Learning the Forward Predictive Model for an Off-Road Skid-Steer Vehicle

Michael Bode

CMU-RI-TR-07-32

September 2007
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

A forward predictive model is used to simulate a vehicle's motion given a sequence of commands that could potentially be executed. Generally, forward predictive models are used by planning systems for Unmanned Ground Vehicles (UGV's) so that commands can be selected such that obstacles are avoided. This report presents a data-driven approach for learning a forward predictive model based on previously recorded vehicle motion. The selected approach is compared to several variations including the conventional forward predictive model that has traditionally been used on the Crusher vehicle. Results are presented using real life data collected on the Crusher UGV.

Contents

1 Introduction

Recent advances in robotic ground vehicles have led to the autonomous traversal of increasingly complex terrain at higher rates of speed. With these advances the motion of a vehicle caused by a particular drive command becomes progressively more difficult to predict. Solving this problem is essential for selecting commands to avoid collisions with obstacles. Most existing off-road robotic vehicles [1, 2] have used physics-based models to predict the effect of issuing a particular command. Learning approaches have been applied to forward predictive modeling [3]; but generally these approaches have been used to learn terrain characteristics (height of supporting surface, soil cohesion, etc.) that are then used as input to physics-based models. In this paper, an alternate approach to learning the forward predictive model that does not rely on a physics-based model is pursued. This is advantageous because it makes no assumptions about vehicle/terrain interaction, but rather learns based on experience how the vehicle will react while operating on the terrain outright.

The forward predictive model is used to simulate where the vehicle would go if issued a particular set of commands. Typically, it predicts vehicle velocities at the end of a specific time interval. These velocities are then integrated to predict the new vehicle position. The model can be queried again using the newly predicted vehicle velocities and position to predict the vehicle's position two time-steps in the future. This process can be iterated for as long as desired to produce the expected vehicle trajectory. As shown in Figure 1, a planning system can then evaluate the trajectories generated from several candidate command sequences and decide which sequence of commands to execute.

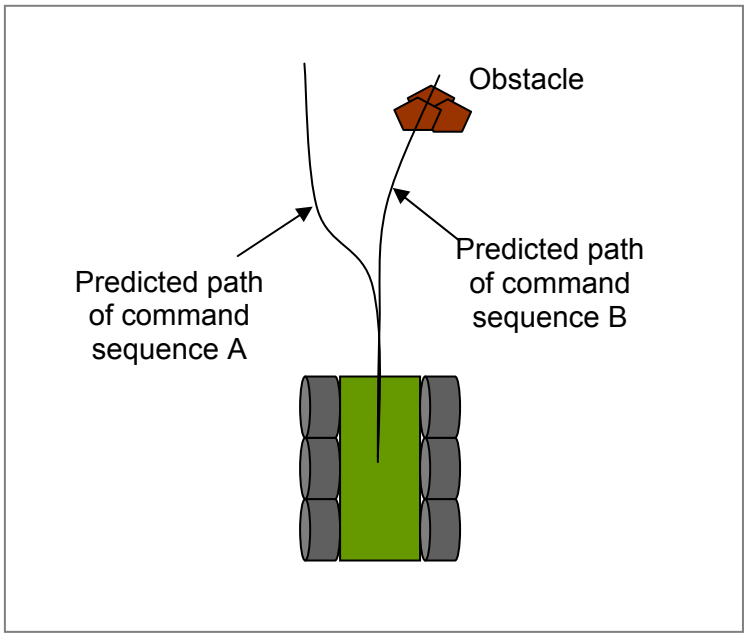

Figure 1: Planner Can Choose Command Sequence A to Avoid Obstacle

2 Experimental Platform

The Crusher platform – a six-ton, six-wheeled Unmanned Ground Vehicle (UGV) – was developed under the DARPA UGCV PerceptOR Integrated (UPI) Project. The UPI project is a Future Combat Systems (FCS) technology feed program with the objective of navigating complex terrain at a high rates of speed with minimal human supervision. The Crusher platform is shown in Figure 2.

Figure 2: The Crusher UGV Platform

Crusher's mobility system consists of a diesel-electric hybrid power system that is used to drive six independent motors located in each of the wheel hubs. Vehicle localization is run-time configurable and can use RTK or WAAS GPS with an optional odometry input. Inertial measurements are made by a three-axis Inertial Measurement Unit (IMU). Additional platform specifications are listed in Table 1.

Table 1: Additional Crusher Platform Specifications

Weight:	6 tons
Drive:	6 wheel independent drive
Horsepower:	282 hp per drive motor
Maximum Speed:	12.8 m/s
Suspension:	6 wheel independent suspension
Tire Diameter:	48 inches
Maximum Step Climb:	54+ inches

3 Conventional Model

3.1 Conventional Model Prediction Functions

The conventional forward predictive model used on the Crusher vehicle imposes a steering mapping over the left and right track commands so that it can operate via translational and rotational velocity controls. In the traditional model, these components are treated as two separate control systems. The model measures a control error as the difference between the current velocities and the commanded velocities. It then applies an empirically tuned gain to the error signal and predicts the final velocity at the end of the simulation period.

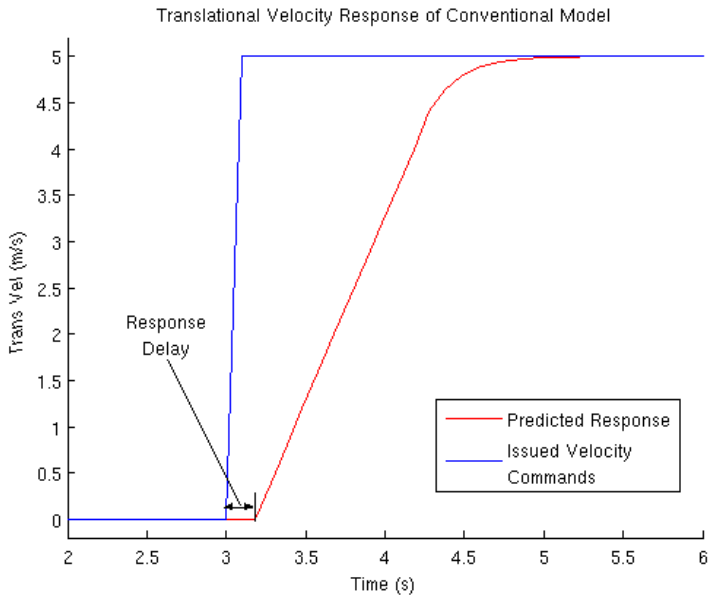

Figure 3: Conventional Model Translational Velocity Prediction

The steering mapping considers the vehicle to be a two-wheeled differential drive vehicle with no wheel slip. This is an extreme simplification. Since the vehicle has tracks of three wheels on each side of the body, when executing a turn forces generated from the terrain will act in opposition of one another. This causes the output vehicle rotational velocity to be smaller than the rotational velocity requested of the steering mapping. The conventional model compensates for this through an empirically derived scale offset to model the vehicle rotational velocity. This scale offset is dependent on the translational velocity of the vehicle – the faster the vehicle is traveling the greater the rotational velocity loss while turning. Figure 4 shows the output of the rotational velocity model.

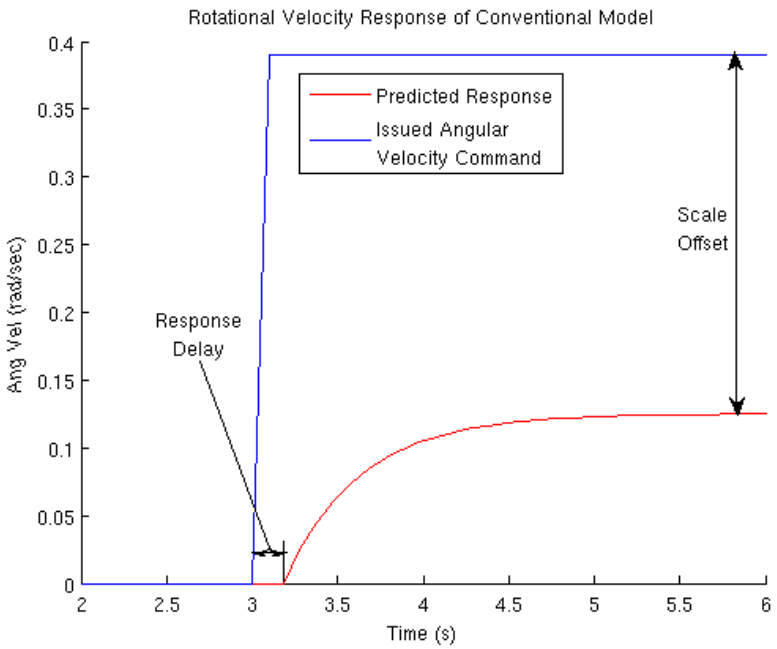

Figure 4: Conventional Model Rotational Velocity Prediction for a Translational Velocity of 5m/s

Since cross track velocity is not a degree of freedom in which the robot can articulate, the conventional model always predicts that the cross track velocity will be zero.

3.2 Limitations

Perhaps the most severe limitation of the conventional model is that it completely ignores terrain geometry. Variations in wheel slip are highly dependent on the slope and geometry of the underlying terrain. Given a sequence of commands, this creates a correlation between the vehicle motion and the terrain geometry. Figure 5 and Figure 6 illustrate this dependency. In each example, the vehicle was given an identical script of commands to execute while the starting configuration was changed to cause the vehicle to traverse varying terrain geometries. The scripts in both examples consisted of five seconds of driving commands. In Figure 5, the vehicle issued varying translational velocity commands while the rotation velocity command was held to be zero. In both cases of cross-slope terrain geometries (positive and negative roll) the vehicle slides several meters in the downhill direction – denoted as a "+Y" displacement for a positive roll and a "–Y" displacement for a negative one. An additional point to note is that the trajectory length for the uphill terrain geometry is several meters shorter than the rest. This is due to an increased amount of longitudinal wheel slip when the vehicle is driving uphill.

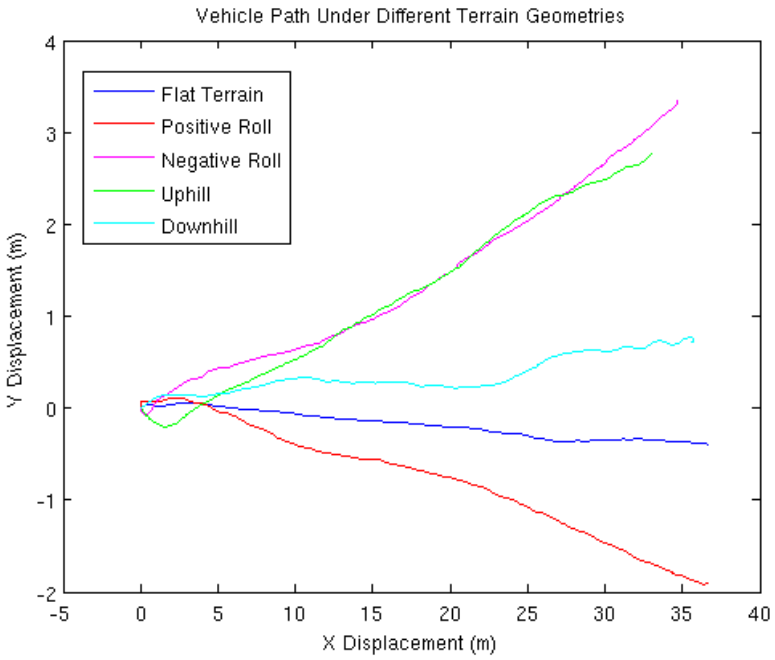

Figure 5: Translation Velocity Commands with Different Terrain Geometries

Figure 6 illustrates these effects when non-zero rotational velocities are commanded. The paths vary significantly not only in vehicle displacement, but also in predicted vehicle heading. The difference in terminal headings over the five seconds of commands is 103 degrees for the cases of initial positive and initial negative rolls.

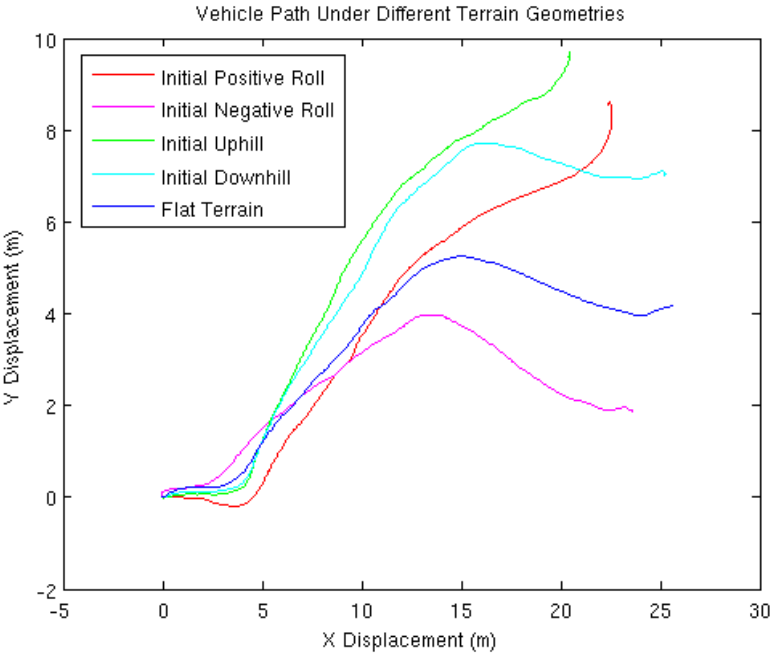

Figure 6: Translation and Rotational Velocity Commands with Different Terrain Geometries

In addition to ignoring terrain geometries, the conventional model assumes zero cross track velocity. This is clearly a poor prediction when the vehicle is sliding down a steep cross-slope as illustrated in Figure 5. Additionally, executing a turn at a high speed of 5m/s often causes the vehicle to slide through the turn. Empirical evidence shows that the cross-track velocities during these events can reach as high as 4.2m/s.

In addition to geometry, terrain material properties play an important role in motion prediction. Loose terrain such as gravel and sand cause much larger longitudinal wheel slip than well-packed terrain. This wheel slip results in translational velocities much smaller than the ones the conventional model will predict.

In extreme cases of loose soil, the terrain is incapable of supplying enough force on the wheels to move the vehicle forward. Once shear soil failure occurs, the wheels proceed to dig up the terrain causing the vehicle to sink deeper and deeper. Since the conventional model has no knowledge of the terrain - let alone the temporal changes to the terrain as the vehicle is traversing it - it is beyond the scope of the conventional model to predict and adapt to these events.

3.3 Physics-based Models

The rigid body dynamics describing the planner motion of a skid-steer vehicle are summarized by the following equations [4]:

$$v_{AT}' = \Sigma_i Fx_i(\xi_h, \xi_s) / m_s + v_{XT} \cdot \omega$$
$$v_{XT}' = \Sigma_i Fy_i(\xi_h, \xi_s) / m_s + v_{AT} \cdot \omega$$
$$\omega' = q(Fx_i(\xi_h, \xi_s), Fy_i(\xi_h, \xi_s), M_r)) / J_s$$
$$\psi' = \omega$$

The above sums are computed over each wheel. In the equations above: v_{AT} is the along track velocity; v_{XT} is the cross track velocity; ω is the rate of heading change; q is the moments function; M_r is the resistive moment due to yawing; m_s is the sprung mass; and J_s is the sprung inertia of the vehicle. Unfortunately, the functions Fx and Fy cannot be evaluated explicitly because they rely on numerous quantities that are not known *a priori*. These include the size, shape, and location of the contact patch between each wheel, as well as the terrain soil cohesion and shear angle and amount of sinkage at each contact patch.

To make the equations of motion tractable, most models make several assumptions. Table 2 enumerates most of the assumptions that common physics-based models make and explains why they are violated by the Crusher platform. The numerous violations of these assumptions make Crusher an ideal platform for deriving a forward vehicle model from empirical evidence.

Table 2: Model Assumptions Violated by the Crusher Platform

Typical Assumption	Violation
Negligible amount of wheel slip/slide	Wheel sliding is required for a skid-steer vehicle to turn
Supporting terrain can withstand shear forces applied by wheels	Very few surfaces can withstand the shear forces generated by Crusher (including asphalt)
Constant terrain material properties	Off-road vehicle encounters varying terrain – rocks, sand, mud, etc.
Negligible suspension travel	Significant and independent suspension travel
Wheel ground contact modeled as a point	Low tire pressure for traction results in a one square-foot terrain contact patch

4 Learning Approach

4.1 Design Guidelines

When setting out to improve the predictive modeling, a decision must be made as to which modeling deficiencies to address. This decision is based on not only which deficiencies produce the greatest modeling error, but also which ones have easily measurable features that can be used to make improvements in the model predictions.

As illustrated in Section 3.2 the effects of terrain geometry and cross-track velocity components that the vehicle generates in practice are substantial. These effects of are observable in several of the features recorded by the Crusher autonomy system. The roll and pitch reported by the onboard IMU give a reliable estimate of the geometry of the terrain in the vehicle footprint. The online perception system constructs a high fidelity elevation map of the terrain supporting surface by using techniques to subtract vegetation and to smooth the terrain to unobserved areas. Over-flight prior data has also been collected at all of the Crusher testing sites. This vantage point provides an optimal perspective of the terrain geometry. Additionally the positioning system reports highly accurate component velocities, which provide a readily available ground truth.

The field of terra-mechanics identifies five primary material properties (soil cohesion, internal friction angle, sinkage coefficient, shear deformation modulus and the maximum shear before soil failure) which influence the forces generated by the terrain and in turn the resulting vehicle motion. Observations of these properties are difficult to make extroceptively. Generally, approaches have been applied using vision techniques to classify terrain patches into a handful of categories (gravel, sand, grass, concrete, etc.) [5]. The current onboard perception system makes no such classifications; even if it did, it is unclear if a handful of terrain classifications are sufficient to improve the predictions that are made. An effort could be made to proprioceptivly learn the terrain material properties in the vehicle footprint and then correlate the visual appearance of the footprint with local visual features to make terrain property predictions

extroceptivly. This, however, is beyond the scope of this preliminary research. For these reasons the initial version of the learned forward predictive model is naïve to the terrain material properties.

Similar to the terrain properties, the temporal changes in terrain are difficult to quantify. The Crusher autonomy system has placed a control system around the event of complete terrain locomotion failure. When the autonomy system detects that the vehicle is not making progress in the intended direction, the planning system is given a signal to construct a new plan which avoids the trouble area. This system has worked well in practice and therefore improving modeling during these events is not essential. For this reason the learning system does not actively model how the terrain properties change over time.

In addition to deciding what problems the learned model will remedy, there are several other requirements that the learned vehicle model must address. These include efficiency as well as safety requirements.

The planning subsystem is accustomed to evaluating several thousand command sequences per second. To be suitable for use in the existing infrastructure, the learned model must be computationally efficient so that the planning subsystem would be able to evaluate a similar number of command sequences per second using it.

The Crusher vehicles have logged over one thousand kilometers of autonomous operation. This provides an immense repository of data that can be used for training a learned model. While using the entire data repository for training is not essential, the selected learning approach should nevertheless be adept at handling large quantities of data.

For safety considerations, offline learning is preferable to online learning because the quality of the model can be evaluated prior to its use on a large and potentially dangerous machine. A particular failure mode of concern would be an online learning algorithm learning that executing very high velocity commands while the vehicle is stuck in mud or gravel produces modest vehicle velocity. If the planning system were operating under this assumption when the vehicle is not stuck, the model would severely under predict the net motion of the vehicle making collisions very likely.

4.2 Feature Selection

With the purpose of learning the mapping between commands and the produced vehicle trajectory given the current vehicle state, enough information must be available in the vehicle state to make accurate predictions. Selecting the features that state will be comprised of relies on both physical principles and empirical evidence.

The component velocity measurements, produced at a rate of 100Hz, provide useful information with regard vehicle momentum. If the vehicle was commanded to go 5m/s, the resultant vehicle velocity in the near future would be significantly different if the vehicle was currently stopped compared to if it was

already going 5m/s. This dependency indicates that the current vehicle along-track, cross-track, and rotational velocities have strong predictive power and should be included as features.

Since the main deficiency that the learned forward predictive model is trying to redress is the effect of terrain geometries, features must be included that are indicative of the terrain geometry. Vehicle attitude (roll and pitch) provides an estimate of the slope of the terrain under the vehicle footprint. These features are reported from the positioning system at a rate of 100Hz. Since the lateral weight distribution of the vehicle is approximately symmetrical, from a dynamics standpoint the vehicle should respond in a near symmetrical manner if it is rolled to the right or rolled to the left. Due to this symmetry the square of the reported roll is also a useful feature. The longitudinal weight distribution is not symmetrical with a majority of the vehicle weight in the rear half of the vehicle. Therefore, the square of the pitch value is less useful in making predictions. Because of the high accuracy and low latency of these measurements as well as the information they reveal about the terrain geometry, the vehicle pitch, roll, and roll squared are included as features.

The vehicle also produced several other measurements of vehicle state that provide evidence of the underlying terrain geometry. These include the angles of the suspension arms that each of the six wheels are attached to and the current amount of pressure on each of these suspension arms. While this information can be useful in reconstruction of the terrain geometry, under normal operating conditions these features add little information. Only when a wheel is traversing a large discrete step do these features reveal more terrain geometry information than the vehicle roll and pitch by themselves. In addition to needlessly increasing the dimensionality of the feature space, these measurements are reported by the vehicle at a rate of only 10Hz. Furthermore, the suspension system operates by compressing a hydraulic-gas which causes the reported pressures to vary as a function of temperature. Because of the questionable accuracy and high latency of these features as well as their diminished predictive power, the suspension arm angles and pressures are disregarded.

Finally, in order to predict the vehicle motion generated from a command sequence, the sequence in question must be included in the feature set. The motor controllers can accept commands at a rate no higher than 10Hz. For this reason input commands are included as features at a rate of 10Hz for the duration of the prediction timescale.

4.3 Choice of Learner

The velocities generated by vehicle-terrain interaction are nonlinear with respect to several of the above features. Therefore there is a preference for a non-linear learning model. This preference combined with the large amount of training data available and the desire for offline training makes a neural network a logical choice of learner. For the desired feature set and predictions, the general structure of the neural network used is shown in Figure 7.

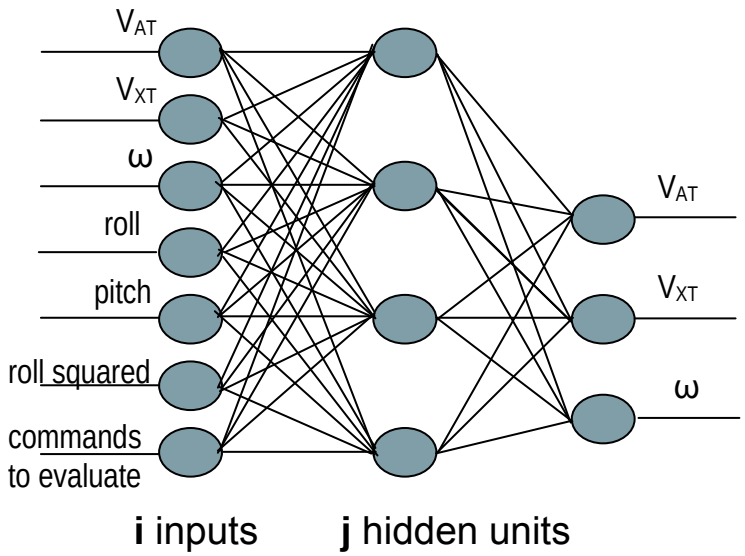

Figure 7: General Structure of Neural Network Model

4.4 Timescale Selection

The neural network is trained to make predictions of the vehicle's velocities over a fixed timescale. To predict motion over a time larger than the training timescale, the outputs of forward propagation can be integrated to estimate the vehicle motion over one time-step. With the output from the first forward propagation, the network can be evaluated again to predict motion over the second time-step. Chaining together predictions in this way requires that all inputs must be predicted in the future. While the network only predicts the future velocity components, the roll and pitch of the vehicle can be re-estimated after each integration step using a ground plane estimate and an attachment model. Likewise, since the commands are a time varying input to the model, future command inputs are obtained by advancing time in the command sequence. Note that for general input features, future input predictions cannot be obtained without explicitly making the feature an output of the network. The network predictions can be chained together until the desired prediction time is reached as shown in Figure 8.

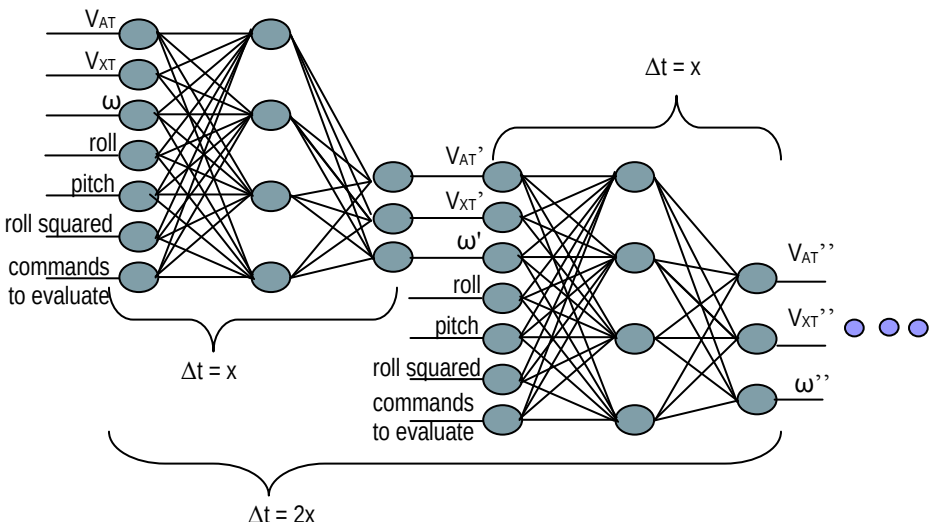

Figure 8: Chaining Together Multiple Predictions

An important consideration in training the network is the timescale over which to make predictions. If the timescale is chosen too small, the data will not reveal the effects of the issued commands. In this case, the network will learn to predict that the commands have no effect on the motion of the vehicle and the output velocities will be predicted to match the input velocities. If the timescale is chosen too large then the input features will change in an unpredictable manner over the time scale and become irrelevant. For example, the vehicle roll and pitch will remain relatively constant over a small interval of time but will vary widely over a large one. Therefore, if a large time scale is chosen the predictive strength of such features is greatly diminished. Additionally, the network is approximating the continuous functions of velocity by a set of values at discrete times. An assumption of constant acceleration is used to determine the vehicle position at times in the intervals between these time points. As the timescale grows larger, the spacing between these points increases and the constant acceleration model produces larger prediction errors.

4.5 Network Structure

Cross-validation was used extensively to choose the network structure. First the prediction timescale and the number of hidden units were determined via cross-validation. The prediction timescale was selected from a set of candidate durations. This set included: 0.1, 0.15, 0.2, 0.25, 0.3, 0.5, 1.0 and 1.5 seconds. The number of hidden units was allowed to vary from 5 to 15. Cross validation compared the ground truth velocity measurements to the predictions made using each timescale over trajectories that were three seconds long. This procedure found that the most accurate predictions were produced when using a timescale of 0.25 seconds and 8 hidden units.

5 Results

5.1 Quantifying Model Performance

Data for the results in this section was collected from the green Crusher vehicle operating under autonomous control. The data was collected on fairly homogeneous and sloped terrain but it did include several sections of deep mud and soft soil. The data was collected in August 2007 at the Ft. Carson military base in Colorado Springs, CO. The planning system nominally performs obstacle avoidance out three seconds into the future. For this reason all of the data was divided into trajectory segments of three seconds in length. Unless mentioned otherwise, the data set was comprised of nearly 7 hours of autonomous operation and consisted of 7350 random training trajectories and 817 held-out test trajectories.

To quantify the performance of a given predictive model, the following technique was used:

1. Sample the ground truth trajectory at 100Hz.
2. Given an initial starting state and a command sequence, use the model construct the predicted trajectory as a function of time.
3. Sample this trajectory at a rate of 100Hz. To do this, the predictive model was used to find the component velocities at each of the time points and a constant acceleration model was used to generate samples between the time points.
4. For each point of the sampled predicted trajectory, measure the x-y distance to the corresponding point on the ground truth trajectory. Average all of these errors to compute the Mean RMS error of the model for a given trajectory.

This technique approximates the time dependent integral of the difference between the predicted and actual trajectories. Figure 9 illustrates how this integral is measured. In the figure, the blue trajectory is the ground truth trajectory of the vehicle sampled at 100Hz. The red trajectory is a predicted trajectory to be compared to the ground truth. The green lines measure the distance between the points at the same corresponding time on each trajectory.

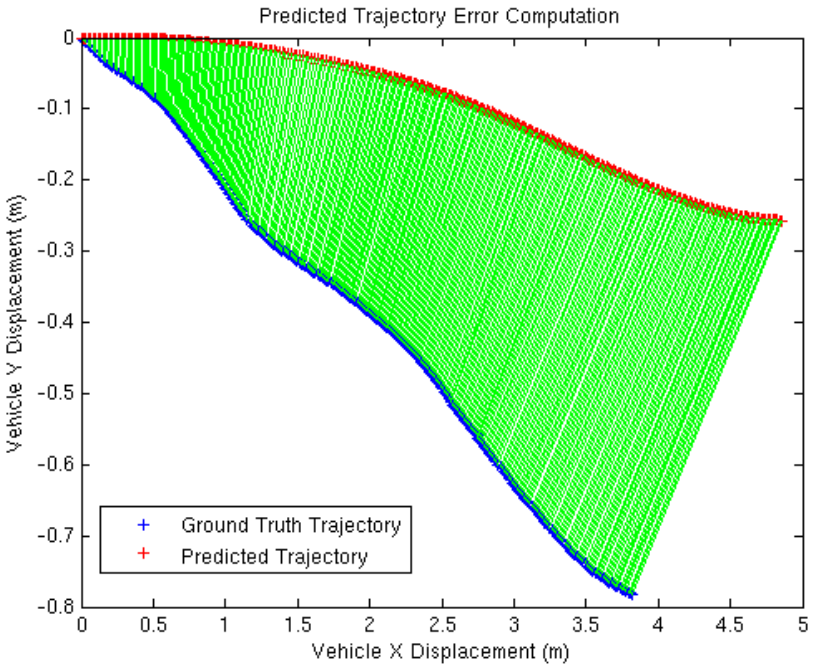

Figure 9: Measuring the Error of a Predicted Trajectory

Using this technique to measure error, it is first useful to compare the magnitude of the prediction error using a learned model to the magnitude of the error using the conventional model. On the 817 held out trajectories, the mean RMS errors of both models are show in Table 3. The learned model error is nearly a 60% percent reduction in the error of the conventional model error.

Table 3: Prediction Error of Conventional and Learned Predictive Models on Test Set

	Mean RMS Displacement Error
Conventional Model	1.2708 m
Leaned Model	0.5268 m

Analysis of the individual test trajectories reveals that the trajectories where the learned model makes the most significant improvement are those where the vehicle is attempting to turn while driving up a steep hill. To understand why this is, it is useful to evaluate what physically happens when the vehicle is executing a turn. When the vehicle is commanded to turn, one track of wheels is issued a high velocity command while the other track is sent a lower velocity command. This difference in track speeds is what creates a turn. As the desired curvature increases, this difference will cause the command to the slower track of wheels to become smaller. As this command approaches zero, the fast track of

wheels is responsible for generating all of the propulsive force for the vehicle. While terrain is generally capable of supplying this force on flat, level terrain, this is generally not the case as the vehicle is attempting a hill climb. When driving up a steep slope, the force needed to propel the vehicle forward increases because of the additional component necessary to counteract gravity. A single track of wheels that tries to generate this force from the terrain tends to result in shear soil failure and very little propulsive force. Figure 10 shows a plot of the ground truth and predicted trajectories using both the learned model and the conventional one for such a test trajectory.

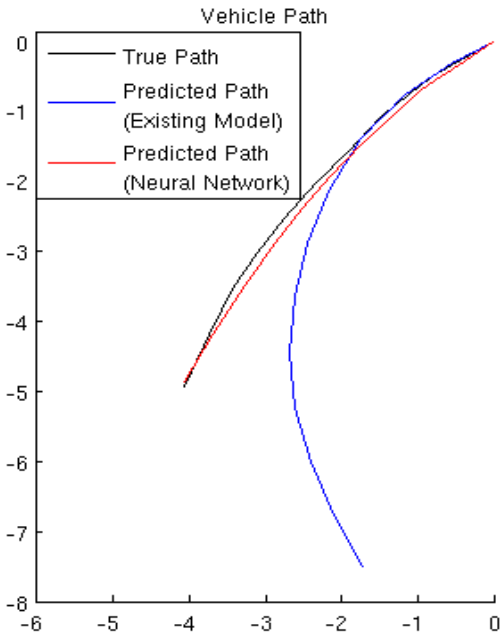

Figure 10: Conventional vs. Learned Model Predictions for Turning while Driving up a Slope of 22 Degrees

The planning system on the Crusher vehicle generally evaluates a set of potential curvature commands to decide which one to execute. It is therefore interesting to observe what these candidate curvatures look like when evaluated from varying starting states. Figure 11 and Figure 12 show the comparisons between the conventional model (the left pane) and the learned model (the right pane).

Figure 11 is derived from the vehicle starting on a 20-degree cross slope. In this instance, the bottom of the page represents downhill and the top represents uphill. Notice that the model predicts significant progress if the vehicle chooses to go straight or turn down the hill (traveling about 12 meters), but predicts that the vehicle will struggle if attempting to turn up the hill (only traveling about 5 meters for the sharpest turn). These predictions are in agreement with the physical phenomenon that is routinely observed when the vehicle tries to turn up a hill. Figure 12 shows the same set of candidate curvatures but this time the vehicle is placed heading straight up a 30-degree slope. In this example, the model predicts that the best course of action for making progress is to drive

straight. Again, this agrees with intuition and the terrain mechanics that are known to occur when turning on a steep hill.

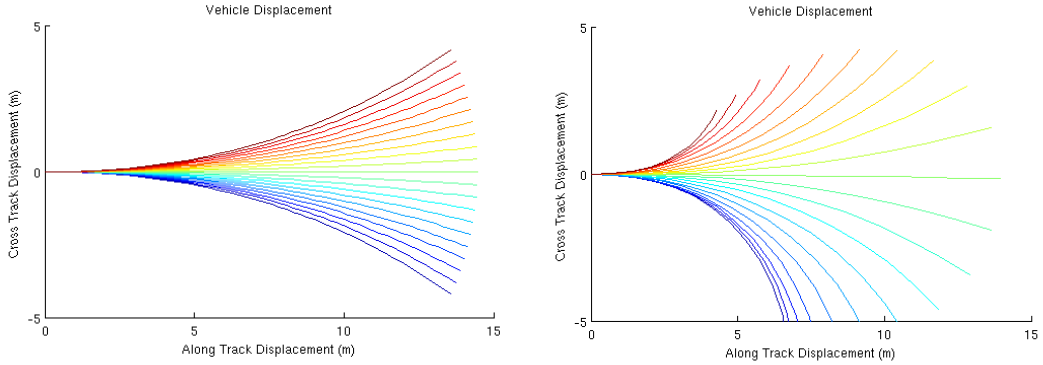

Figure 11: Predicted Candidate Trajectories of Conventional Model (left) and Learned Model (right) for Vehicle on a 25 Degree Cross-slope

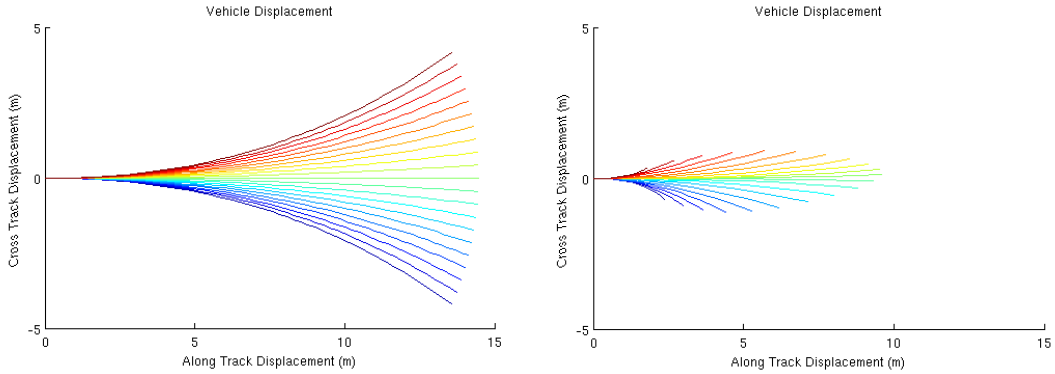

Figure 12: Predicted Candidate Trajectories of Conventional Model (left) and Learned Model (right) for Vehicle on a 30 Degree Up-slope

While predictive performance improvement of the learning approach over the conventional model shows that it is a step in the right direction, it is prudent to quantify the effects of the various design decisions made along the way. The following sections are devoted to this analysis.

5.2 Effect of Timescale

As stated in Section 4.4, choosing the optimal timescale over which to make predictions is critical to the accuracy of the learned model. Figure 13 shows the mean RMS error of several learned models each trained to make predictions over a different timescale. As expected the models predicting over a very short

timescale tend to perform poorly. This stems from three causes:

1. Over a small timescale, the learner has a more difficult time correlating issued commands to resulting actions
2. System noise becomes a dominating effect at small timescale. For example, the average control delay in the system is 0.18 seconds, but there is an average of +/-0.05 second noise on this latency.
3. Predicting over a smaller timescale results in more predictions being chained together to predict the entire trajectory. Errors accumulate at each of the timescale predictions.

Additionally as expected, when the timescale becomes too large error begins to increase. This tends to be the result of the following causes:

1. Over a large timescale, features such as roll and pitch are not as relevant because the vehicle's attitude is likely to change significantly over even a modest period of time.
2. To determine the vehicle's position at a time that is in between the timescale predictions, a linear acceleration model is used. This model will tend to be less accurate over a large timescale.

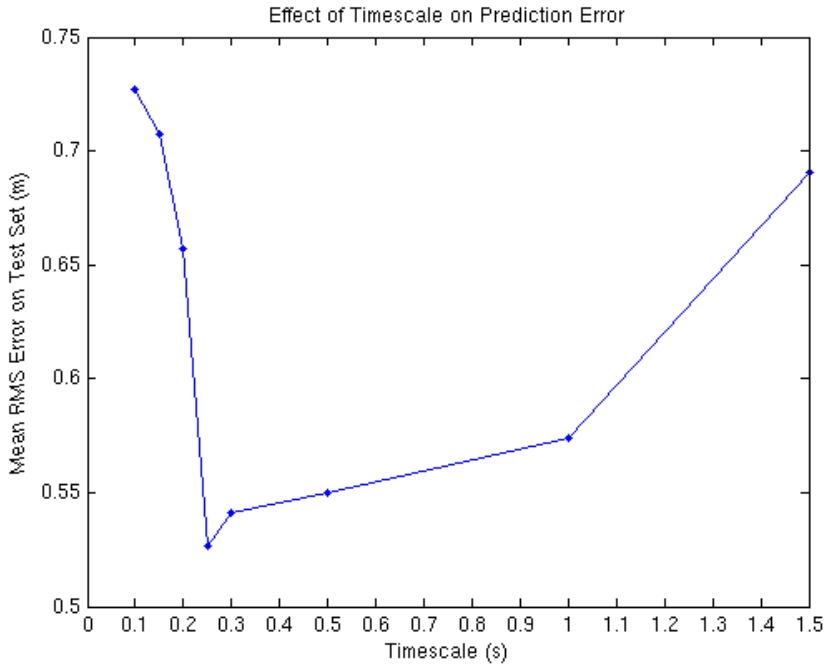

Figure 13: Effect of Timescale on Prediction Error

This experiment indicates that a timescale of about 0.25 seconds suffers minimally from the effects associated with both small and large timescales.

5.3 Effect of Amount of Training Data

While the seven hours of autonomous operation are small relative to the total amount of autonomy data collected with the Crusher vehicle, it would be ideal if the learned model could quickly be trained on data when the vehicle is operating in a new geographic area. For the next experiment, the learned model was trained with a smaller percentage of the training data to see how the reduction in training data quantity affects the quality of predictions. Figure 14 shows a plot of the model prediction error as a function of the percentage of original training data. This plot reveals that the model performs admirably even with only 10% of the original training data. This corresponds to about 40 minutes of time for data collection prior to training.

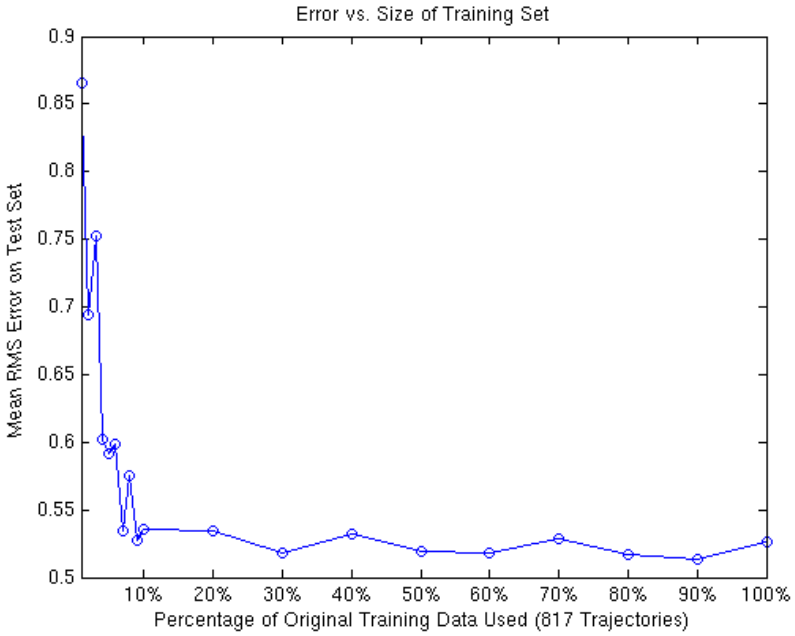

Figure 14: Effect of Training Set Size on Prediction Error

5.4 Effect of Non-linear Model

One of the reasons for choosing the neural network as the learning algorithm was its ability to model non-linear functions of the input features. From the relationship between the features and the underlying physics, this would seem to be a necessity. Unfortunately, choosing a non-linear model comes at the cost of the model making very poor predictions in areas of the feature space where training data is limited. A linear model, on the other hand, is more likely to make reasonable predictions in such areas of the feature space. Because of this side effect of the non-linear model, it should only be chosen if it significantly outperforms a learner using a linear model. To perform this analysis, a linear model was trained and tested on identical data. Performance results from the conventional model, a linear model and the non-linear neural network model are summarized in Table 4. To even the playing field a second linear model was

trained with the squares of all of the original features added as additional features.

Table 4: Linear vs. Non-linear Model Error

	Mean RMS Displacement Error
Conventional Model	1.2708 m
Linear Model	0.7820 m
Linear Model with Added Squares of Features	0.7312 m
Non-Linear Neural Network Model	0.5268 m

The better of the two linear models has a prediction error that is 42% less than the conventional model. The non-linear model however results in a 59% error reduction. If the online system generally operates in the realm of the feature space where training data is plentiful, the additional performance improvement through the non-linear model would justify the choice of using the neural network representation. Depending on the application there may be strong desire for the model to perform well in areas of the feature space where there is little training data. If this is the case, a linear model might be a reasonable choice given its significant improvement over the conventional model.

6 Extensions

6.1 Deep Neural Network

The optimal performance from the method above was achieved when the neural network minimized the error over a 0.25 second timescale. This is shorter than the three second interval over which the planner performs obstacle avoidance. To produce a trajectory over the duration used by the planner, multiple predictions were chained together. As predictions are chained together, errors accumulate. Further performance gains may be possible if the learner considered not only the error over the 0.25 second interval, but also minimized error over the entire three second trajectory. One technique for doing this is to view the series of chained predictions as a deep neural network. For the 0.25 second timescale, this amounts to 12 predictions being chained together resulting in a 24 layer deep neural network. A naïve approach would be to train the network using the three second ground truth trajectories as training input. The implementation of this approach suffers from two major roadblocks. First, this model has 2376 adjustable parameters (198 for each chained prediction). To learn in such a high dimensional space, two things are required: a good initial starting point so that local minima are avoided and an extremely large amount of training data. While the network parameters learned for the 0.25 second predictions can provide the former, training with a large amount of data is unavoidable. A solution to this problem is weight-tying and is described in Section 6.2. Second, the planning

system cares little about predicting the terminal velocities at the end of the three second interval. What is much more important is the model's ability to predict the correct vehicle position at all points in time within the three second interval. A technique for emphasizing this during the training process is to inject error measurements into the back propagation at each chained step where ground truth measurements are known. This approach is described in Section 6.3.

6.2 Weight-Tying

Section 5.3 showed that the chained shallow network model was capable of generating highly accurate predictions with about 40 minutes of training data. The straightforward implementation of a deep neural network would have 2376 weights that must be learned during training and would likely take several orders of magnitude additional training data to achieve similar results. This negatively affects not only the amount of time required to collect the training data, but also the time required to train the neural network. Since each single time-step prediction is modeling the same physical system, there is no strong reason for allowing the weights from one sub-network to vary from the other sub-networks in the deep network. By tying the weights together in this manner, the number of weights to learn is reduced to 198 which is the same number of weights learned in the shallow model approach.

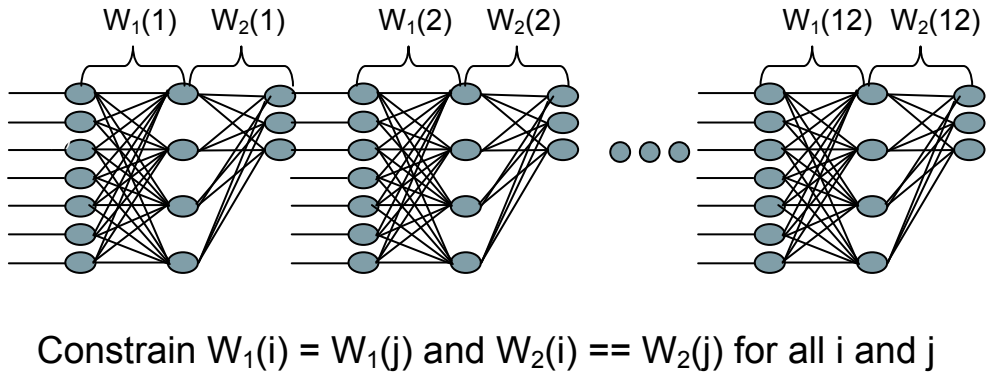

Figure 15: Weight Tying in the Deep Network

Tying the weights together in this way necessitates a slight change to the standard back propagation training. During the back propagation, the computed error gradients for a given weight are summed over each of the chained networks. The weight is then updated by taking a step along this summed gradient.

6.3 Error Measurement Injection

Standard back propagation on the deep network would operate by measuring the output error of the network and then computing the gradient errors of the output with respect to the hidden units. When transitioning to the previous sub-network, the gradient errors of the output with respect to the input are propagated backwards. This creates a flow of error information from the downstream sub-networks to the upstream ones. To influence the training so that it will minimize

not only the error measured at the end of the three second interval, but also the error measured at each of the 11 other timescale prediction points in the trajectory, the measured error at each of the timescale predictions has been injected into the standard back propagation. Figure 16 illustrates the error flow of this back propagation technique from sub-network 12 to sub-network 11. The injected error term is measured at the end of sub-network 11 and is highlighted in the figure.

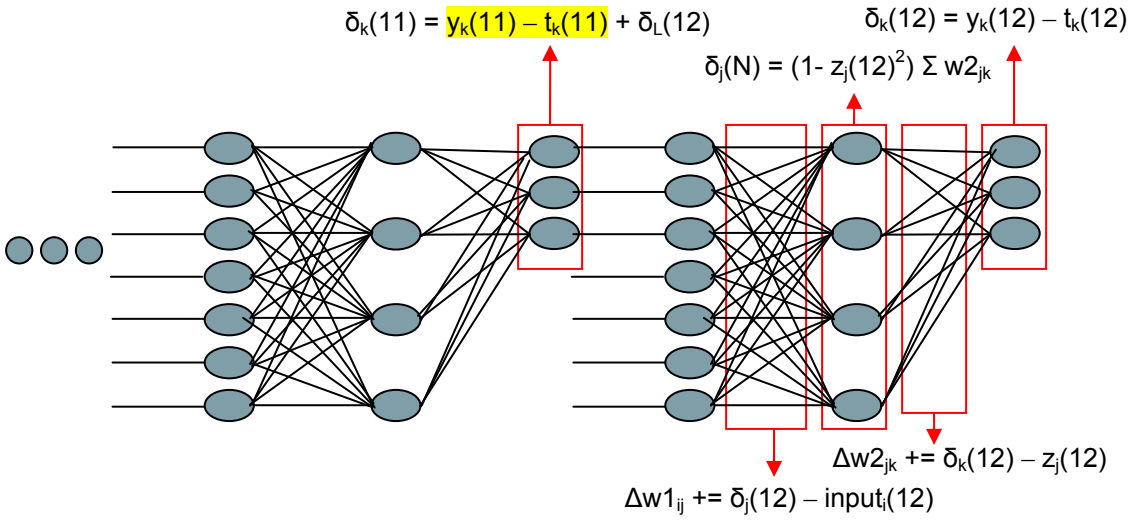

Figure 16: Error Injection into Back Propagation

6.4 Results

For the results presented below, the deep neural network was given the exact same training and test sets that the shallow model used in Section 5. Training a deep network in the manner described above results in a modest performance improvement for the 0.25 second timescale. Table 5 shows the mean RMS error of each of these models on the test data set. The results indicate that on average the deep network improved the prediction accuracy by just over one centimeter.

Table 5: Error Comparison of the Shallow and Deep Network Models for a 0.25 Second Timescale

	Mean RMS Displacement Error
Conventional Model	1.2708 m
Shallow Neural Network Model	0.5268 m
Deep Neural Network Model	0.5152 m

In theory, the deep network should help reduce the error that is accumulated as

the individual networks are chained together. This should manifest as a significant performance improvement over small timescales and a negligible improvement over very large ones. Figure 17 shows a plot of the shallow network model error and deep network model error as a function of timescale. As expected the deep network's predictions are significantly better than those of the shallow network for the timescales of 0.1 and 0.2 seconds. For timescales of 0.25 and 0.3 seconds the deep network results in modest performance improvements. For timescales 0.5 seconds and above, the performance deep network and the shallow network are equivalent. Figure 18 put this error in perspective by adding the comparison to the conventional model.

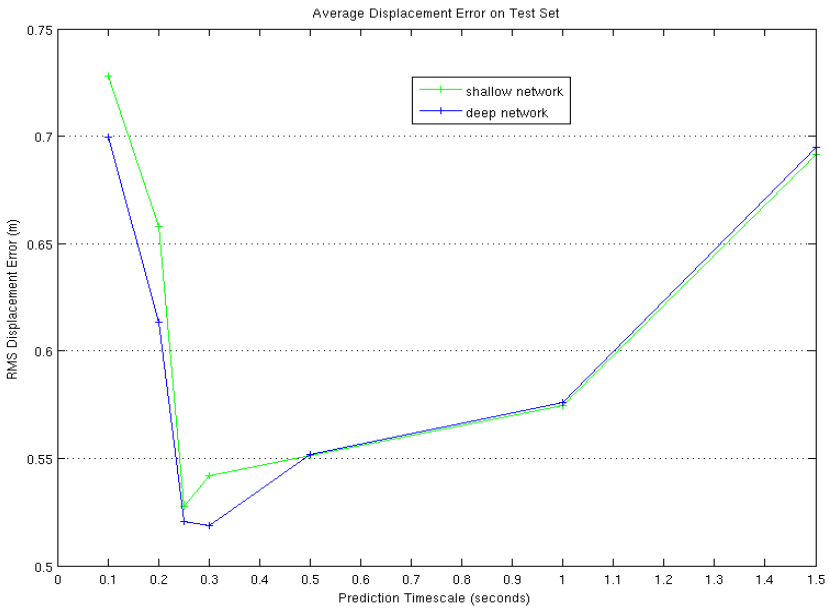

Figure 17: Shallow (green) and Deep (blue) Network Performance as a Function of Timescale

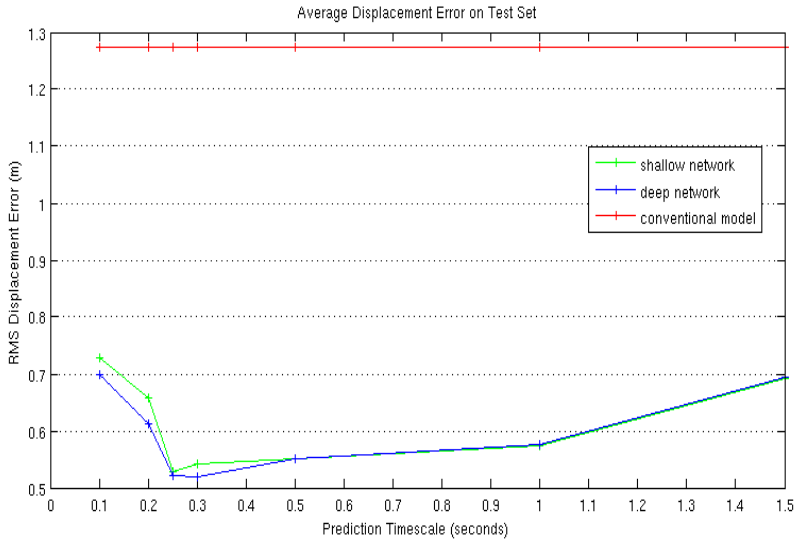

Figure 18: Shallow (green) and Deep (blue) Network Performance as a Function of Timescale with Conventional Model (red) show as Reference

Finally, Figure 19 shows the reconstruction of a typical test set trajectory using the conventional model, the shallow network learned model, and the deep network learned model. To illustrate an unequivocal performance improvement by the deep network a time scale of 0.10 seconds was chosen for this figure.

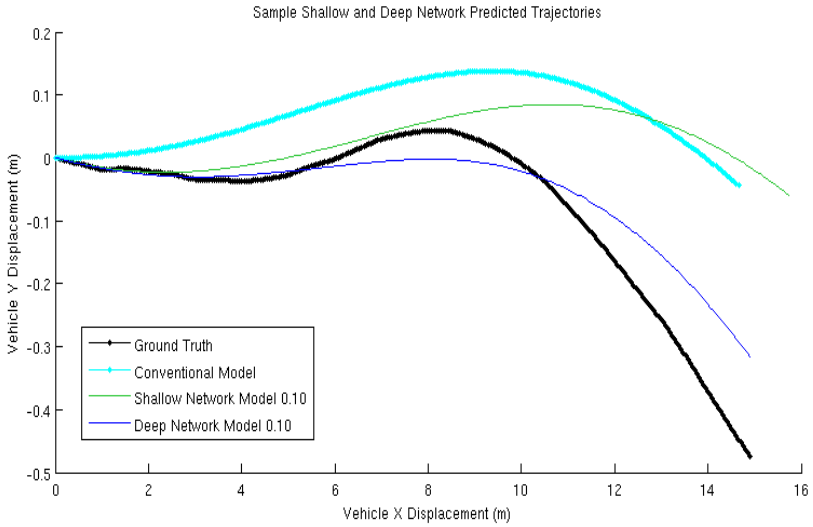

Figure 19: Three Second Trajectory Predictions Using a 0.1 Second Timescale

7 Conclusion

Taking into account terrain geometries in forward predictive modeling for off-road mobile robot operation has an unequivocal benefit on the accuracy of the prediction being made. Even a simple linear model that exploits geometric

information substantially out performs a model that is agnostic towards the underlying terrain shape. Data driven learning approaches can produce high quality predictions without delving into the complexities of the underlying terra-mechanics. A repeating neural network chain can be used to make predictions over an arbitrary long time interval. This allows for a total prediction interval that is tailored to the desired look-ahead of the obstacle avoidance system. This approach is computationally inexpensive to evaluate and can handle large amounts of training data. Moreover, it has resulted in a nearly 60% error reduction over the conventional model. If a significant amount of training data exists, further performance gains can be achieved by minimizing the error over the entire interval that the obstacle avoidance system evaluates. This has been implemented as a deep network with tied weights and supplemental error injection. If the underling network makes predictions over large timescales, these gains tend to be marginal, but if the underling timescales are small these gains can be significant.

References

[1] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner. Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments. *The International Journal of Robotics Research*, Vol 25, No 5/6, 2006

[2] C. Shoemaker & J. Bornstein. The Demo III UGV Program: A Testbed for Autonomous Navigation Research. *Proceedings of the IEEE International Conference on Intelligent Control*, 1998

[3] Wellington, C. & Stentz, T. Online Adaptive Rough-Terrain Navigation in Vegetation. *IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[4] J.T. Economou, R.E. Colyer, A. Tsourdos, B.A. White. Fuzzy logic approaches for wheeled skid-steer vehicles. *Proceedings of the IEEE Vehicular Technology Conference,* 2002.

[5] A. Angelova, L. Matthies, D.Helmick, G. Sibley, P. Perona. Learning to Predict Slip for Ground Robots. *Proceedings of the Robotics: Science and Systems Conference,* 2006.