Robust and Adaptive Rough Terrain Navigation Through Training in Varied Simulated Dynamics

Sean J. Wang[1], Si Heng Teng[2], and Aaron M. Johnson[1]

***Abstract*— We propose a model-based reinforcement learning approach for robust and adaptive long-horizon navigation in rough terrain environments. Offline, we train an adaptive dynamics model using a wide range of simulated systems. This model can adapt to any new system using state-transition observations from that system. Predictions from the model capture uncertainty about the system's exact dynamics stemming from insufficient observations. Online, we use a divergence constrained path planner to find routes that are robust to the robot's current understanding of dynamics. In our results, we show this allows for long-horizon driving strategies that are conservative when state-transition observations are limited but have improved performance after giving few state-transition observations.**

I. Introduction

Autonomous rough terrain navigation in off-road environments is challenging as it requires careful reasoning about how the robot should traverse different obstacles and uneven terrain. For this task, robust long-horizon decision making is crucial as failing to consider long term consequences of immediate actions may result in the robot getting trapped in a dead end. Furthermore, the robot's dynamics may be highly divergent due to the precarious nature of driving over uneven terrain. As such, robustness towards potential prediction uncertainty is essential to prevent ill-informed actions from leading to failure or catastrophic damage.

In addition to these challenges, off-road driving often involves a diverse range of terrains that each uniquely impact vehicle dynamics. The robot must be mindful of these different effects and properly accommodate its driving strategy to match. For instance, in environments with loose dirt, the vehicle's tires may not provide enough traction.

To begin to address these issues, in [1], the authors propose a method for robust adaptation to new real-world dynamics through the use of adaptive probabilistic dynamics models trained solely in simulation. By varying simulated dynamics during training, the adaptive dynamics model is trained to tailor predictions towards any particular system dynamics using state-transition observations previously collected under those dynamics. Furthermore, the predictions from the probabilistic model captures uncertainty stemming from insufficient observations leading to an ambiguous understanding of dynamics. The authors used the model within an uncertainty aware model predictive control framework, specifically Risk-Aware Model Predictive Path Integral [2]. The resulting controls framework allows the robot to drive conservatively and robustly upon initialization, when it has a poor understanding of dynamic, but also increase driving performance as it collects more observations to adapt the model and improve its understanding.

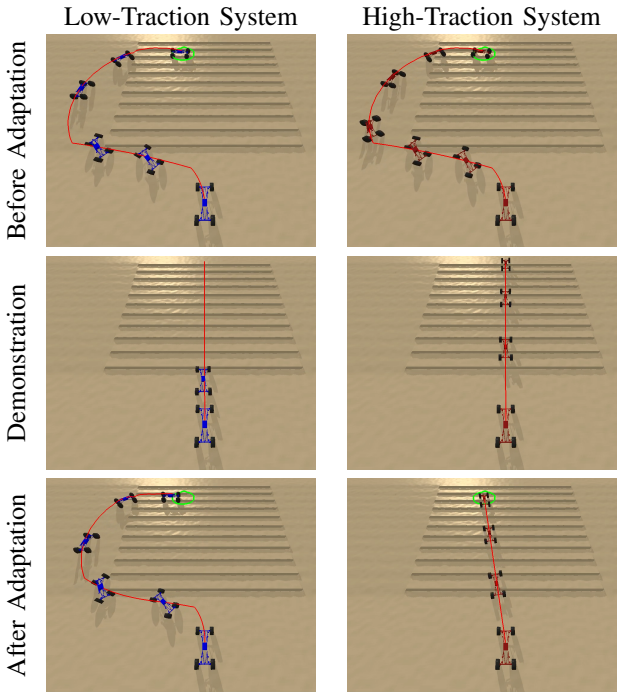

Fig. 1: A low-traction and high-traction robot navigating a stair obstacle. Before adaptation, both systems choose a safe route that avoids driving directly up the stairs (top). The model was adapted to each system using data collected through one demonstration (middle). After adaptation, the planner matches the route to the capabilities of each system, with the high-traction system's route going directly up the stairs (bottom). The reference trajectory chosen by the planner (or manually for demonstration) is shown in red. The goal is shown in green. Robot's actual path is overlaid.

Furthermore, in [3], the authors propose a model-based reinforcement learning framework for long horizon navigation in rough terrain. This framework assumes minimal changes to the vehicle's dynamics and collects a large driving dataset to train a model to predict the dynamics when driving over uneven terrain. Online, the framework uses a "divergence constraint" [4] to select a long-horizon plan through the terrain that is robust to modeling uncertainty.

In this workshop paper, we extend and combine these concepts to result in a long-horizon rough terrain navigation framework that can quickly and robustly adapt to new system dynamics. Similar to [1], our approach leverages a wide range of simulated systems, each with different dynamics,

[1] Department of Mechanical Engineering, [2] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.
{sjw2,sihengt,amj1}@andrew.cmu.edu

to train a dynamics model that can probabilistically adapt to new system given state-transition observations. However, we adopt a similar dynamics model architecture from [3] to allow the model to predict the driving dynamics when driving over uneven terrain. Similar to [3], we use the predictive model and the divergence constraint to find robust routes through rough terrain environments. We show in our experimental results that this approach enables effective rough terrain navigation that can robustly adapt to new dynamics. Furthermore, this approach balances robustness with adaptability, Fig. 1.

II. Background

The robust and adaptive driving approach in [1] follow the model-based reinforcement learning (MBRL) paradigm, where a dynamics model is first learned then used for controls. However, unlike most other MBRL frameworks, this approach trains the dynamics model solely in simulation and uses minimal real world data to adapt it to the true dynamics of the real world system. The dynamics model used consists of two modules: the System Identification Transformer and the Adaptive Dynamics Model.

System Identification Transformer (SIT): Given prior state transition observations (consisting of the robot state s_i, action a_i, and next state s_{i+1}) collected under the system's current dynamics, $\mathcal{H}_t = \{(s_i, a_i, s_{i+1}) | i < t-1\}$, the SIT, denoted by $\mathcal{T}_\theta$, extracts an understanding about the system's dynamics as a succinct latent context vector, $c_t = \mathcal{T}_\theta(\mathcal{H}_t)$.

Adaptive Dynamics Model (ADM): The ADM, denoted by $\mathcal{P}_\theta$, probabilistically models the system's stochastic state transition dynamics given the current context vector c_t from the SIT, which encodes an understanding of dynamics,

$$\hat{s}_{t+1} \sim \mathcal{P}_\theta(s_{t+1} | s_t, a_t, c_t), \tag{1}$$

where $\hat{s}_{t+1}$ is the predicted next state.

Model Training: Training of the SIT and ADM are done solely in simulation. During training, simulation parameters are randomized to generate many different systems with varying dynamics. While all the training systems differ from each other, they all share the same morphology as the target (real world) system. The SIT and ADM are trained jointly using a negative log-likelihood loss for predicted state-transition distributions. As a result of training on randomized systems, the SIT and ADM can work in tandem to extract an understanding of dynamics and probabilistically predict state-transitions for new unseen systems, including the real world system, given prior state-transition observations.

Robust Model Predictive Controls: Online, a Risk-Aware Model Predictive Path Integral is used to robustly control the robot under the current probabilistic understanding of vehicle dynamics captured by the SIT and ADM. By continuously collecting new state-transition observations and updating the context vector at each time step, the probabilistic understanding of dynamics can continually improve and adapt to the system's exact dynamics. In scenarios where prior state-transition observations only provide an ambiguous understanding of dynamics, e.g. upon initialization, the controls framework results in a more robust and conservative driving strategy. As knowledge of system dynamics improves with more observations, the control framework results in a higher performing driving strategy tailored towards the particular system's dynamics.

Divergence Constrained Rough Terrain Navigation: In [3], the authors collect training data offline to train a neural network to predict the robot's state transition when driving over uneven terrain, denoted as $\mathcal{P}_\theta(s_{t+1} | s_t, a_t)$. In this framework, the robot's state also includes a robot-centric terrain height map in s_t, which is cropped from the global map based on the robot's position and orientation. The neural network dynamics model uses a Convolutional Neural Network (CNN) to process the robot-centric height map for each prediction. The output of the neural network is a predicted state-transition distribution, from which a new robot state can be sampled, including a newly cropped robot-centric height map.

During decision making, the system chains predictions from the model to predict the robot's trajectory as a nominal trajectory (taking the mean prediction at each time step) and a particle distribution (propagating a set of particles sampling from the predicted distribution). For each prediction, the framework calculates a divergence metric defined as

$$u(\bar{d}, \hat{D}) = \max_t \frac{1}{I} \sum_{i=1}^{I} \| \hat{s}_t^i - \bar{s}_t \|_2^2, \tag{2}$$

where $\bar{s}_t$ is a predicted state on the nominal trajectory and $\hat{s}_t^i$ is a particle prediction. The framework performs trajectory optimization, constraining solutions to have low divergence to ensure that the robot can robustly track the nominal trajectory given the uncertainty in the learned model.

III. Method

For this work, following [1], we train a probabilistic dynamics model consisting of a SIT and ADM. Following [3], both models are conditioned on a terrain height map to account for the effects that uneven terrain has on dynamics. We then use a divergence constraint to choose trajectories that the robot can track robustly under model prediction uncertainty. In this framework, we opt to use the divergence constraint within an RRT planning framework [4] instead of within trajectory optimization [3]. An expanded explanation of the approach is included in [5, Chapter 5].

Terrain-Aware System Identification Transformer: The SIT we train extracts relevant information about the target system's dynamics given state-transition observations. We follow the notation from [3], where states s_t include the robot's position, velocity, and a robot-centric terrain height map, and actions a_t include steering and throttle commands. The architecture we use for the SIT includes a combination of CNNs [6] and Self Multi-head Attention Networks [7]. For each robot-centric observation, we first process the robot-centric heightmap using a shallow ResNet like architecture [8], where each layer consists of CNNs with a residual connection. The output of the ResNet is concatenated with

the rest of the robot-centric observations (robot state, action, and state-transition). All of the processed robot-centric observations are then fed into a self-attention network, resulting in a single context vector c_t. We opted for a self-attention network given the variable size of $\mathcal{H}_t$, which grows as more observations are collected.

Adaptive Dynamics Model: The ADM predicts the robot's state-transitions given the context c_t from the SIT network, the robot's current state, and reference trajectory. Like the SIT network, the robot state representation includes the robot's position, velocity, and a robot-centric height map which is fed into a CNN. In our implementation, the ADM is directly learning the closed-loop dynamics of the robot, i.e. predicting state-transitions based on the reference trajectory $\bar{d}_t$ chosen for the robot to track.

Model Training: We sampled a large set of simulated systems by randomizing physical parameters, such as tire contact friction. For each system, we collect 256 time steps of data, resetting the robot whenever it reaches the end of the reference trajectory or upon failure.

During model training, we iteratively sample a system variant from the collected data and trained the SIT and ADM networks end-to-end. From the 256 time steps of data, we choose a 32 continuous time step trajectory to predict with the Adaptive Dynamics Model. We provide the SIT with all the data collect prior to that randomly chosen 32 time step trajectory, thus varying the amount of data given to the SIT.

Divergence Constrained Planning: Online, we provide the SIT with all available state-transition observations to tailor ADM predictions to the given system. We use the stochastic ADM predictions within a robust planning framework to find paths through the obstacle field that are robust towards uncertainty in the current dynamics understanding.

We make the planner robust by using the divergence constraint from [3] to ensure that solution trajectories can be robustly tracked under modeling uncertainty. Our framework consists of an RRT search followed by trajectory pruning to improve optimality of the solution (Algorithms 1 and 2, in the Appendix), similar to [9–11]. During RRT tree expansion, candidate trajectories are only chosen if model predictions satisfy the divergence constraint. During the pruning phase, we randomly shorten the solution trajectory by interpolating between states and replace the solution with the shortened path if the divergence constraint is still satisfied.

IV. Results

We evaluated the navigation framework's ability to 1) choose robust strategies when unsure about the target system's capabilities, and 2) adapt its strategy to the target system's capabilities given system observations. We chose two systems for evaluation. One system had low traction and could only drive up gentle slopes. The other system had high traction and could drive up steep slopes and even some stairs. Here, we qualitatively analyze the framework's decision-making process and strategy adaptation.

For each test, we generated feasible trajectories to the goal using the divergence constrained RRT and trajectory pruning, then tracked the trajectories using the robot's low-level trajectory tracker. For each system, we ran two trials. One trial before any adaptation data was collected on the particular system then one trial after giving the SIT data to adapt the model to the specific system. Adaptation data was collected through one short trial of the robot tracking a manually defined trajectory.

The initial trial is similar to Domain Randomization [12], where the robot's policy is trained to be robust across a wide spectrum of systems. Before the model is adapted to any particular system, its probabilistic predictions capture possible outcomes for a wide spectrum of systems. The divergence constrained RRT planner then finds a strategy that will be effective across this range of dynamics. On the other hand, after adaptation, the model and resulting driving strategy is tailored to the particular target system.

Driving Up Stairs: In the first test, the robots were tasked with driving to the top of a set of stairs surrounded on both sides by ramps. Both systems were capable of driving up the ramp, but only the high-traction system could drive directly up the stairs. While driving up the ramp is a safer route, driving directly up the stairs is a more direct and lower cost route. The results of the trials for both systems before and after adaptation are shown in Fig. 1.

Before adaptation, the path planner always chooses the more conservative path of driving up the ramp. For both systems, we collected adaptation data by having the robots track a trajectory going straight up the stairs. Notice that the high-traction system was able to climb the stairs and track the trajectory faithfully, whereas the low-traction system failed and got stuck at the bottom of the stairs. After adapting the dynamics model to the low friction system, the strategy chosen by the planner did not change much and still avoided going straight up the stairs. However, after adapting to the high friction system, the planner chose the more aggressive strategy of driving directly up the stairs, showing a gained understanding of the robot's capabilities. Notice that the conservative strategy of driving up the ramp still required the robot to drive across the stairs. This means that the planner must not simply avoid stairs. The model must understand that ascending stairs may be problematic, while descending or driving across them is feasible.

Slope Driving Given Stair Demonstration: As another simple example, we tested the approach on a similar environment. However, in this scenario, the low-traction system was unable to drive up the ramps. The robot was placed on one side of the obstacle with the goal on the other. The direct route involved driving up and down the ramp sides of the obstacle, which only the high-traction system was capable of. The safer, but less direct, route involved driving around the obstacle to the goal. The results of the trials for both systems before and after adaptation are shown in Fig. 2.

Similar to the previous results, the path planner initially chooses the conservative path of driving around the obstacle for both systems. Again, we collected adaptation data by having the robots track a single trajectory going up the stairs which only the high-friction system was able to track. Again,

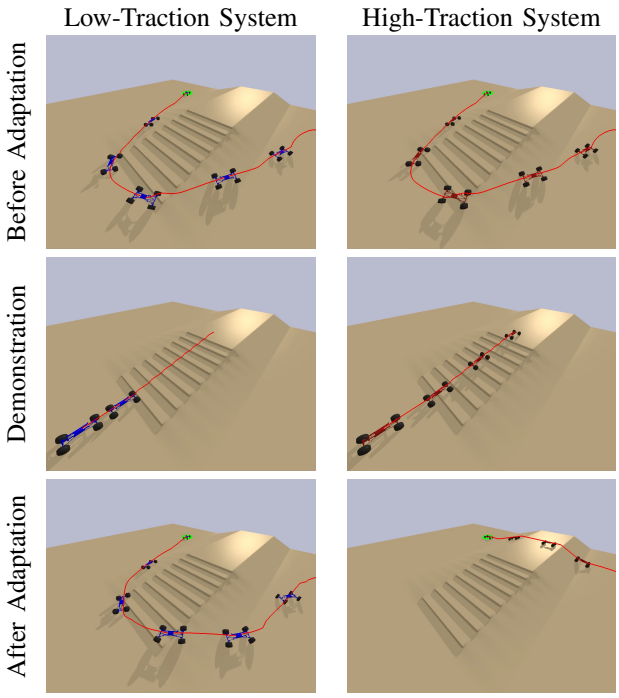

Fig. 2: A low-traction robot and high-traction robot navigating a ramp obstacle. Before adaptation, both systems choose a safe route that avoids driving directly up the ramp (top). The dynamics model was adapted to each system using data collected through one demonstration (middle). After adaptation, the planner matches the route to the capabilities of each system, with the high-traction system's route going directly up the ramp towards the goal (bottom). The reference trajectory chosen by the planner or manually chosen (for demonstration) is shown in red. The goal is shown in green. Robot's actual path is overlaid.

after demonstration, the planner changed only the plan for the high-traction system to be more aggressive and direct. However, in this scenario, the model took an observation of the two systems on stairs and correctly predicted each system's capabilities on a different steep ramp obstacle. This suggests that the SIT and ADM are not simply memorizing obstacles, but gaining a generalized understanding of the system's capabilities.

V. Discussion

While the results in this work are drawn from relatively simple toy environments, they highlight some key advantages of the adaptive rough terrain navigation framework developed in this work. Many of these key advantages stem from combining concepts from our previous works [1,3].

First, this framework is capable of non-myopic dynamics-aware decision making. In situations where the dynamics model has determined that the robot may be incapable of driving over the obstacle, the path planner chooses the path around the obstacle despite it being longer and less direct. The planner considers the robot's dynamics capabilities and does not simply avoid certain obstacles. This can be seen in the stair example. If the planner just avoided the stairs completely, the robot would not be able to make it to the goal. Instead, the planner distinguishes the differences in traction encountered when ascending the stairs as opposed to descending or driving across them.

This framework also allows for robust navigation given the current understanding of the target system's particular dynamics and capabilities. In the trials, the planner chose the more conservative route upon initialization when the target system observations are virtually non-existent and provide a vague understanding of the target system's particular capabilities. Furthermore, the understanding of the target system's capabilities is not static but improves as more observations are collected. This can be seen with the high-traction system. After collecting observations through one demonstration of driving up stairs, the planner changes its path to drive directly up the stairs or ramp, matching the robot's capabilities.

Furthermore, adaptations to the divergence constraint in this work make it relevant to other high-level planning frameworks and low-level planners than used in [3]. In this work, we show the divergence constraint used in an RRT-based decision making framework while using a different trajectory tracker (pure-pursuit). Although this tracker is still relatively simple, the modifications to the divergence constraint does open up many more possibilities.

Currently, the System Identification Transformer trained is only effective within the two environments shown in Fig. 1 and 2. Ideally, the SIT could effectively extract an understanding of the system's capabilities through observing the system interacting with any type of obstacles. However, this would require training it with a large dataset containing a diverse set of target system observations ($\mathcal{H}$) with many different combinations of obstacle interactions along with a diverse set of simulated dynamics. To simplify the problem, we used only a couple environments for training and testing to limit the number of obstacles the robot might encounter.

When we tried training the SIT by randomizing the terrain, following the approach in [3], it was unable to properly distinguish between different systems. This could be perhaps due to the sparsity of relevant information in the observation set $\mathcal{H}$ during training. Or perhaps this approach just required more data, a larger neural network architecture, and more engineering effort.

Another limitation of our current planning framework is its inability to vary velocity, thus not fully exploiting the vehicle dynamics potential, as could be achieved with the framework in [3]. In our current implementation, the planner used a fixed velocity to steer between different states. Future work could allow for velocity variations in the steer function thus enabling the planner to choose appropriate speeds.

Finally, a potential extension to the work is to incorporate safety constraints into the planner. Currently, the planner only constrains solutions to satisfy the divergence constraint, which mainly focuses on dynamic feasibility given prediction uncertainty. In the future, the planner can also incorporate other important considerations such as track boundary, vehicle rollover [13], or later acceleration constraints [1].

REFERENCES

[1] S. J. Wang, H. Zhu, and A. M. Johnson, "Pay attention to how you drive: Safe and adaptive model-based reinforcement learning for off-road driving," in *IEEE Intl. Conference on Robotics and Automation*, 2024.

[2] J. Yin, Z. Zhang, and P. Tsiotras, "Risk-aware model predictive path integral control using conditional value-at-risk," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 7937–7943.

[3] S. J. Wang, S. Triest, W. Wang, S. Scherer, and A. Johnson, "Rough terrain navigation using divergence constrained model-based reinforcement learning," in *Conference on Robot Learning*, 2021.

[4] A. M. Johnson, J. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016.

[5] S. J. Wang, "A practical approach to learning dynamics for rough terrain navigation," Ph.D. dissertation, Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA, February 2024.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[9] L. Petit and A. L. Desbiens, "RRT-Rope: A deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3d environments," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 1111–1118.

[10] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme, "Gradient-informed path smoothing for wheeled mobile robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1710–1717.

[11] B. Hu, Z. Cao, and M. Zhou, "An efficient RRT-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3292–3302, 2021.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30.

[13] T. Han, A. Liu, A. Li, A. Spitzer, G. Shi, and B. Boots, "Model predictive control for aggressive driving over uneven terrain," 2023.

APPENDIX

Algorithm 1: Divergence Constrained RRT Search

Input : Starting State: s_{start}
Goal Location: s_{goal}
Target System Observations: $\mathcal{H}$
Output: Solution Trajectory: $\bar{d}_{RRT}$

```
// Initialize tree root and initial
   prediction distribution at s_start
```
$z_{new} \leftarrow \text{NewNode}(s_{start}, \hat{D}_{last} = (s_{start}, ..., s_{start}))$
$T \leftarrow \text{InitializeTree}(z_{new})$
while $z_{new} \neq goal$ **do**
```
    // Generate reference trajectory
       to random state
```
    $s_{rand} \leftarrow \text{RandState}()$ or $goal$ with probability p_{goal}
    $z_{near} = (s_{near}, \hat{D}_{last}) \leftarrow \text{Nearest}(T, s_{rand})$
    $\bar{d} \leftarrow \text{Steer}(s_{near}, s_{rand}, T_{expand})$
```
    // Predict Trajectory Distribution
       & Divergence
```
    $\hat{D}, u \leftarrow \text{Predict}(\bar{d}, \hat{D}_{last}, \mathcal{H})$
```
    // Add new node to tree if
       divergence constraint satisfied
```
    if $u < U_{max}$ **then**
        $s_{new}, \hat{D}_{last} \leftarrow \text{LastTimeStep}(\bar{d}, \hat{D})$
        $z_{new} \leftarrow \text{NewNode}(\bar{s}_{new}, \hat{D}_{last})$
        $T \leftarrow \text{InsertNode}(T, z_{near}, z_{new})$

```
// Work backward through tree to
   find solution reference
   trajectory
```
$\bar{d}_{RRT} \leftarrow \text{Backward}(T, z_{new})$

Algorithm 2: Divergence Constrained Pruning

Input : Initial Solution: $\bar{d}_{RRT}$
Goal Location: $goal$
Target System Observations: $\mathcal{H}$
Output: Refined Solution: $\bar{d}_{prune}$

$\bar{d}_{prune} \leftarrow \bar{d}_{RRT}$
for *number of prune iterations* **do**
  `// Choose random segment to replace`
  $t_s, t_e \leftarrow$ RandomWindow$(\bar{d}_{prune})$
  $\bar{d}_{best} \leftarrow \bar{d}_{prune}[t_s : t_e]$
  `// Iterate through possible replacements to find best option`
  for $\bar{d}_{sub} \in$ CandidateSubTrajs$(\bar{d}_{prune}[t_s], \bar{d}_{prune}[t_e])$ **do**
    `// Predict divergence of resulting complete trajectory`
    $\bar{d}_{cand} \leftarrow$ Substitute$(\bar{d}_{prune}, t_s, t_e, \bar{d}_{sub})$
    $u \leftarrow$ EvalDivergence$(\bar{d}_{cand}, \mathcal{H})$
    `// Choose substitute if feasible and more optimal`
    if $u < U_{max}$ ***and*** Cost$(\bar{d}_{sub}) <$ Cost$(\bar{d}_{best})$ **then**
      $\bar{d}_{best} \leftarrow \bar{d}_{sub}$
  $\bar{d}_{prune} \in$ Substitute$(\bar{d}_{prune}, t_s, t_e, \bar{d}_{best})$