction and the current state of the vehicle, control policy is learned from expert demonstrations. In [16], the authors demonstrate how by utilizing Imitation from Observation (IfO) a robot can learn a good navigation policy for a route using the demonstrator's ego-centric video despite viewpoint mismatch. In [9] RL is used with the objective of maximally prolonging each episode in order to sequentially learn navigation policies. [34] proposes a deep RL approach for lateral vehicle motion control by using sequences of waypoints generated by a planning module to predict steering on simulated data. [33] leverages inverse RL to infer the cost function that justifies the expert behavior given the expert's observations and state-control trajectory. It then optimizes for a model of expert behavior. Finally, [37] introduces self-supervised environment synthesis that can utilize limited real-world data to construct navigation scenarios and synthesize training environments, which help mitigate the performance drop when bringing a model trained on simulation to the real world.

Some recent works have also demonstrated how LLMs and vision-language models can be leveraged for robot navigation tasks. In [18] LLMs are used toward recomposing API calls to generate new policy code to process perception outputs and parameterize control primitives given natural language commands. [15] additionally uses vision-language models for geometric mapping of the environment.

In the following section, we discuss the Description-based Navigation System (DNS), a control scheme that places special emphasis on data efficiency. DNS, which is grounded in the latest advances in few-shot learning, draws inspiration from the human ability to traverse unfamiliar environments using minimal address information to reach a desired destination.

3 Description-Based Navigation System

The proposed process of setting up a mobile robot for an upcoming route consists of the following steps:

1. Identify a set of waypoints along the route: These waypoints are only selected at locations where the vehicle has to execute a new high-level navigation action, e.g. at an intersection where a vehicle should take a turn.
2. Assign a discrete navigation action to each waypoint and form an address lookup table (LUT).
3. Retrieve example images of each waypoint location: In this approach, we are assuming that for each waypoint, one or a few example images are available.
4. Obtain representations for each waypoint by processing the images through the model.
5. Provide the waypoint representations to the vehicle so that they become identifiable upon future exposure to similar visual cues.

Given the limited number of samples available for each landmark as well as the variety of potential routes in many applications, few-shot learning is adopted to train the high-level navigation model so that it can quickly adapt to previously unseen waypoints. Individual memory slots are assigned to specific landmarks and paired with their corresponding high-level navigation decisions. Memory slots are populated using the extracted landmark representation (see Section 4 for more detail). The above process, hereafter referred to as the route teaching stage, is schematically shown in Figure 2.

At drive time, the captured images are continuously processed using the same model as that used during the teaching stage. The generated representations are compared against memory content associated with the upcoming waypoint for landmark detection. Upon recognition of a waypoint, the vehicle refers to LUT to retrieve the corresponding high-level action which is then provided to the low-level maneuver control unit for execution. This process is schematically demonstrated in Figure 3.

The maneuver control unit is a neural network that receives incoming images paired with an action condition (e.g. turn right) and controls the vehicle to execute the desired action (Figure 4). The end-to-end maneuver control module is trained via imitation learning to receive images as input and, conditioned on the high-level action, issue acceleration, deceleration, and steering commands. This controller maintains its latest action condition until a different high-level decision associated with a newly detected waypoint is made. Figure 5 schematically demonstrates this point.

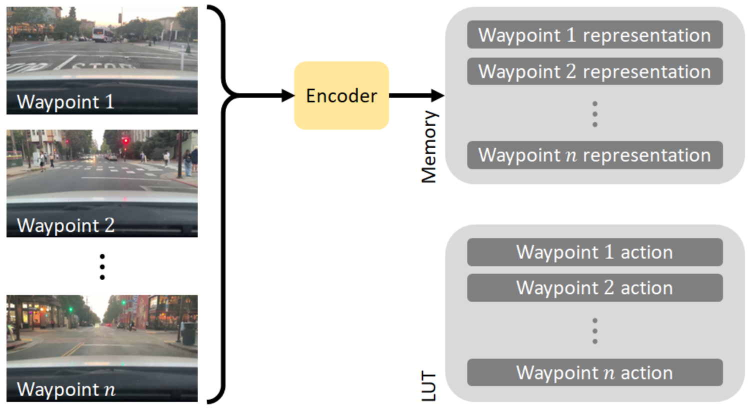


Figure 2: Route teaching stage. Prior to vehicle departure, the image representations from predetermined waypoints are used to populate the corresponding memory slots along with the high-level navigation command for future reference.

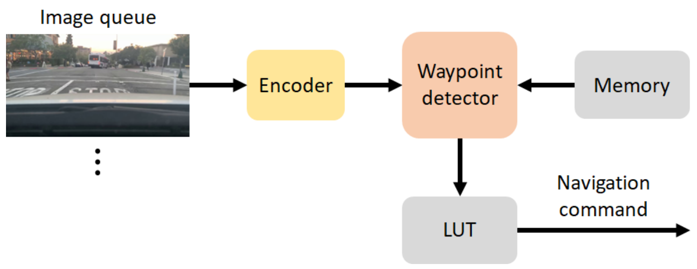


Figure 3: High-level navigation module. During inference, the navigation module processes the incoming images to compare them against memory content and detect waypoints. Upon detection, the corresponding action is retrieved from the lookup table and issued to the maneuver control unit.

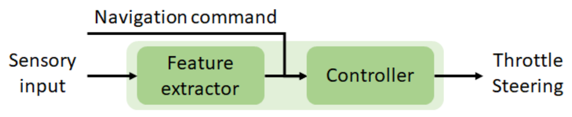


Figure 4: The low-level maneuver control module is composed of a feature extractor that processes the camera input, which is then concatenated with the discrete navigation command and presented to the continuous controller to obtain steering and acceleration/deceleration outputs.

4 Few-shot learning for waypoint detection

In this section, we describe the formulation of few-shot classification and elaborate in detail on how the proposed few-shot technique is adapted for waypoint detection training and deployment.

4.1 Few-shot classification: problem formulation

As part of a few-shot classification training, we are given dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ of labeled samples, where each $y_i \in \{1, \ldots, K\}$ is the label of the corresponding example $\mathbf{x}_i$. For training, many w-way s-shot classification tasks, or episodes, can be randomly formed, where w denotes the total number of participating classes and s denotes the number of labeled examples or shots, commonly referred to as the support set, for each class. The model is then trained by minimizing classification loss on q unlabeled samples (queries). Trained under this scheme, the model learns to rely on a small number of labeled samples (shots) for new tasks with novel participating classes.

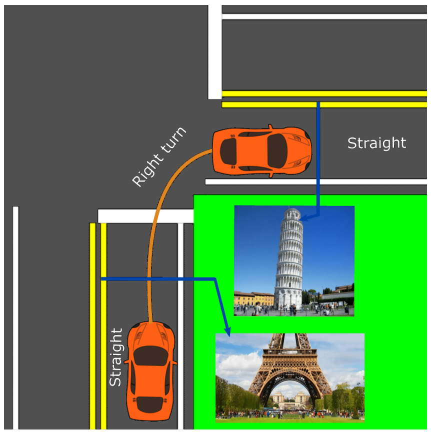


Figure 5: Example right turn. The vehicle maneuver control is initially conditioned to drive straight. Upon detecting the first waypoint (view with the Eiffel Tower), the condition switches to right turn, and upon detecting the next waypoint (view with the Tower of Pisa) it switches back to straight.

4.2 Enhanced metric few-shot learning

In metric few-shot learning techniques, one aims to find a suitable representation that places instances from the same class close to each other while separating instances from different classes in the embedding space. This will facilitate accurate classification. In Prototypical Networks [30], for instance, this is done by minimizing the Euclidean distance between a query and its true class prototype defined as the mean of the support sample representations for that class, while maximizing its distance from all other class prototypes.

Deep variational embeddings in the context of few-shot learning have been shown to improve classical metric few-shot learning techniques [39, 11], which can suffer from noise due to data scarcity and lack of interpretability. In these methods, rather than estimating a class mean based on a few sample vector representations, a more expressive class representation is adopted, for instance, by assuming a Gaussian form. In [39], each sample is mapped to a Gaussian distribution of its associated class through a neural architecture, whose output is interpreted as mean and the variances that form a diagonal class covariance matrix. Individual sample outputs from each class are then combined to obtain a more accurate estimate of class mean and covariance.

We adopt a similar scheme, in which we retrieve class distribution mean and diagonal covariance estimates for each sample through a neural network and combine samples to form a more accurate class distribution. For class k, we obtain the combined distribution $\mathcal{S}^k = (\mu^k, \Sigma^k)$ by obtaining combined estimates for mean and covariance. The combined estimate of class k mean is simply the average of individual mean estimates:

$$\mu^k = \frac{1}{s}\sum_{i=1}^{s} \mu_i. \tag{1}$$

We adopt a linear combination of diagonal covariance estimates to obtain a combined minimum variance estimate [13]. Assuming equal weights, it reduces to an arithmetic mean:

$$\Sigma^k = \frac{1}{s}\sum_{i=1}^{s} \Sigma_i. \tag{2}$$

Given a query $\mathbf{x}_j$, the associated distribution $\mathcal{Q}_j = (\mu_j, \Sigma_j)$ is obtained from the network and viewed as the estimate of the class distribution associated with that query. To quantify the distance between class support and queries one possibility is to rely on a distribution-to-distribution symmetrized Mahalanobis distance. In this form, the distance between the query and class k distribution becomes

$$d^2(\mathcal{Q}_j, \mathcal{S}^k) = (\mu_j - \mu^k)^T (\Sigma_j^k)^{-1} (\mu_j - \mu^k) = \sum_{i=1}^{d} \frac{(\mu_{j,i} - \mu_i^k)^2}{\Sigma_{j,ii}^k}, \tag{3}$$

where d indicates the dimensionality of the embedding space and

$$\Sigma_j^k = \frac{1}{2}\left(\Sigma_j + \Sigma^k\right). \tag{4}$$

4.3 DNS with distribution embeddings

Figure 6 shows the waypoint detection model. In order to train the proposed technique for use in few-shot navigation, we collect and use a dataset of various courses, with multiple repetitions (or laps) of each. For each lap of each course, the frames are split into segments corresponding to positive or negative examples for each waypoint within that course. Frames at which the high-level navigation action corresponding to the n^{th} waypoint can be safely initiated will be used as positive class examples for that waypoint. All the frames beyond the positive segment of the $(n-1)^{\text{th}}$ waypoint and prior to the n^{th} waypoint's positive segment will be used as the negative class examples for the n^{th} waypoint.

In each classification task during training, we construct the support distribution for the positive class (i.e. a given waypoint) as follows: first, we choose a random lap from a random course, then we make a random selection of s consecutive images from a random waypoint in the lap. We then pass each image through the backbone and present to mean and covariance modules to retrieve individual distributions and obtain a combined prototypical estimate of the distribution associated with the waypoint. We then obtain query distributions by sampling s consecutive frames from either the positive or negative classes. q_p positive query distributions are constructed from the positive segment associated with the same waypoint but in different recordings of the path. q_n negative distributions are taken from the corresponding negative segment in either the same lap or different laps. Combined distributions are then created in a similar fashion as the positive support distribution.

Next, the distance between each of the $q_p + q_n$ queries and the positive support distribution is calculated across individual embedding space dimensions (i.e. separately for each $i \in \{1, \ldots, d\}$ in Equation 3) and the elementwise distance vector is presented to a classifier module as it preserves more relevant information for the subsequent classification. The classifier estimates the probability of the query matching the waypoint. Binary cross-entropy loss is then calculated between true and predicted waypoint assignments for queries in order to update model parameters.

At inference, the vehicle memory is populated with one or multiple combined distributions per waypoint, corresponding to consecutive frames from a single previous recording of waypoint locations. Similar to training time, query distributions are formed by combining a number n_q of the most recent consecutive frames captured via the vehicle camera. The elementwise distance vector between the incoming query and the memory distribution associated with the upcoming waypoint is calculated and presented to the classifier module to determine whether the waypoint is reached. If the output probability is larger than a threshold value, the waypoint is detected. In the case of multiple memory distributions for a waypoint, the maximum probability is considered. Once a decision is made on the executable action, it is sent to the maneuver control unit to update the network condition and accordingly change the throttle and steering response.

5 Experiments

5.1 Dataset and training

The dataset consists of 36 courses, recorded in 11 University of California, Davis buildings. Of 36 courses, 18 are collected on a clockwise and 18 on a counterclockwise loop, with CW/CCW pairs covering the same closed paths. The recordings are repeated multiple times so that there are between 2 and 8 completed recordings for each course. Sample images are shown in Figure 7.

The recordings are done using a remote-controlled vehicle equipped with cameras. Figure 7 shows the setup of the RC vehicle. It is equipped with Arduino Mega for analog control of steering and throttle, as well as an onboard ZOTAC mini PC with 8 GB RAM, Nvidia GeForce RTX 2070 Mobile GPU, and Intel Core i7-9750H processor to run custom navigation models. Two ELP fisheye cameras with a $180°$ field of view are mounted on the car, each turned $30°$ outwards, creating a $60°$ relative angle between the two camera orientations. Fisheye cameras were used given their wide field of view. While the images will be distorted, given that the distortion profile is fixed and deterministic, this will not pose any significant issues. The car can be set to manual or autonomous, which we will use for online evaluation.

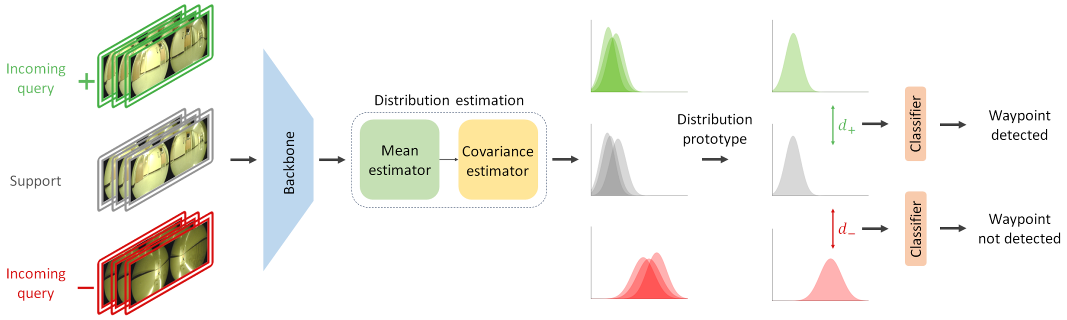
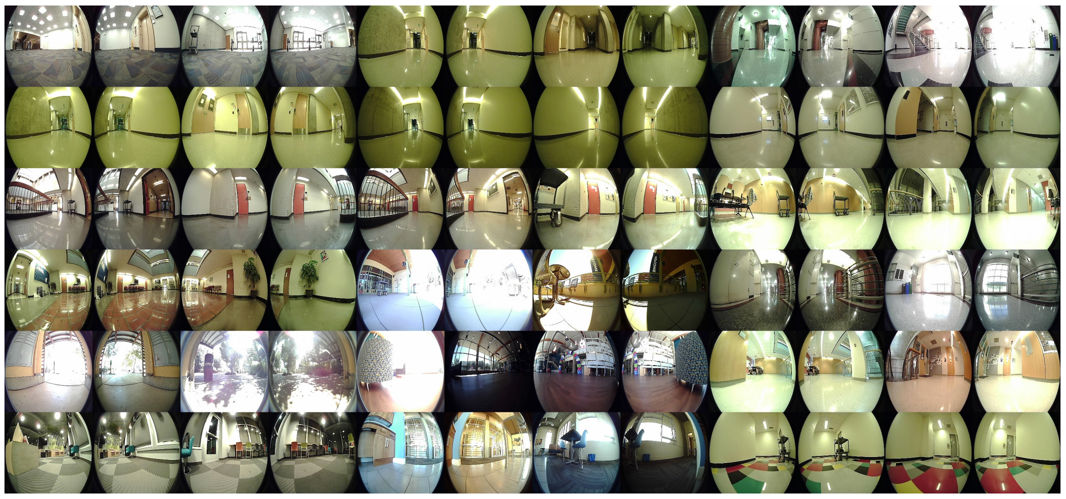


Figure 6: For each waypoint, a sequence of frames from an existing recording at the waypoint location (gray outline) are passed through the backbone and the distribution estimation model to obtain a corresponding class distribution for each frame (shown in gray). The distributions are then combined to form the waypoint prototype, which will be stored in memory for use during wayfinding. For a query frame sequence, the distribution is similarly obtained. The elementwise distance between the memory prototype and the real-time calculated query distribution is obtained and fed to the classifier for detection (shown in green and red respectively for a match and a mismatch). Once a waypoint is detected, the corresponding navigation command is retrieved from the LUT to condition the low-level continuous control module.



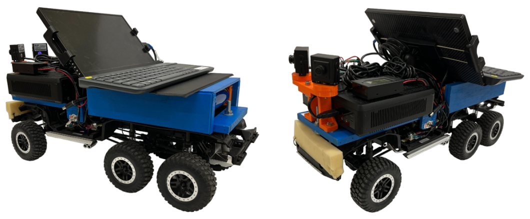


Figure 7: Sample dataset images (top) and RC vehicle used for data collection and evaluation (bottom).

To segment each course into various landmark/non-landmark segments after collecting data, we use the output steering signal. A fixed number of frames before the onset or completion of a turn are generally considered to be positive samples for waypoint detection, i.e. any one of these frames would have been a safe candidate for initiating the waypoint action at the time of recording.

To train the model, 12 CW/CCW pairs are used. 6 courses are used for validation in order to identify the best-performing model. The remaining courses are reserved for testing. Since our main focus in this work is on landmark detection and given the limited data-collection budget, the data for training the maneuver control module included the test locations in the form of repeats of each road segment with random actions at various intersections. The corresponding action is provided as a condition to the control module during its training. We note that while the downstream low-level control module has been exposed to test courses, the course remains unseen for the waypoint detection model, and successful navigation hinges on successful landmark detection. We do not consider the effect of obstacles on the controller performance. Waypoint detection module training, however, is done with augmentations that promote robustness to scene occlusion (please see supplementary material).

We use the ResNet-50 architecture [14] as the waypoint detection model backbone. We estimate the sample class mean through a linear layer applied to backbone-generated features. The covariance module and the classifier are fully connected modules with 2 and 4 layers, respectively. To ensure positive variances, Softplus activation is used at the covariance output. Sigmoid is applied at the classifier output, which denotes the probability that a waypoint match is detected given the elementwise distance input.

For initializing the ResNet-50 encoder, we use weights obtained through pretraining on ImageNet [29] in the self-supervised manner described in [1], which is made available by the authors. We believe a backbone pretrained on ImageNet is a good candidate initialization given the amount and diversity of real-world visual data that it contains. We train the model in two phases. In phase 1, we freeze the backbone and only train the mean and covariance modules, as well as the classifier. This prevents dramatic unfavorable updates to the backbone as a result of random initialization of other components. Next, we pick the best-performing model from this phase and jointly fine-tune all model parameters. For training the control unit, we use an EfficientNet-B0 backbone [32] and augment the output by the high-level navigation condition before passing through a fully connected segment whose output value will be used as a signal to steer the vehicle. For simplification, we fix the vehicle throttle and only consider steering. More details on training and hyperparameters are included in the supplementary material.

5.2 Offline evaluation

We use the 6 unseen test courses to evaluate the performance of the waypoint detection model offline. For evaluation on each course, we construct several positive class distributions per waypoint using samples from a single lap and store them in memory. We then provide frames from other laps in the same course in sequence and construct query distributions by combining the most recent consecutive frames. The elementwise distance between the query and memory distributions from the upcoming waypoint is then provided to the classifier for detection, where the maximum match probability among memory distributions is considered. The accuracy is evaluated using the F_1 score between ground truth and the predicted label of each road segment. This gives us a conservative estimate of model performance (please see supplementary material).

5.3 Online evaluation

For online evaluation, the trained model is loaded onto the vehicle computer. Test location waypoint images extracted from a single lap recording (memory lap) and their corresponding actions are provided to the model to construct memory distributions and the LUT. It is then set to autonomous driving mode to navigate the path. Each course test is repeated four times in total, twice for each of the two different memory laps. The course-level (completed course) as well as the waypoint-level (correctly identified waypoints) success rates are reported.

5.4 Results and ablation study

Table 1 shows the results from offline evaluation. It also demonstrates the effects of backbone pretraining as well as distribution embedding with the proposed symmetrized Mahalanobis metric versus point embedding and Euclidean metric adaptation. Pretraining significantly enhances the model's performance. Self-supervised pretraining gives superior performance compared to its supervised counterpart. Distribution embedding also improves the performance.

In online evaluation, each of the 6 courses consists of 8 waypoints. Each course is evaluated 4 times, corresponding to two repetitions for each of the two memory laps. We also consider a separate, long course (Figure 8) consisting of 20 waypoints and make two recordings with which we populate the

Table 1: Offline evaluation results and ablation study. Metric denotes whether the Euclidean or the proposed symmetrized Mahalanobis metric was used in the few-shot learning context.

Backbone Pretraining	Metric	F_1 Score (%)
–	Symmetrized Mahalanobis	66.7
Supervised (ImageNet)	Euclidean	81.8
Supervised (ImageNet)	Symmetrized Mahalanobis	82.3
Self-supervised (ImageNet)	Euclidean	85.7
Self-supervised (ImageNet)	Symmetrized Mahalanobis	**86.8**

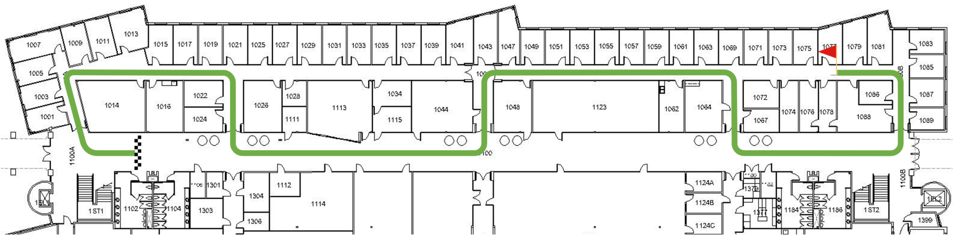


Figure 8: Long course for online evaluation.

memory and test in a similar manner. This results in 28 total course-level and 272 waypoint-level evaluations. The model successfully completed all courses except for two, in one of which one waypoint was not detected. In that experiment, a large part of the frame near the waypoint was occluded by a group of pedestrians walking in front of the vehicle. As for the other, the waypoint detection was successful but the control module failed to complete the maneuver, likely due to strong glare. This puts the course-level and waypoint-level success rates at 92.9% and 99.3%, respectively.

6 Conclusion and future work

In this work, we proposed a two-stage, end-to-end technique for autonomous navigation consisting of a high-level waypoint detector based on few-shot learning, as well as a low-level maneuver control unit that controls the vehicle conditioned on the high-level navigation input. This technique only requires a minimal amount of data from critical waypoints on an unseen route and has a low memory and computation demand. We believe the demonstrated offline and online results serve as proof of concept and motivate further developments with the aim of significantly reducing the need for positioning devices, expensive repeated training, and extensive data from target environments.

We are conducting additional research to gauge the approach, its robustness, strengths and limitations, as well as efficient ways to improve it in more diverse environments including outdoor settings. Utilizing public road data (e.g. Google Street View) as a low-cost source of waypoint support examples constitutes another aspect of our continued work. Besides more realistic autonomous driving implementations, for instance, on a full-scale vehicle that uses street images paired with a more sophisticated maneuver control mechanism, several exciting directions remain unexplored. One is the regime in which the car has missed or misidentified a waypoint, which will likely throw the vehicle entirely off-path. Robustness to outdoor landscape changes over time for instance due to seasonal variations is another direction of focus. Going forward, addressing these problems without substantial data collection, computation, and hardware overhead will be topics of particular interest.

Acknowledgments

The authors would like to thank Dechen Gao, Aryan Mondkar, and Mohamed Shais Khan for their contributions, including their help with data collection.

Supplementary Material

Waypoint detection training and evaluation details

At each frame, the left/right camera images are separately processed. They are resized to 224×224 and normalized. During training, random rotation, color jitter, and coarse dropout are applied to the images. After obtaining the 1000-dimensional elementwise distances associated with each camera, the two vectors are concatenated and presented to the classifier.

At each waypoint, 15 frames leading to the steering initialization/completion are regarded as positive frames for that waypoint, corresponding to a conservative viable range to start steering earlier than where the steering was initiated when recording. The number of consecutive frames s combined into a single distribution is set to 10 for both support and query. The same number is also used in evaluation. During training, each training episode has 1 positive and 6 negative queries (q_p and q_n, respectively). Parameters are updated after processing each episode batch size of 36 in phase 1 and 3 in phase 2. A total of 240 and 4000 iterations are processed in phases 1 and 2, respectively. Adam optimizer is used with an initial learning rate of 10^{-4} in phase 1 and 10^{-5} in phase 2. The learning rate is divided by two after 160 iterations in phase 1 and every 1000 iterations in phase 2. In phase 1, the model is evaluated on the validation set every 32 iterations as well as at the end. It is evaluated every 200 iterations in phase 2.

For offline testing of a course, waypoint frames of a single lap are processed to obtain support distributions associated with waypoint locations and store them in memory. A sliding 10-frame window across a 15-frame range yields 6 different combined memory distributions per waypoint. Once the memory is populated, we run through each test lap frame by frame, each time comparing the most recent combined distribution with each of the 6 memory distributions, and obtain the match probability as the maximum of the 6 resulting probabilities. Moving average is applied to the generated probabilities to reduce noise. A waypoint is detected if the maximum of the 6 probabilities is above a cutoff probability. Once a waypoint is traversed, the expected upcoming waypoint is updated and a new binary classification task is set up. The evaluation is repeated for all configurations of memory/test laps and average performance is reported.

Waypoint-level F_1 score is calculated as follows: If a high-probability frame occurs in a non-waypoint segment, it will be counted as a false positive, and if no high-probability frames are detected in a waypoint segment, a false negative will be recorded. Each waypoint range is stretched by a fixed small number of buffer frames before and after the original 15-frame range to improve the metric. The frames prior to a waypoint account for the common occurrence of rising probabilities when nearing a waypoint and hence, erroneously amplifying the false-positive rate (see Figure 9). The frames after the nominal waypoint frames approximate positions where a turn could still be safely initiated although a bit later than in the recording. As such, the offline F_1 heuristic presents an underestimation of performance. Per our observations, nearly all false positives correspond to early detections which often pose no challenge in practice as implied by the large gap between online and offline evaluation results.

For online evaluation, besides populating the memory, a lookup table is also stored with keys corresponding to waypoint IDs. Instead of feeding consecutive test frames offline, the car is set to autonomous to navigate the course. We did not notice any latency in online experiments. Both through the steering module training procedure and observing vehicle detections during the test, we ensure that successful performance at each waypoint follows a valid detection. If the car fails at a waypoint, it is set up immediately after that waypoint, and the upcoming waypoint ID is manually updated to evaluate the remainder of the course.

Low-level maneuver control module details

Data for the low-level control module is collected by running the car in test locations. At waypoint positions, each maneuver action (straight/left turn/right turn) is repeated 3-5 times. This data is added to the previously collected test location data to sensitize the model to the action condition and avoid memorizing a specific navigation action at waypoints.

The model is composed of a pretrained EfficientNet-B0 backbone, whose 1280-dimensional output passes through a linear layer and ReLU activation. At this stage, the 500-dimensional output is concatenated with a 3-dimensional one-hot vector denoting the action condition (left turn/right

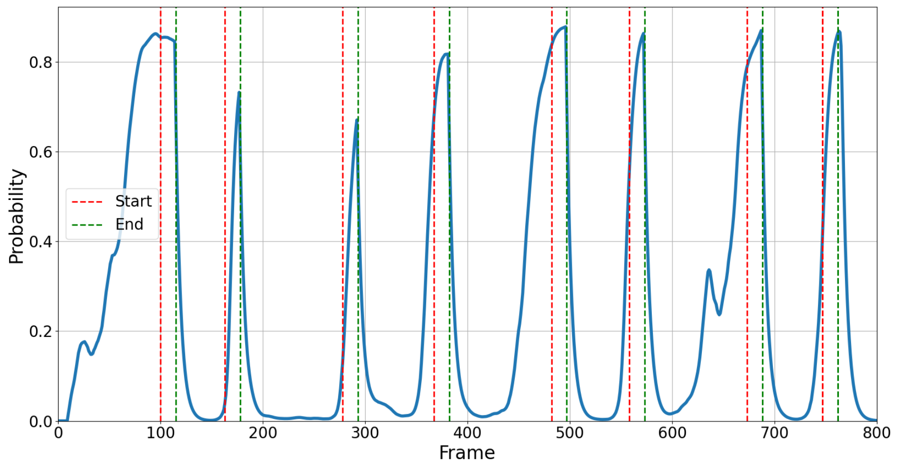


Figure 9: Plot of waypoint match probability values for a sample course during offline evaluation. Consecutive red and green vertical lines denote the original 15-frame range associated with each waypoint. As exemplified in this plot, several false positives can occur due to rising probabilities before reaching the nominal waypoint frames (for instance in the range between the third and fourth waypoints). These are, however, not an issue in practice as online experiments show.

turn/straight), which then passes through two more layers with ReLU and sigmoid activations and output dimensions of 100 and 1, respectively. The output is considered as steering.

The model is trained using mean square error (MSE) loss between true and predicted steering. The two camera images are concatenated, resized to 104×224, and normalized. During training, random color jitter, Gaussian blur, and horizontal flip are applied to the image. In the case of horizontal flipping, the steering is mirrored and in case the action condition is turning, it is adjusted to reflect a turn in the opposite direction. The model is trained for 100 epochs with Adam optimizer and a fixed learning rate of 10^{-4}.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[2] Jun Cheng, Liyan Zhang, Qihong Chen, Xinrong Hu, and Jingcao Cai. A review of visual slam methods for autonomous driving vehicles. *Engineering Applications of Artificial Intelligence*, 114:104992, 2022.

[3] Kyoungtaek Choi, Jae Kyu Suhr, and Ho Gi Jung. Map-matching-based cascade landmark detection and vehicle localization. *IEEE Access*, 7:127874–127894, 2019.

[4] Gustavo Claudio Karl Couto and Eric Aislan Antonelo. Hierarchical generative adversarial imitation learning with mid-level input generation for autonomous driving on urban environments. *arXiv preprint arXiv:2302.04823*, 2023.

[5] Zhongying CuiZhu, Francois Charette, Amin Ghafourian, Debo Shi, Matthew Cui, Anjali Krishnamachar, and Iman Soltani. One-shot learning of visual path navigation for autonomous vehicles. In *Machine Learning for Autonomous Driving Workshop at the 36th Conference on Neural Information Processing Systems, New Orleans, USA*, 2022.

[6] Yanyan Dai and Suk-Gyu Lee. Perception, planning and control for self-driving system based on on-board sensors. *Advances in Mechanical Engineering*, 12(9):1687814020956494, 2020.

[7] Dominic Dall'Osto, Tobias Fischer, and Michael Milford. Fast and robust bio-inspired teach and repeat navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 500–507. IEEE, 2021.

[8] César Debeunne and Damien Vivet. A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, 20(7):2068, 2020.

[9] Ambedkar Dukkipati, Rajarshi Banerjee, Ranga Shaarad Ayyagari, and Dhaval Parmar Udaybhai. Learning skills to navigate without a master: A sequential multi-policy reinforcement learning algorithm. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2483–2489. IEEE, 2022.

[10] Farzin Foroughi, Zonghai Chen, and Jikai Wang. A cnn-based system for mobile robot navigation in indoor environments via visual localization with a small dataset. *World Electric Vehicle Journal*, 12(3):134, 2021.

[11] Stanislav Fort. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv:1708.02735*, 2017.

[12] Laura García Cuenca, Enrique Puertas, Javier Fernandez Andrés, and Nourdine Aliane. Autonomous driving in roundabout maneuvers using reinforcement learning with q-learning. *Electronics*, 8(12):1536, 2019.

[13] Franklin A Graybill and RB Deal. Combining unbiased estimators. *Biometrics*, 15(4):543–550, 1959.

[14] 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.

[15] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[16] Haresh Karnan, Garrett Warnell, Xuesu Xiao, and Peter Stone. Voila: Visual-observation-only imitation learning for autonomous navigation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2497–2503. IEEE, 2022.

[17] Jing Li, Hui Qin, Junzheng Wang, and Jiehao Li. Openstreetmap-based autonomous navigation for the four wheel-legged robot via 3d-lidar and ccd camera. *IEEE Transactions on Industrial Electronics*, 69(3):2708–2717, 2021.

[18] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[19] Shiwen Liu, Kan Zheng, Long Zhao, and Pingzhi Fan. A driving intention prediction method based on hidden markov model for autonomous driving. *Computer Communications*, 157:143–149, 2020.

[20] Sagar Manglani. Real-time vision-based navigation for a robot in an indoor environment. *arXiv preprint arXiv:2307.00666*, 2023.

[21] Xiaodong Mei, Yuxiang Sun, Yuying Chen, Congcong Liu, and Ming Liu. Autonomous navigation through intersections with graph convolutionalnetworks and conditional imitation learning for self-driving cars. *arXiv preprint arXiv:2102.00675*, 2021.

[22] Ryusuke Miyamoto, Yuta Nakamura, Miho Adachi, Takeshi Nakajima, Hiroki Ishida, Kazuya Kojima, Risako Aoki, Takuro Oki, and Shingo Kobayashi. Vision-based road-following using results of semantic segmentation for autonomous navigation. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 174–179. IEEE, 2019.

[23] Sivapong Nilwong, Delowar Hossain, Shin-ichiro Kaneko, and Genci Capi. Deep learning-based landmark detection for mobile robot outdoor localization. *Machines*, 7(2):25, 2019.

[24] Teddy Ort. *Autonomous Navigation without HD Prior Maps*. PhD thesis, Massachusetts Institute of Technology, 2022.

[25] Teddy Ort, Jeffrey M Walls, Steven A Parkison, Igor Gilitschenski, and Daniela Rus. Maplite 2.0: online hd map inference using a prior sd map. *IEEE Robotics and Automation Letters*, 7(3):8355–8362, 2022.

[26] Josue Ortega, Henrietta Lengyel, and Zsolt Szalay. Overtaking maneuver scenario building for autonomous vehicles with prescan software. *Transportation Engineering*, 2:100029, 2020.

[27] B Padmaja, CH VKNSN Moorthy, N Venkateswarulu, and Myneni Madhu Bala. Exploration of issues, challenges and latest developments in autonomous cars. *Journal of Big Data*, 10(1):61, 2023.

[28] Shivam K Panda, Yongkyu Lee, and M Khalid Jawed. Agronav: Autonomous navigation framework for agricultural robots and vehicles using semantic segmentation and semantic line detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6271–6280, 2023.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[30] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[31] Enrico Sutera, Vittorio Mazzia, Francesco Salvetti, Giovanni Fantin, and Marcello Chiaberge. Indoor point-to-point navigation with deep reinforcement learning and ultra-wideband. *arXiv preprint arXiv:2011.09241*, 2020.

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[33] Tianyu Wang, Vikas Dhiman, and Nikolay Atanasov. Inverse reinforcement learning for autonomous navigation via differentiable semantic mapping and planning. *Autonomous Robots*, pages 1–22, 2023.

[34] Asanka Wasala, Donal Byrne, Philip Miesbauer, Joseph O'Hanlon, Paul Heraty, and Peter Barry. Trajectory based lateral control: A reinforcement learning case study. *Engineering Applications of Artificial Intelligence*, 94:103799, 2020.

[35] Liyana Wijayathunga, Alexander Rassau, and Douglas Chai. Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review. *Applied Sciences*, 13(17):9877, Aug 2023.

[36] Xiaodong Wu, Bangjun Qiao, and Chengrui Su. Trajectory planning with time-variant safety margin for autonomous vehicle lane change. *Applied Sciences*, 10(5):1626, 2020.

[37] Zifan Xu, Anirudh Nair, Xuesu Xiao, and Peter Stone. Learning real-world autonomous navigation by self-supervised environment synthesis. *arXiv preprint arXiv:2210.04852*, 2022.

[38] Guodong Yin, Jianghu Li, Xianjian Jin, Chentong Bian, and Nan Chen. Integration of motion planning and model-predictive-control-based control system for autonomous electric vehicles. *Transport*, 30(3):353–360, 2015.

[39] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1685–1694, 2019.