Predictive Neural Network Models For Autonomous Vehicle Driving on Multiple Terrains

Subhransu Mishra[1], Gowtham Garimella[1], Christoffer Heckman[2], and Marin Kobilarov[1]

***Abstract*— This work considers the problem of predicting the dynamics of a small unmanned ground vehicle (UGV) navigating on unknown terrains. Predicting the dynamics over long horizons accurately helps avoid obstacles by making accurate control decisions early on. We propose a Recurrent Neural Network (RNN) to predict the position and orientation of the vehicle driving at 8 meters per second (18 mph) up to a time horizon of 2 seconds. Our method is different from traditional dynamic models of a car in the sense that we do not explicitly encode the terrain dependent parameters into the model. In contrast, the RNN model can estimate both the type of the terrain and the terrain parameters by processing through a few steps of sensor data such as GPS, IMU and wheel encoders. We tested the network on a 1/5th scale RC car driving on three terrain surfaces namely grass concrete and sand which have vastly different dynamic behaviors. Our network is able to make position predictions with an accuracy of 0.5 m and orientation of the car with an accuracy of 7 degrees in the next two seconds. In addition, the network is also able to classify the terrain accurately 74% of the time (3 terrains considered) where random chance would predict at 33%. The dynamic model created in this work can be used in predictive control schemes such as Model Predictive Control (MPC) or to simply determine the safe driving conditions of a vehicle based on the terrain.**

I. INTRODUCTION

This paper considers the autonomous navigation of small Unmanned Ground Vehicles (UGV) on natural terrains, motivated by applications such as package delivery, surveillance, search and rescue operations [1], [2], [3] in environments that do not necessary have regular roads that are easily traversable. For instance, consider a ground vehicle transitioning between two different surfaces or driving aggressively on a terrain (such as in Fig. 1). In order to safely navigate these terrains and reach a goal state, the vehicle needs an accurate model that can predict it's motion for a long horizon. The terrain conditions often introduce sudden changes in the tire-to-surface interaction that, if not modeled, can lead to excessive lateral accelerations, wheel slip and possible loss of control. In such scenarios, it is necessary to accurately detect the terrain properties online and predict the motion of the car over a sufficiently long future horizon that ensures obstacle avoidance or safe stopping. We propose a neural network model for identifying the dynamical model of a UGV, which can represent the vehicle motion faithfully over a long horizon by automatically recognizing the driving terrain online based on sensor data.

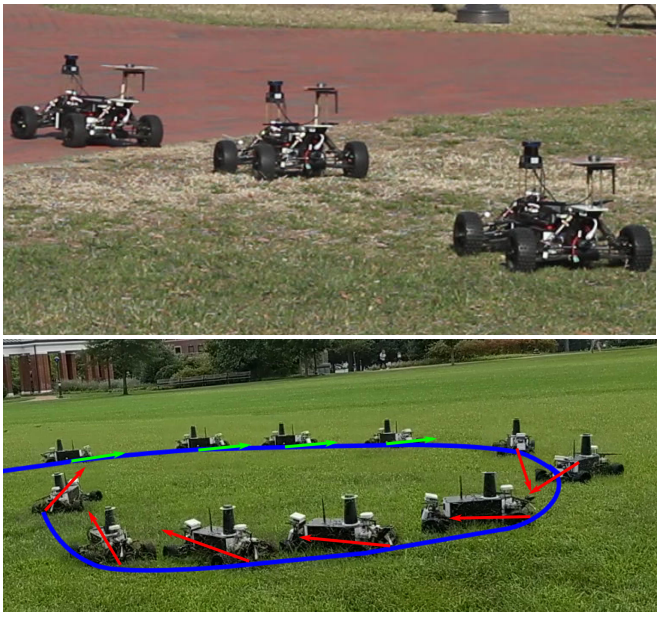

Fig. 1. A small (1/5 scale) ground vehicle driving on different terrains. The top figure shows the car transitioning between grass and brick surfaces. The bottom picture shows car sliding over a grass terrain with red arrows indicating sideways slip.

A. Related Work

Earlier work concentrated on modeling the dynamics of a four wheeled robot using first principles models [4]. Using the first principles models, several control approaches have been developed that control the vehicle on flat terrains under different slip conditions [5], [6], [7]. These models, although sufficient for a flat terrain with fixed terrain parameters, cannot capture the full nonlinear dynamics of a car on terrains with varying conditions. Simple first principles models have also been developed for performing control on rough terrains [8] at low speeds. Initial control frameworks on these terrains were restricted to low speeds where the dynamics was predictable using the simplified models [9]. Aggressive driving on different terrains has been made possible by combining improved robot dynamics modeling with better control approaches. Several authors considered high fidelity physics simulators to predict the car dynamics [10], [11]. The physics-based models are used in either a Model Predictive Control (MPC) setting [12] or in a reinforcement learning approach where a control

[1]Subhransu Mishra, Gowtham Garimella, and Marin Kobilarov are with the Department of Mechanical Engineering at Johns Hopkins University, Baltimore, MD, USA. smishra1|ggarime1|marin@jhu.edu

[2]Christoffer Heckman is with the Computer Science Department at University of Colorado, Boulder, USA. christoffer.heckman@colorado.edu

policy is found using a transfer learning approach [13], [14]. These methods require an accurate representation of the terrain which can be constructed from LiDAR data as shown in [15], [16], [17]. Tuning the parameters for a high fidelity simulation model is usually hard since it is dependent on the specific terrain and the vehicle being used. Recent work done by Aghli and Hackerman [18] have shown ability to tune these parameters by choosing selected motion segments with maximum information regarding the parameters.

Unlike physics-based models, neural networks have also been used to learn the dynamics of robots [19], [20]. These methods combine traditional first principles models with neural network models to precisely predict the dynamics of the robot. The neural network models have been combined with MPC techniques to control robots under uncertainty [21], [22]. The authors were not able to find any current literature where neural network methods are employed to learn the dynamics of a mobile robot on multiple terrain surfaces.

Instead of using neural networks for learning dynamics, end to end control policies which take in sensor data and produce a control output have also been developed for aggressive driving. Sallab et al. [23] showed that deep reinforcement learning can be applied to race car driving in simulation. Yunpeng et al. [24] further showed that deep neural networks can be used to learn control policies for driving aggressively at 8 m/s on a 1/5th scale RC car using only a monocular camera and wheel encoders. Convolutional neural networks (CNN) have been used to predict costs for MPC optimization using a vision sensor in [25]. Our approach, in contrast to above methods, learns an accurate model over multiple terrain surfaces. The learned model is the first step in developing a model based control scheme which can use a neural network control policy or an MPC policy.

Neural networks have also been used to classify the terrain type based on cameras [26], LiDAR [27], and a subset of camera, Lidar and vibrational sensors [28], [29], [30]. These approaches try to avoid the terrains that are hard to drive on rather than learning the dynamics on those terrains. In this work, we try to predict the dynamics of the car on different terrains and, therefore, allow for safe navigation of the vehicle on multiple terrains.

B. Contributions

We develop a general Recurrent Neural Network (RNN) architecture that is not specific to any particular vehicle model or terrain type and is expected to adapt to different environments. The network is applied to predicting the dynamics of the ground vehicle shown in Figure 1 traversing two terrain types: grass and brick. The terrains are assumed to be relatively flat and thus the dynamics of the car can be captured using planar (as opposed to 3D) relative pose displacements, gyro and acceleration data, encoder odometry of each wheel, and control inputs. We show that with sufficient training data across different velocities and maneuver types it is possible to learn a model that can predict the motion equally well irrespective of the terrain surface type, e.g. the network can predict the position of the vehicle 2 seconds in the future traveling at 7 m/s with 0.6 cm accuracy on average. It is shown that training the network using data from both terrains types is critical for achieving such performance. In addition we show that RNN can be made to encode the terrain type explicitly using controls and sensor measurement and not rely on vision based sensors. We tested with 3 different terrains types and obtained an accuracy of 75% when driving with high lateral acceleration or high forward acceleration.

.

Measurement	Source	Accuracy	Frequency (Hz)
Position	GPS	0.2 m	5
Forward Velocity	Motor Encoder	0.1m/s	50
Wheel velocities	Wheel encoders	2°	50
Body Z angular velocity	MEMS-Gyro	0.1rad/s	100
Body X, Y angular velocity	MEMS-Accelerometer	0.1m/ss	100
Steering angle	Servo-motor	3°	50

TABLE I

THE TABLE SHOWS THE SENSOR MEASUREMENTS USED BY THE NETWORK, THE SOURCE SENSORS, THE MEASUREMENT ACCURACY AND THE ACQUISITION FREQUENCY

II. Neural Network Architecture

In this work we use a Recurrent Neural Network (RNN) to predict the behavior of the car. We are interested in predicting the pose of the car g which is an element of $SE(2)$. The pose is fully defined by the position ($x \in \mathbb{R}, y \in \mathbb{R}$) of the center of the rear axle of the car as well as the orientation ($\theta \in \mathbb{S}^1$) of the car. The network is also designed to perform other auxiliary tasks which may help in predicting the pose g better. The auxiliary tasks performed by the network are to predict the terrain its driving on and predicting a set of sensor measurements $z \in \mathbb{R}^{n_z}$, where n_z is the sensor dimension. The terrain prediction is performed by learning a probability distribution vector ψ on a set of pre-defined terrain labels S_ψ and the label with maximum probability is chosen as the predicted terrain label.

A. Network inputs and outputs

1) Outputs: RC car is equipped with a suite of sensors with varying accuracies and data collection frequencies as shown in Tab I. The pose measurements $\hat{g}$ are obtained by concatenating the position measurements (x, y) from GPS and the orientation θ of the car from yaw measurement of the IMU. The auxiliary sensor measurements $\hat{z}$ are chosen as front left and right wheel encoders as well as z angular velocity obtained from IMU ($\hat{z} \in \mathbb{R}^3$). These sensor measurements have been chosen by experimenting with different combinations of sensors and finding a set that results in good prediction performance.

2) Inputs: The recurrent network takes as input the current sensor measurements $\hat{z}_i$ and the current controls u_i. The vehicle is controlled using an Electronic Speed Control (ESC) to control the longitudinal velocity of the vehicle

$v \in \mathbb{R}$ and a servo to control the steering angle $\phi \in \mathbb{R}$ of the vehicle. The control inputs to the vehicle are given by the desired velocity and steering angle $v_d \in \mathbb{R}, \phi_d \in \mathbb{R}$ i.e $u = [v_d, \phi_d]^T$. The car is controlled at a frequency of 50 Hz while the position measurements are received at a frequency of 5 Hz. If we supply the network with only a single control input, the network loses predictive capacity. To avoid the loss in control information, we fit a second order polynomial $(u_p(t) = c_0 + c_1 t + c_2 t^2)$ to the controls between (t_i, t_{i+1}) where $t \in \mathbb{R}$ represents the time and $c_{0:n} \in \mathbb{R}^2$ corresponds the polynomial coefficients. We feed the stacked coefficients $c_i = [c_{0,i}^T, c_{1,i}^T, c_{2,i}^T]^T$ to the neural network at each step as shown in Fig 2. Assuming the controls between the times are given by $u_0, \cdots, u_{N_u}$(ignoring the subscript i for clarity in notation), the coefficients for the second order polynomial are obtained by performing a least squares fit as

$$\min_{c_{0:2}} \sum_{j=0}^{N_u} \|c_0 + c_1 t_j + c_2 t_j^2 - u_j\|_2^2 \tag{1}$$

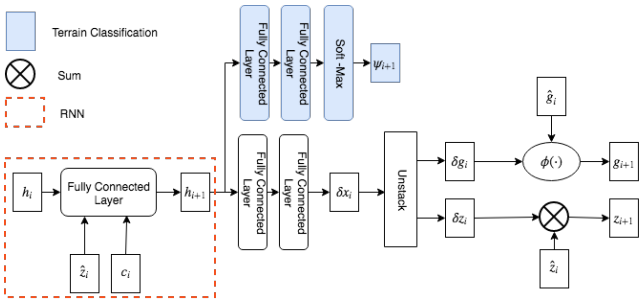

Fig. 2. Neural Network Architecture for predicting the next car pose g_{i+1} given the current sensor measurements $\hat{z}_i$, control coefficients c_i and the hidden state h_i. The transform function ϕ applies a relative pose δg to current pose g.

B. Predicting relative pose and change in sensor measurements

A naive implementation of the neural network involves predicting the pose of the RC car at the next time step g_{i+1} directly using the pose measurements, controls and other auxiliary sensor measurements at the current step $(\hat{g}, \hat{z}, u_i)$. This approach has two main issues. First, the position coordinates x, y have a very large output scale. When the workspace is very large, a large network size is required to predict the position coordinates accurately. Further, the training data is required to cover the entire position space uniformly which increases the size of the training data drastically.

The second issue has to deal with angle measurements which lie on S^1. In our example, the orientation of the car wraps around 2π, and, thus, there are jumps in the predictions which are not easy to model. The common solution to this problem is to embed the angles in a higher dimension such as quaternions [31]. This adds additional dimensions to the state.

In this work, we use the invariance of the car dynamics [32] to mitigate the issues mentioned above. The invariance in the dynamics allows for us to predict the *change* in the pose $\delta g_i \in \mathbb{R}^3$ and change in sensor measurements δz_i which combined with $\hat{g}_i$ and $\hat{z}_i$ predicts g_{i+1}, z_{i+1} as explained in [33], without explicitly predicting the pose and measurements.

The overall architecture of the neural network is shown in Fig 2. The RNN part of the network is shown in dashed lines. The recurrent network takes in the sensor measurements $\hat{z}_i$, control polynomial coefficients c_i and the hidden state h_i and predicts the hidden state at next step h_{i+1}. The hidden state represents information accumulated over time and is equivalent to the state of the system in a traditional dynamic model. The size of the hidden state has been experimented with and is chosen as 20 dimensional state ($h_i \in \mathbb{R}^{20}$).

The hidden state is then passed through a three fully connected layers with 20 nodes each to obtain the change in pose δg_i and the change in sensor measurements δz_i as $\delta x_i = [\delta g_i^T, \delta z_i^T]^T$. The change in pose is then then applied to the measured pose to obtain the predicted pose at next step as

$$g_{i+1} = \phi(\hat{g}_i, \delta g_i) = g_i \exp(\delta g_i), \tag{2}$$

where the exponential function maps a vector from R^3 to SE(2). The predicted sensor measurements live on a Euclidean manifold ($z \in \mathbb{R}^3$) and are obtained by simply adding the change in sensor measurements to the previous sensor measurements i.e $z_{i+1} = \hat{z}_i + \delta z_i$.

C. Terrain Classification

The neural network is also trained to predict the terrain on which the vehicle is driving based on the hidden state from the RNN. By training on this auxiliary task, we force the hidden state to learn the representation of the terrain as part of the hidden state. The hidden state h_{i+1} is processed through a couple of separate hidden layers and finally a softmax layer to get the normalized probability vector ψ_{i+1}. The maximum element in the vector corresponds to the predicted terrain label.

III. Training

The network is trained on a series of trajectories collected by driving the car manually over different surfaces. We used four different surfaces in this work: grass, turf, concrete and sand. For each of the surfaces, we varied the desired velocities from 0 m/s to 10 m/s and the steering command from -20 degrees to 20 degrees.

The network training is performed by minimizing a training cost L that consists of minimizing the error between predicted pose g_{i+1} and measured pose $\hat{g}_{i+1}$, the error between predicted sensor measurements z_{i+1} , measured sensor measurements $\hat{z}_{i+1}$ and a cross entropy loss minimizing the error in the terrain probability distribution vector ψ_{i+1}. The

loss function is given by

$$L = \sum_{i=1}^{N} \Big(\|Q_{g_i}(g_{i+1} - \hat{g}_{i+1})\|_2^2 + \|Q_{z_i}(z_{i+1} - \hat{z}_{i+1})\|_2^2 + Q_{CE}\ \text{CE}(\hat{\psi}_{i+1}, \psi_{i+1}) \Big), \tag{3}$$

where $Q_{(\cdot)}$ is the weighting matrix that scales the importance of each of the tasks, CE represents the cross entropy loss and $\hat{\psi}_{i+1}$represents a one-hot vector with the index corresponding to label getting a value of one. The weight Q_{g_i} is further scaled inversely by length of the relative motion δg_i to normalize the errors across different velocities. Since the pose of the car g lives in $SE(2)$, the subtraction "$-$" between the poses $g_{i+1}, \hat{g}_{i+1}$ in equation (3) refers to the operator that finds the distance between two elements in $SE(2)$.

The cost function (3) is minimized using the truncated back propagation algorithm where the car trajectories are truncated into segments of 5 steps(1 second) and the gradients of the network weights are computed based on back propagation up to the segment length. Although the gradients are truncated at the segment length, the hidden state is propagated across continuous segments to ensure the terrain information and vehicle velocities and accelerations are not lost.

A. Testing

During the testing phase, the network predicts the pose of the car for the next 2 seconds based on the current hidden state h_0 and controls control polynomial coefficients $c_{1:N}$. Since we do not know the future pose and sensor measurements $\hat{z}_{1:N}, \hat{g}_{1:N}$, we replace the measured quantities with predicted quantities i.e $z_{1:N}, g_{1:N}$ in network architecture shown in Fig 2. Accurate prediction of car dynamics based on predicted poses and sensor values requires an accurate initialization of hidden state. The hidden state is initialized by running the network with past sensor measurements and past controls for a fixed period of time to ensure the hidden state is converged. The initialization time is determined based on average time observed in training for the predicted sensor measurements to catch up with observed sensor measurements starting from an zero initialized hidden state.

IV. Results

A. Self driving Platform

The 1/5th scale electric RC car has stock mechanical linkages, drive train and servo motors. The radio receiver, servo motor controller and electronic speed control have been upgraded to talk to an ATmega-2560 based controller board using UART interface. The front two wheels are equipped with magnetic encoders . The steering servo and drive motor are powered by a 4000 mAh 6 cells LiPo battery. The enclosure and mounting brackets are made with laser-cut acrylic plates and 3D printed parts. The computer sub-system includes an Intel quad-core i7 processor with 8 GB of RAM, an ethernet switch and wifi router. It is powered by a 4000mAh 6cell LiPo battery. The software for collecting data and running MPC is setup using the Robot Operating System(ROS) framework. ROS also facilitates communication between computer and low level controller along with time synchronization. The computer sends drive motors rpm and steering wheel angle commands to the low-level controller which is then relayed to the ESC and the servo controller for closed-loop/set-point control. The low level controller sends back sensors information. Localization is performed by 2D LiDAR based localization algorithm or through a combination of IMU and RTK GPS. The neural network algorithm is setup using the TensorFlow python API.

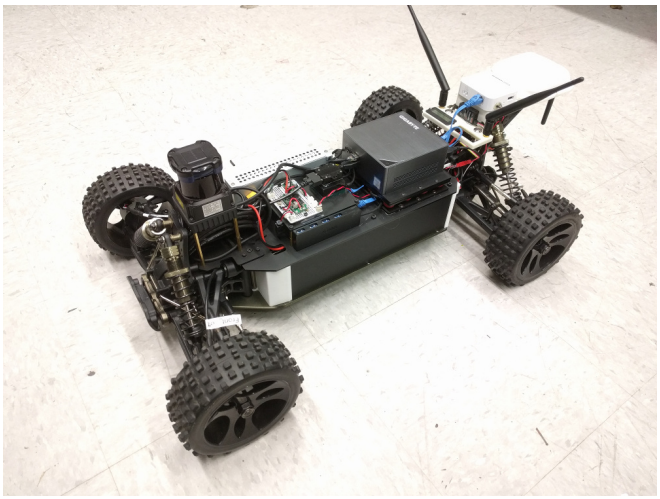

Fig. 3. Assembled car

B. Data

Data from two distinct surface types was collected by manually driving the car at various speeds and angular velocities. While driving the vehicle, care is taken to span the space of controls and sensor measurements as best as possible .The car experiences a maximum lateral acceleration of $8m/sec^2$, forward acceleration of $4m/sec^2$, an angular velocity of $1.2rads/sec$ and forward speeds up to 8 m/sec .The data is divided into training, validation and test set. The training set is used to optimize the neural network parameters. Model selection and tuning is done using the validation set. The plots in the results sections is obtained by running the learned models on the test set.

C. Discussion

The network was trained separately on grass, brick and combined training data set and their performance was evaluated separately on the grass and brick data set. The results are tabulated in table IV-C. The columns of the tables show the root mean squared(RMS) error of the relative pose prediction performed over the same dataset. Lets consider the case where we train the network on brick and combined dataset, separately. The prediction with combined trained network is better than the brick trained model. This indicates that the network learns the terrain parameters without explicitly specifying the terrain, which is evident from the discussion

in the next paragraph. The network is able to make accurate predictions over multiple steps. This is presented in fig 6. Lastly the figure 4 plots the RMS error in angle, x and y position for trajectories sampled from the set with speed greater than 7 m/sec. That gives us an RMS error of 0.12 rad, 0.5 m in x and 0.2 m in y over 2 seconds.

In the Fig 5 we show that the hidden state of the RNN can indeed encode the explicit terrain type just based on the controls inputs and sensor feedback. We collect driving data with high lateral and forward acceleration on three terrains: Turf, Sand and Concrete to train the RNN with auxiliary task. The rows show the true labels and the columns show the predicted labels. For example, the first row shows that out of 286 concrete sample trajectories, 211 samples were labeled correctly.

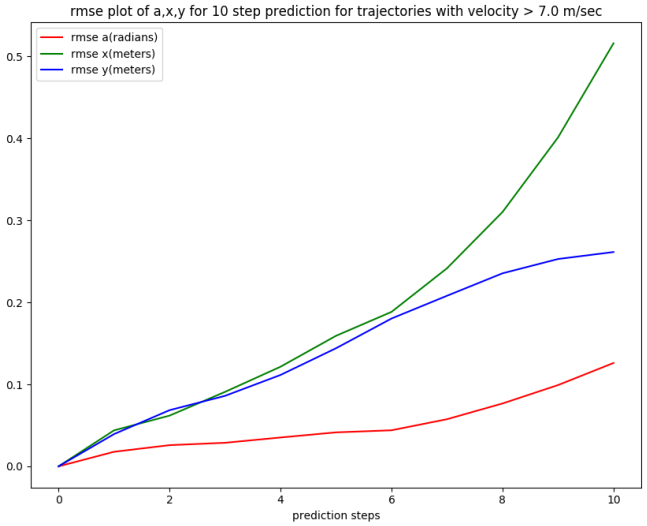

Fig. 4. RMS error of trajectories with minimum speed of 7 m/sec over 2 secs (10 steps)

V. Conclusions

We have shown that the proposed RNN architecture is able learn the dynamics of an RC car accurately without any prior knowledge of the terrain. The network is also able to classify the terrain only based on wheel encoder and GPS data without the aid of vision based sensors. The high accuracy obtained using the RNN model allows for navigating through different terrains safely. Combining the RNN architecture with an MPC architecture to autonomously navigate through multiple terrains is the subject of future work.

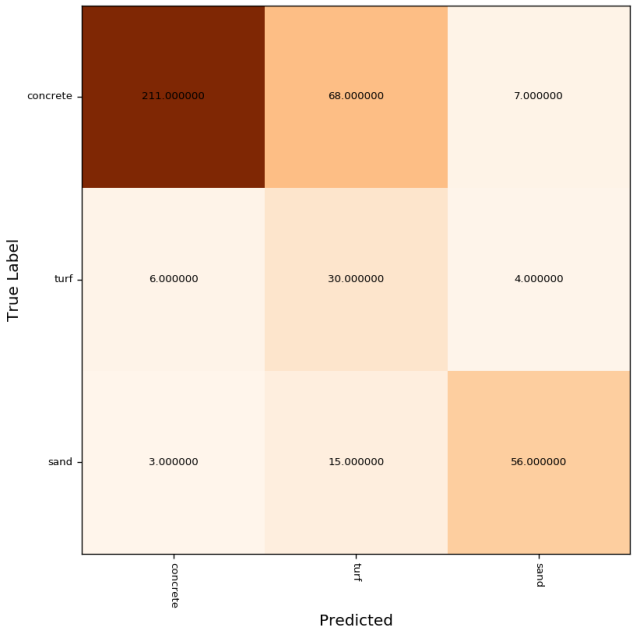

Fig. 5. Confusion matrix for terrain classification over three different terrains namely concrete, turf and sand.

References

[1] S. Karma, E. Zorba, G. Pallis, G. Statheropoulos, I. Balta, K. Mikedi, J. Vamvakari, A. Pappa, M. Chalaris, G. Xanthopoulos, *et al.*, "Use of unmanned vehicles in search and rescue operations in forest fires: Advantages and limitations observed in a field trial," *International journal of disaster risk reduction*, vol. 13, pp. 307–312, 2015.

[2] G. De Cubber, D. Doroftei, D. Serrano, K. Chintamani, R. Sabino, and S. Ourevitch, "The eu-icarus project: developing assistive robotic tools for search and rescue operations," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE international symposium on*, pp. 1–4, IEEE, 2013.

[3] J. Carlson and R. R. Murphy, "How ugvs physically fail in the field," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 423–437, 2005.

[4] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[5] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous audi tts," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 541–547, IEEE, 2012.

[6] R. Gonzalez, F. Rodriguez, J. L. Guzman, C. Pradalier, and R. Siegwart, "Control of off-road mobile robots using visual odometry and slip compensation," *Advanced Robotics*, vol. 27, no. 11, pp. 893–906, 2013.

[7] R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "Model predictive control for vehicle guidance in presence of sliding: application to farm vehicles path tracking," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 885–890, IEEE, 2005.

[8] K. Iagnemma and S. Dubowsky, "Chapter 4 rough terrain control," *Mobile Robots in Rough Terrain*, pp. 81–96, 2004.

[9] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics* (A. Howard, K. Iagnemma, and A. Kelly, eds.), (Berlin, Heidelberg), pp. 69–78, Springer Berlin Heidelberg, 2010.

[10] K. Iagnemma, F. Genot, and S. Dubowsky, "Rapid physics-based rough-terrain rover planning with sensor and control uncertainty," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3, pp. 2286–2291, IEEE, 1999.

[11] A. S. Aquilio, J. C. Brooks, Y. Zhu, and G. S. Owen, "Real-time gpu-based simulation of dynamic terrain," in *International Symposium on Visual Computing*, pp. 891–900, Springer, 2006.

[12] S. L. N. Keivan and G. Sibley, "A holistic framework for planning, real-time control and model learning for high-speed ground vehicle navigation over rough 3d terrain," in *IROS*, 2012.

[13] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 655–671, 2015.

[14] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*, pp. 593–600, ACM, 2005.

[15] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and

Testing →		Brick			Grass		
Training ↓		$rmse(\delta g_\theta)$(rad)	$rmse(\delta g_x)$ (m)	$rmse(\delta g_y)$ (m)	$rmse(\delta g_\theta)$(rad)	$rmse(\delta g_x)$ (m)	$rmse(\delta g_y)$(m)
Brick		0.026	0.038	0.034	0.129	0.073	0.088
Grass		0.102	0.060	0.065	0.034	0.035	0.039
Brick+	Grass	0.065	0.051	0.047	0.051	0.039	0.053

TABLE II

RMS ERROR COMPARISON FOR HIGH SPEED DATASET USING RNN

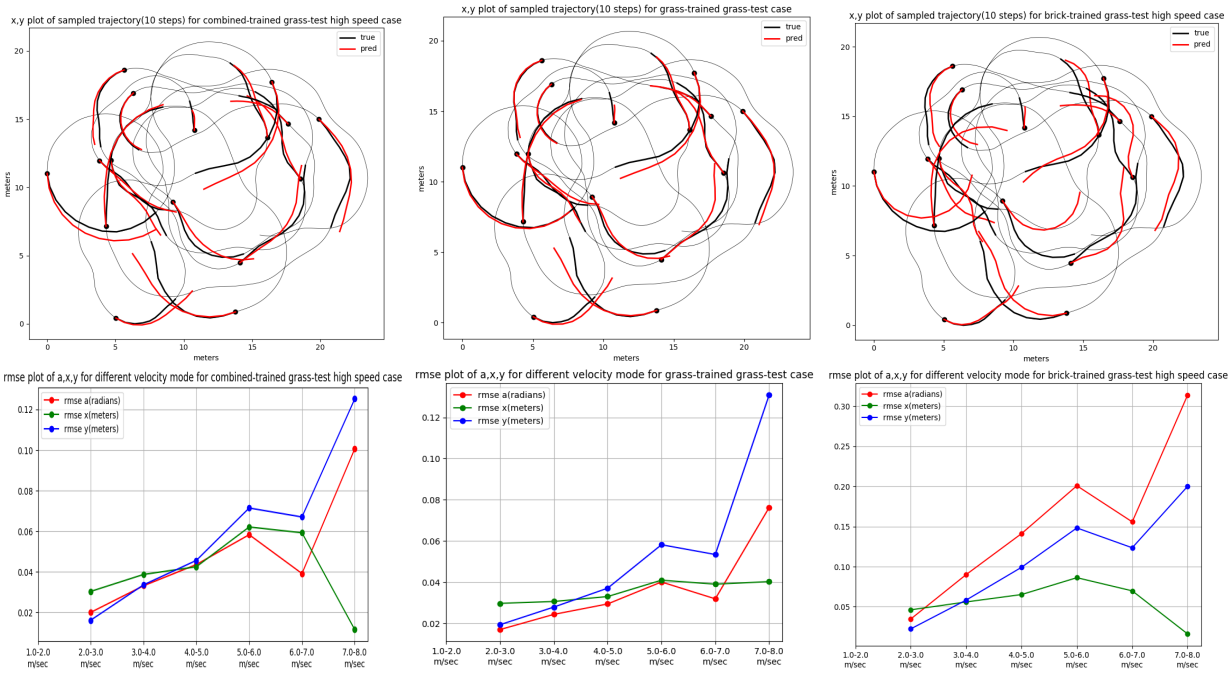

Fig. 6. The top figures show the predicted and observed trajectories when driving on grass terrain using different network configurations. The bottom figures show the RMS errors of the network configurations against longitudinal velocity. The network trained only on brick dataset does not generalize well to grass dataset.

B. Gerkey, *Outdoor Mapping and Navigation Using Stereo Vision*, pp. 179–190. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[16] J. Madsen, A. Seidl, and D. Negrut, "Compaction-based deformable terrain model as an interface for real-time vehicle dynamics simulations," tech. rep., SAE Technical Paper, 2013.

[17] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3d navigation mesh generation for path planning in uneven terrain," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 212–217, 2016.

[18] S. Aghli and C. Heckman, "Online system identification and calibration of dynamic models for autonomous ground vehicles," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, IEEE, 2018.

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1433–1440, IEEE, 2016.

[20] G. Garimella, J. Funke, C. Wang, and M. Kobilarov, "Neural network modeling for steering control of an autonomous vehicle," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2609–2615, Sept 2017.

[21] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control.," in *Robotics: Science and Systems*, 2015.

[22] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

[23] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[24] K. S. K. L. X. Y. E. A. T. B. B. Yunpeng Pan, Chign-An Cheng, "Agile autonomous driving using end-to-end deep imitation learning," in *Robotics Science and Systems*, vol. 14, 2018.

[25] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg, "Aggressive deep driving: Combining convolutional neural networks and model predictive control," in *Proceedings of the 1st Annual Conference on Robot Learning* (S. Levine, V. Vanhoucke, and K. Goldberg, eds.), vol. 78 of *Proceedings of Machine Learning Research*, pp. 133–142, PMLR, 13–15 Nov 2017.

[26] Y. N. Khan, P. Komma, and A. Zell, "High resolution visual terrain classification for outdoor robots," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1014–1021, Nov 2011.

[27] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of field robotics*, vol. 23, no. 10, pp. 839–861, 2006.

[28] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt, "Geometric and visual terrain classification for autonomous mobile navigation," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 2678–2684, IEEE, 2017.

[29] K. Iagnemma and C. C. Ward, "Classification-based wheel slip detection and detector fusion for mobile robots on outdoor terrain," *Autonomous Robots*, vol. 26, pp. 33–46, Jan 2009.

[30] C. A. Brooks and K. Iagnemma, "Self-supervised terrain classification for planetary surface exploration rovers," *Journal of Field Robotics*, vol. 29, no. 3, pp. 445–468, 2012.

[31] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Computer Vision*

(ICCV), 2015 IEEE International Conference on, pp. 2938–2946, IEEE, 2015.

[32] M. Kobilarov, *Discrete geometric motion control of autonomous vehicles*. University of Southern California, 2008.

[33] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4692–4697, IEEE, 2015.