Ensembles for Online Dynamics Modeling of Off-road Terrain

Matthew Sivaprakasam[1], Samuel Triest[2], Wenshan Wang[2], Sebastian Scherer[2]

***Abstract*— Planning for autonomous ground vehicles has improved greatly over the past decades. They have become effective by using algorithms that take into account inherent structure and geometry in the surrounding environment. However, the same techniques are more likely to fail when brought off-road where there is less structure and complex terrain. There is still a need for planning techniques that effectively handle physical interactions between a vehicle and its surroundings. Motion planning for off-road terrain often employs some sort of dynamics model specific to a given robot, but these often only include properties specific to the robot itself, and don't account for the fact that these properties are affected by the robot's physical environment. We propose an approach which includes an ensemble of classical models. These dynamics models are generated offline in a semi-realistic simulation that accounts for a wide array of physical environmental properties. A higher-level model is then trained online to choose which dynamics model to use given the performance of previous decisions. This approach results in more accurate state predictions compared to the baseline in simulation in complex terrain.**

***Index Terms*— Learning from Experience, Dynamics, Model Learning for Control**

I. INTRODUCTION

Path planning and control for robots has made great strides in the past years. In planning, hand-tuned costs can be designed to take advantage of structure and geometry in an environment in order to find an optimal path [1]. In order to ensure that a path is kinodynamically feasible, some planning algorithms will also employ some sort of vehicle model that determines what states a robot can reach from its current state. A controller can then use the same model when outputting commands in order to track the desired path. This general approach tends to work well for organized and structured environments [2], [3]. For example, autonomous driving in urban environments is a relatively constrained problem, when it comes to physical interactions with the environment. Due to urban infrastructure, there is a limited number of types of surfaces a car is expected to drive on. Each of these surfaces are generally consistent in terms of physical properties like friction and deformability, which means that simpler models can be used to predict how a vehicle will respond on them.

The consistencies taken for granted in structured environments are often unknown and erratic in off-road environments. Due to obstacles like unpaved surfaces, shrubbery, rocks, and deformable objects, the physical properties of the ground can change even within the span of a meter. This makes it much more difficult to use a single simple model to make predictions the same way it might be used in a simpler environment, as the optimal model changes as the environment changes. Moreover, it is difficult to determine the best model prior to deployment without having extensive knowledge of the given environment beforehand.

In order to use kinodynamic planners and controllers effectively in off-road environments, accurate vehicle modeling is needed. Complex models can be used, for example neural networks, but they require extensive amounts of diverse data in order to be robust to a variety of terrains. Additionally, they rely primarily on data collected prior to deployment rather than on data collected in real-time. In this work we instead propose an ensemble comprised of simple models, that learns online in order to make effective predictions. Transfer functions to predict velocity and steering angle were trained on different types of surfaces, and each model uses one of them to predict the next state of the robot. The effectiveness of each expert in previous decisions is taken into account when making the next decision. By updating the model online, we demonstrate higher accuracy in state predictions compared to the baseline of using a single static model.

II. RELATED WORK

There are several prior works related to the deployment of robots in off-road environments. Xiao *et al.* introduced a data-driven kinodynamic planner for fast off-road navigation [4]. They learn a neural net that takes in IMU information and a desired state and outputs control commands. It takes in no vision-based perception (e.g. lidar, camera), and instead train the inverse kinodynamics model on the IMU data alone. Their work establishes the fact that physical changes in an environment are hard to model by hand but can still significantly affect the dynamics of a vehicle. While Xiao's work employs a model trained offline, online approaches are also becoming more common [5]–[7]. Work by Kumar *et al.* introduces a framework with both an offline and an online phase [8]. The first phase involves learning an encoder to map environment properties (for example friction and terrain height), and then using that as an input for model-free reinforcement learning. The goal of the online component is to map the previous states and actions to the latent vector since physical properties of the environment aren't easily available outside of simulation. By using the different inputs for the online component they are able to address some of the problems that arise with sim-to-real transfer. There are also a number of papers that use learned dynamics models

[1]Matthew Sivaprakasam is with the University of Pittsburgh, Pittsburgh, PA 15213, USA. mjs299@pitt.edu

[2]Samuel Triest, Wenshan Wang, and Sebastian Scherer are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {striest,wenshanw,basti}@andrew.cmu.edu

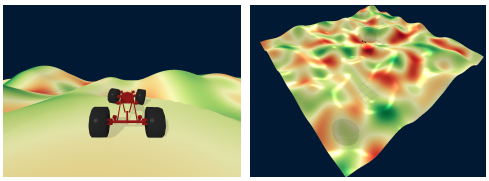

Fig. 1. The car used in simulation, and an example of a randomized terrain. Red areas correspond to lower friction and green areas correspond to higher friction.

directly [9], [10]. Work by Tremblay *et al.* proposes a way of supplementing prior state information with IMU, camera, and lidar information in order to improve dynamics predictions [11]. They treat each of the sensors as different experts, and learn how to prioritize each modality based on the state, which provides a promising method of indirectly extracting environmental information off-road. Xiao's, Kumar's, and Tremblay's works all show the promise of using previous state information in making new decisions in unstructured environments.

III. METHODS

A. Physically-realistic Simulation Environment

Due to the necessity of realistic physics, a custom simulation environment was set up in pybullet [12]. The vehicle in simulation was a highly-configurable 4-wheeled robot with parameters such as max throttle, max steering angle, mass, friction, and suspension limits, damping, spring force. A configurable sensor suite was also implemented for the robot, including front-facing camera, IMU, lidar, shock travel (from suspension), heightmaps, and friction maps. Together, the accurate physics alongside thorough sensors are an attempt to reduce future sim-to-real transfer problems.

B. System Identification and Dynamics Modeling

In order to perform system identification, data was systematically collected in various simulation environments that were created by iterating through a range of slopes and frictions. In each environment, step responses were collected in a range of magnitudes for both commanded velocity and steering angle. This allowed us to model transfer functions that map from commanded to actual velocity and steering, which was accomplished by fitting third-order ARX (autoregressive with exogenous input) time-series models. These models, trained on the step response data using the L-BFGS optimization algorithm [13], maintain buffers of the past 2

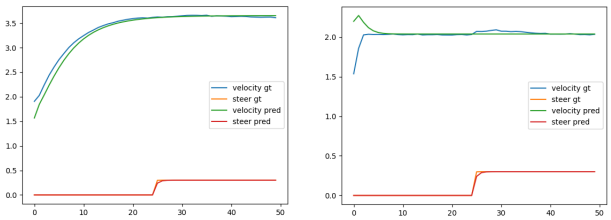

Fig. 2. Velocity and steering transfer functions collected on two surfaces with different friction.

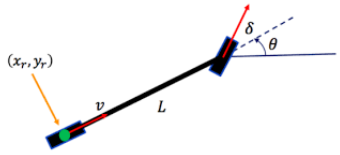

Fig. 3. Kinematic Bicycle Model

states and commands and use them along with the current state and command to output a predicted state. The transfer functions were generated systematically, so for N different friction values and M slope values in data collection, there are a resultant NxM total transfer functions each for velocity and steering angle as shown in Fig. 2.

For modeling the system dynamics, the kinematic bicycle model is used Fig. 3. The model has a three-dimensional state space comprising of location and heading in 2D space (using the rear axle as reference point). It has a single parameter L (vehicle length) and, given an input velocity and steering angle (yaw), predicts the change in state using the following update rules:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v * cos(\theta) \\ v * sin(\theta) \\ v * tan(\delta)/L \end{bmatrix}$$

where x and y are horizontal and vertical position, θ is heading, v is velocity, and δ is yaw. To predict the next state of the robot, the input velocity and yaw are first predicted by feeding in the commanded and current velocity and yaw into their corresponding transfer functions. The predictions along with the current state are then fed into the bicycle model which outputs the next state.

As previously mentioned, all the transfer functions were generated in environments with different physical characteristics (slope and friction). Given the same input states, there is a variation in predictions across all the transfer functions, which impacts the predictions that come out of the bicycle model (Fig. 4). In theory, the variations together encompass the different possibilities of resultant states given any physical properties of the environment, provided that they are within the range of properties the ensemble of transfer functions themselves were trained on. Moreover, the set of transfer functions that results in the most accurate

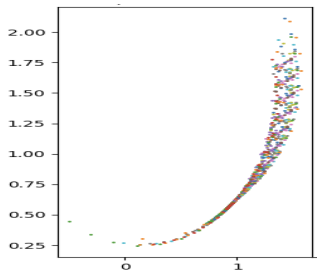

Fig. 4. Predictions using various transfer functions forward-sampled 10 steps into the future

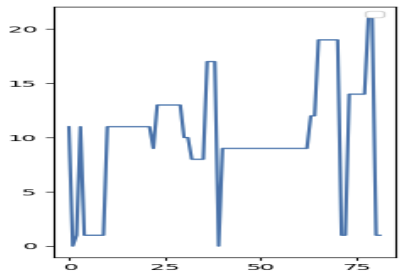

Fig. 5. The index of the best expert over time

predictions changes over time as a vehicle moves through complex terrain (Fig. 5).

C. Online Learning

In a "constant" environment, in which the physical properties are the same across the whole terrain, there would exist a single transfer function from the ensemble that always results in the best predictions. In this case, it would only take monitoring the predictions from the entire ensemble for a few steps to evaluate which transfer function has the highest performance. However, this strategy doesn't work in more realistic environments, where there are lots of variations in the terrain. Changes in the physical properties of the terrain means that the expert from the ensemble with the best performance also changes as the vehicle navigates the terrain (see Fig. 6). In order to address this problem, our method acts online by taking immediate feedback into consideration when choosing which transfer function to use to predict the next state.

Prior to the online component, an ensemble of experts is initialized offline. Each expert consists of a pair of transfer functions, one for velocity and one for yaw generated from a specific friction and slope, and a kinematics model (in this case they are all the same bicycle model). The experts take in an observed state (x, y, θ) and command (v, δ), feed it through their corresponding transfer function and models, and output a prediction for the next state. A vector of weights, one for each expert, is also initialized so that all experts start with the same weight.

The online component runs with each step in simulation. In a given step, all experts output a prediction based on the current state and command, and a pre-determined policy is used to select an expert based on their current weights. The next state is then observed, and a loss vector is generated by calculating the L2 distance between each expert's predicted state and the ground truth state (x, y, θ). This loss vector is used to update the weights vector based on the policy, and the new weights are used to make the decision in the next simulation step. For the Randomized Weighted Majority (RWM) policy [14], *all* weights are updated as follows:

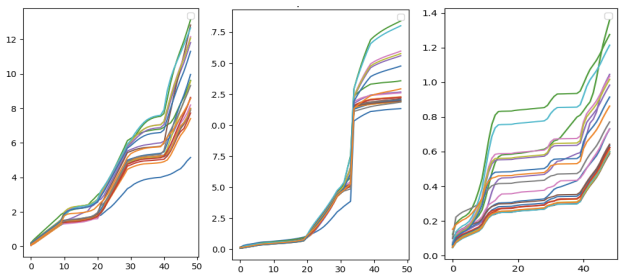

Fig. 6. The cumulative losses of various experts over the past 50 steps at different points in time. Loss is defined as the distance between the predicted state and the ground truth state.

$$W[i] = W[i] * e^{-\eta * l[i]} \tag{1}$$

where W is the weights vector, i is the index of the expert in the ensemble, η is a fixed hyper-parameter, and l is the loss vector. The weights are normalized to sum to 1 after each update. For the Generalized Weighted Majority (GWM) policy [14] only the weight for the chosen expert is updated:

$$W[i] = W[i] * e^{-\sqrt{log(S/T)} * l[i]} \tag{2}$$

where S is the number of experts, and T is the total number of steps that have occurred. For the Exponential-weight algorithm for Exploration and Exploitation (EXP3) policy [15] again only the weight for the chosen expert is updated:

$$p = \frac{W[i]}{\sum W} \tag{3}$$

$$\eta = \sqrt{\frac{log(S)}{T * S}} \tag{4}$$

$$W[i] = W[i] * e^{-\eta * l[i]/p} \tag{5}$$

All the policies implemented so far have the same decision process for choosing an expert, which involves using the weights as a probability vector and making a random decision based on those probabilities. However we did also use another strategy where, instead of choosing an expert, a new prediction is generated by taking the weighted average of all the expert predictions. These policies allow the learner to use the effectiveness of past predictions in the decision process for current predictions.

IV. RESULTS

As established earlier, there is not one transfer function that consistently results in the best predictions. Since the loss of each expert relative to the others changes over time (Fig. 6), the improvement our method provides can be indicated both by observing the distribution of expert predictions, as well as by comparing it to the baseline method (using a single transfer function for all predictions). Fig. 7 shows examples of the predictions of all experts in 2D space over time. As the vehicle moves across terrain in simulation, we observed that there are some areas with a small spread of predictions as well as areas with a wide distribution of predictions. Even when the distribution gets wider, the learner's predictions stay close to the ground truth, which shows the benefit of learning off of the varying loss of the experts relative to one another. For example, Fig. 8 shows how using a single

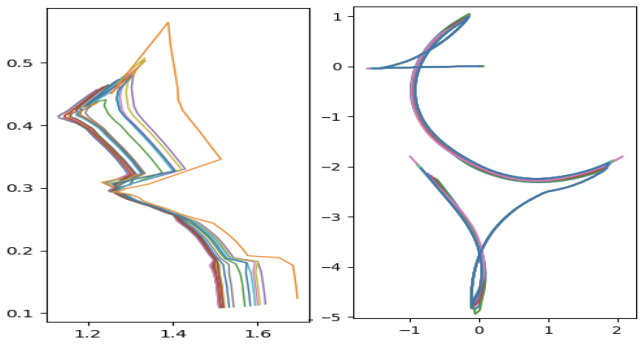

Fig. 7. Predictions of all experts in 2D space as the vehicle traverses terrain in two separate experiments.

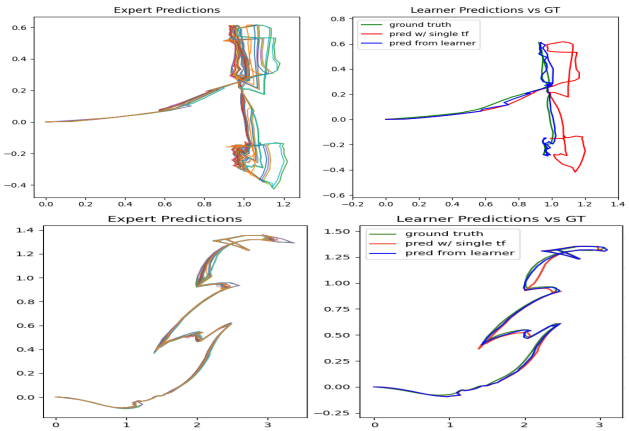

Fig. 8. Left: the distribution of all expert predictions in 2D space; Right: the accuracy of predictions from the learner compared to using a single transfer function.

transfer function for each prediction results in some areas with inaccurate predictions compared to using a learned prediction.

Additionally, the regret of the learner over time is observed, where regret is defined by the difference between the learner's cumulative loss and the minimum possible cumulative loss. Fig. 9 shows the regret of the learner over time in two different experiments. Decreasing regret values indicate good performance of the learner, since this corresponds to being as accurate as the best expert at any given time. The spikes of increased regret at various points correspond to changes in the terrain. This can be explained by the temporal nature of the learner, in that it will have higher weights for the expert that has been the most accurate recently. Therefore, when the terrain changes suddenly, the best expert changes drastically and the learner needs time to adjust accordingly.

Fig. 10 shows the performance of the different policies we experimented with. The more jagged loss of the GWM and EXP3 policies are likely explained by the fact that only one expert's weight is adjusted in each time step, compared to the smoother changes in loss from the RWM and weighted average resulting from all weights being updated at every step. Overall, the weighted average policy consistently outperformed the other policies.

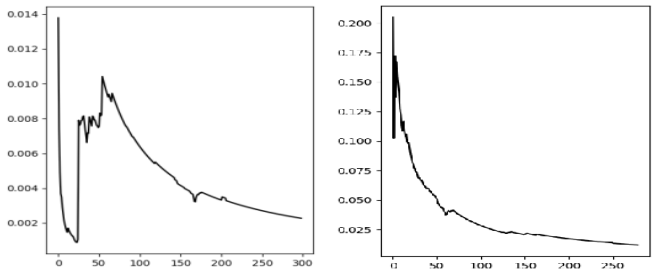

Fig. 9. The regret of the learner over time in two different experiments.

V. DISCUSSION AND FUTURE WORK

In this work we first establish that a wide range of transfer functions can be used to collectively represent the results of a vehicle driving in uneven terrain with variable friction, and that their individual performances vary as the terrain varies. We then demonstrate that online ensemble methods can be used to provide an improvement over using a single model by taking into account previous rewards when making new decisions. This behavior allows it to perform at least almost as well as the best expert, which varies greatly over time, at any given step.

Our immediate next step involves validating our method on a physical platform. We have an all-terrain vehicle (ATV), shown in Fig. 11 fitted with the same sensors that we setup our simulator. We plan to collect an ensemble of transfer functions on the ATV, and see if we observe the same improvement from online ensembles in order to show that our contribution has real-life applications.

Our method is also lightweight enough to integrate into downstream tasks, like planning and control. There are both planners, like kinodynamic RRT*, and controllers, like MPC, that work best when they have highly accurate vehicle models. They involve forward-sampling the vehicles state over multiple steps, so any error in the model ends up causing a drift in the predictions from the ground truth. In theory, the improvement that the online ensemble provides should help reduce this drift.

One limitation however, is its reliance on systematically collected data, which is difficult to accomplish in real-life. One way to address this is by collecting fewer transfer functions and just interpolating their parameters to artificially

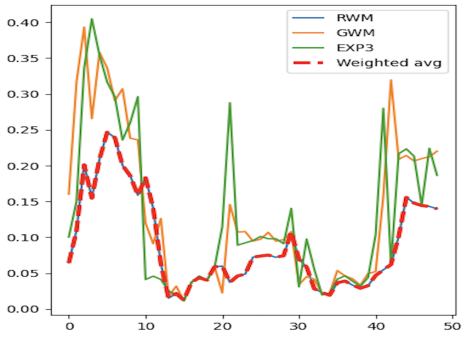

Fig. 10. The loss of the various learner policies over time.

Fig. 11. Our physical platform for real-life testing

generate more. However, it would probably be better to incorporate some approach that doesn't require any prior knowledge of the environment at all. This way, the ensemble would be agnostic to the range of properties in a given environment.

ACKNOWLEDGMENT

This work was supported by the Robotics Institute Summer Scholars program at Carnegie Mellon University. The authors would like to thank Rachel Burcin and Dr. John Dolan for their support and organization of this program.

REFERENCES

[1] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 709–715.

[2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08*, 2008.

[3] D. Ferguson and M. Likhachev, "Efficiently using cost maps for planning complex maneuvers," in *in Proc. of the Workshop on Planning with Cost Maps, IEEE Int. Conf. on Robotics and Automation*, 2008.

[4] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[5] M. Zambelli and Y. Demirisy, "Online multimodal ensemble learning using self-learned sensorimotor representations," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 2, pp. 113–126, 2017.

[6] R. Zou, Y. Liu, G. CHU, J. Zhao, and H. CAI, "Online active ensemble learning for robot collision detection in dynamic environments," *Journal of Mechanics in Medicine and Biology*, vol. 21, p. 2150035, 05 2021.

[7] A. S. Panda, R. Prakash, L. Behera, and A. Dutta, "Combined online and offline inverse dynamics learning for a robot manipulator," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.

[8] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.

[9] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, "Learning dynamics model in reinforcement learning by incorporating the long term future," 2019.

[10] N. Takeishi and Y. Kawahara, "Learning dynamics models with stable invariant sets," 2020.

[11] J.-F. Tremblay, T. Manderson, A. Noca, G. Dudek, and D. Meger, "Multimodal dynamics modeling for off-road autonomous vehicles," 2021.

[12] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[13] *Practical Methods of Optimization*. John Wiley & Sons, Ltd, 2000, pp. i–xvii. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118723203.fmatter

[14] N. Littlestone and M. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–261, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890540184710091

[15] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002. [Online]. Available: https://doi.org/10.1137/S0097539701398375