Context-aware Cost Shaping to Reduce the Impact of Model Error in Receding Horizon Control

Christopher D. McKinnon and Angela P. Schoellig

***Abstract*— This paper presents a method to enable a robot using stochastic Model Predictive Control (MPC) to achieve high performance on a repetitive path-following task. In particular, we consider the case where the accuracy of the model for robot dynamics varies significantly over the path–motivated by the fact that the models used in MPC must be computationally efficient, which limits their expressive power. Our approach is based on correcting the cost predicted using a simple learned dynamics model over the MPC horizon. This discourages the controller from taking actions that lead to higher cost than would have been predicted using the dynamics model. In addition, stochastic MPC provides a quantitative measure of safety by limiting the probability of violating state and input constraints over the prediction horizon. Our approach is unique in that it combines both online model learning and cost learning over the prediction horizon and is geared towards operating a robot in changing conditions. We demonstrate our algorithm in simulation and experiment on a ground robot that uses a stereo camera for localization.**

I. INTRODUCTION AND RELATED WORK

Model Predictive Control (MPC) has been demonstrated as an effective tool for tasks such as path-following. In particular, repetitive path-following is relevant for many applications including package delivery, patrol, and transit. The main challenge is that robot dynamics can change, e.g., due to weather or added payload, and are often not known exactly. Errors in the model for the robot dynamics (dynamics model) affect the cost computed in MPC and hence tracking performance. We present a method to take into account errors in predicting the cost over the MPC horizon and hence improve tracking performance; especially when the accuracy of the dynamics model varies over the path both spatially and temporally.

The most common way to improve the performance of Model Predictive Control (MPC) is to improve the dynamics model. Researchers in this area continue to demonstrate new and improved methods for learning robot dynamics. Many of these methods are based on Gaussian process regression [1]–[6], local linear regression [7]–[11], and neural networks [12], [13]. All of these methods, however, limit the expressive power of the model in some way to meet the computational requirements of MPC, which requires solving a (nonlinear) optimization problem at each sampling time. For GP-based methods, the number of basis points is limited and kernel hyperparameters are often fixed. Both neural networks and local linear regression assume a fixed form for the system dynamics in terms of a fixed feature or neural network architecture. This can limit the performance of these algorithms because the accuracy of the chosen function approximation may vary–especially if the robot is deployed in changing operating conditions. In addition, such models are usually trained to predict for one time-step in contrast to the long horizons used in MPC. Making accurate multi-step predictions remains an active area of study [14], [15].

Alternatively to improving the model for robot dynamics, the methods in [16]–[18] learn the value function– the discounted sum of rewards from a given state to the completion of the task–from data and leverage MPC for efficient short-term trajectory optimization. In [16], [17], the authors iteratively expand a set of sampled safe states–and their associated value–so that an MPC-based controller can minimize the total cost to complete the task by using a model-based prediction over the MPC horizon and the value of the sampled safe states for the cost of the final state in the horizon. In [18], the authors learn the value function using function approximation and showed how combining it with receding horizon control could accelerate learning. While estimating the value function using data does improve performance, these methods still rely on a dynamics model for making predictions over the MPC horizon. In contrast, we correct the cost over the MPC horizon to account for varied model accuracy. We also account for changes in operating conditions by inferring which previous runs resulted in similar cost prediction error to the current run.

In addition to learning just the value function, the authors of [19] proposed a method to modify the cost function of MPC over the prediction horizon and demonstrated it in simulation. The goal was to enable a short-horizon controller to mimic the behaviour of a longer-horizon controller computed offline using a better model for robot dynamics. In this case, the long-horizon controller was still model-based. In contrast, we correct the cost over the MPC horizon to account for cost prediction errors based on past experience directly.

Our method is inspired by ideas from reward shaping, whereby the reward function (which provides rewards for incremental actions) is modified to encourage and discourage behaviours that lead to high and low reward respectively [20]. Reward shaping has been shown to be effective for model-based learning in simple scenarios where safety is not a major consideration [21] as well as for episodic tasks [22] like repetitive path following. Our method builds on these ideas to improve performance of a physical robot in challenging outdoor conditions in addition to leveraging stochastic MPC to account for model uncertainty. We use the term cost learning to be consistent with the MPC literature

The authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. email: chris.mckinnon@mail.utoronto.ca, schoellig@utias.utoronto.ca

which considers a cost function rather than a reward function.

In this paper, we present a practical algorithm to improve tracking performance for repetitive path following that leverages the combination of model learning (to efficiently make predictions of the cost associated with a sequence of control actions) with cost learning (to account for systematic changes in the accuracy of these predictions). Intuitively, this is like how humans practice driving to understand how a car handles, but drive at a reasonable speed where we understand the cost/risk of our actions; our approach increases the cost of taking actions where the model under-estimates the cost of taking that action. Second, we show how this algorithm can be integrated seamlessly with a state-of-the-art path-following controller based on stochastic MPC which can account for model uncertainty. Finally, we demonstrate the proposed algorithm in both simulation and experiment and provide a thorough analysis of the effect of the proposed algorithm on several aspects of controller performance.

II. PROBLEM STATEMENT

We consider a ground robot performing a repetitive path following task using stochastic MPC and a model learning algorithm to adapt to changes in dynamics. In reality, the robot state $\mathbf{s}$ evolves according to dynamics $\mathbf{h}^{true}(\cdot)$. However, we only have access to an approximate dynamics model $\mathbf{h}(\cdot)$ that depends on $\mathbf{s}$ and control actions $\mathbf{u}$ and has accuracy that varies over the length of the path. Let $\ell(\mathbf{s}, \mathbf{u})$ be the non-negative scalar cost associated with applying $\mathbf{u}$ in state $\mathbf{s}$ and $V_f(\mathbf{s})$ be the non-negative scalar cost of the final state in the horizon of length H. In addition, let $\mathbf{s}$ and $\mathbf{u}$ be constrained to be within the sets $\mathcal{S}$ and $\mathcal{U}$ with a small acceptable probability of violating these constraints $\epsilon^{\mathbf{s}}$ and $\epsilon^{\mathbf{u}}$. The resulting receding horizon control problem solved at each sampling time k is then:

$$\min_{\bar{\mathbf{s}},\bar{\mathbf{u}}} \quad V_f(\bar{\mathbf{s}}_H) + \sum_{i=0}^{H-1} \ell(\bar{\mathbf{s}}_{k+i}, \bar{\mathbf{u}}_{k+i}) \tag{1}$$

$$\text{s.t.} \quad \mathbf{s}_{k+i+1} = \mathbf{h}(\mathbf{s}_{k+i}, \mathbf{u}_{k+i}), \qquad i = 0...H-1, \tag{2}$$

$$p(\mathbf{s}_{k+i+1} \in \mathcal{S}) \geq 1 - \epsilon^{\mathbf{s}}, \qquad i = 0...H-1, \tag{3}$$

$$p(\mathbf{u}_i \in \mathcal{U}) \geq 1 - \epsilon^{\mathbf{u}}, \qquad i = 0...H-1, \tag{4}$$

$$\mathbf{s}_k \sim \mathcal{N}(\bar{\mathbf{s}}_k, \mathbf{\Sigma}_k^{\mathbf{ss}}), \tag{5}$$

where the state $\mathbf{s}_k$ is given with a mean $\bar{\mathbf{s}}_k$ and covariance $\mathbf{\Sigma}_k^{\mathbf{ss}}$. The decision variable is the mean of the state $\bar{\mathbf{s}}$ and the input $\bar{\mathbf{u}}$, which we also use for computing the cost (1).

The goal of our approach is to minimize the cost (1) incurred in closed loop when $\mathbf{h}^{true}(\cdot) \neq \mathbf{h}(\cdot)$.

III. METHODOLOGY

In this section, we present our approach and how it fits into a state-of-the-art Stochastic MPC formulation that includes online model learning [7]. Our approach is based on modelling the difference between the cost (1) over the prediction horizon predicted using the approximate model for robot dynamics (2) and the cost actually incurred over the horizon. By parametrizing this difference in terms of a variable that can be controlled, we enable the controller to automatically adjust vehicle behaviour to reduce the impact of model mismatch on minimizing the cost function (1) in closed loop. We also use a measure of similarity to previous traversals of the path to construct a local cost correction model using the most relevant data. In this paper, we calculate cost using the mean states and inputs. Our approach would also apply if we used the expected value of the cost because uncertainty predictions are also model-based.

We denote scalars using lower-case characters and functions are followed by round brackets. Vectors are lowercase boldface characters and matrices are uppercase boldface characters.

A. Cost Prediction Error

Let the sampling time k refer to each time a control is computed and timestep i refer to a point along the prediction horizon at a given sampling time. At each sampling time k, we compute the optimal sequence $\{\mathbf{s}_{k+i+1}, \mathbf{u}_{k+i}\}_{i=0}^{H-1}$ and apply $\mathbf{u}_k$ to the robot. After repeating this for H sampling times, we have the actual states the robot visited and the controls that were applied over the initial horizon at time k. This gives us both the cost predicted at sampling time k for the i^{th} step in the horizon, $\ell_{k,i}$, and the actual cost incurred at sampling time $k+i$, $\ell_{k+i,0}$. We can then compute the cost prediction error $\delta\ell_{k,i}$ for sampling time k at timestep i:

$$\underbrace{\ell_{k+i,0}}_{\text{actual}} = \underbrace{\ell_{k,i}}_{\text{predicted}} + \underbrace{\delta\ell_{k,i}}_{\text{cost prediction error}} . \tag{6}$$

We assume that the distribution of $\delta\ell$ depends on a quantity $\mathbf{a}$ which can be predicted using the model for robot dynamics and how far the timestep is along the horizon, i. This results in a dataset $\{\delta\ell_{k,i}, \mathbf{a}_{k,i}\}$ for each timestep $i = 1...H-1$ in the horizon at each sampling time $k = 0...N-H-1$.

In general, $\mathbf{a}$ should be related to a state of the vehicle that affects model accuracy and can be altered using the control inputs. For ground robots, factors that are hard to model often become more significant as speed increases which makes speed a natural candidate. This choice will be platform-dependent and require some expert knowledge to choose effectively. More variables can be included at the expense of data efficiency.

B. Data Management

Let a run be defined as a complete traversal of the path. Since we are considering a repetitive path-following task, we store data $\{\delta\ell_{k,i}, \mathbf{a}_{k,i}\}$ indexed by location along the path and run number. This automatically encodes factors such as local terrain properties (for ground vehicles) and local path geometry that influence the cost prediction error.

C. Model for Cost Prediction Error

The main properties of $\delta\ell_{k,i}(\cdot)$ that we wish to capture are: *(i)* the dependence on $\mathbf{a}$ which may be nonlinear, *(ii)* heteroscedasticity since the predictions of cost may be more or less precise depending on the value of $\mathbf{a}$, and *(iii)* a

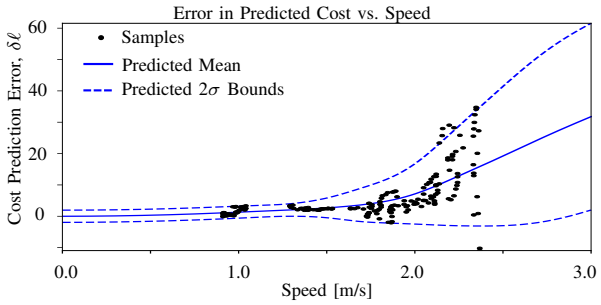


Fig. 1: Example of $\delta\ell$ plotted as a function of forward speed taken from an experiment in snowy conditions. The black dots are samples, the solid blue line is the estimated mean from a model $\delta\ell(\cdot)$, and the dotted lines are upper and lower 2σ bounds. In this case, the variance of $\delta\ell$ changes substantially between 1.5 and 2.5 m/s which can be incoroporated into the cost function for MPC by augmenting the cost with an upper percentile (e.g. 2σ) of the cost rather than the mean to discourage actions that produce a high variance in cost prediction error.

principled mechanism to weight data from previous runs according to a measure of similarity to the current run. One model that has these properties is the mixture of experts [23]. Specifically, we will consider a mixture of experts based on Bayesian linear regression where each expert is fit to data from one run c. Predictions for $\delta\ell_{k,i}$ (dropping subscripts for compactness) are made using:

$$p(\delta\ell|\mathbf{a}) = \sum_c p(\delta\ell|\mathbf{a}, c)p(c|\mathbf{a}), \tag{7}$$

where each expert is:

$$p(\delta\ell|\mathbf{a}, c) = \mathbf{w}_c^T\mathbf{a} + \eta_c, \qquad \eta_c \sim \mathcal{N}(0, \sigma_c^2), \tag{8}$$

and the gating function is a mixture of Gaussians:

$$p(c|\mathbf{a}) \propto p(\mathbf{a}|c)p(c), \tag{9}$$

where $p(\mathbf{a}|c)$ is a Gaussian describing the density of $\mathbf{a}$ for run c and $p(c)$ is the prior probability of observing data from the same distribution as run c. Parameters $\mathbf{w}_c$ and σ_c^2 are updated continuously and so depend on k. Property *(i)* can be achieved through the feature $\mathbf{a}$ (which can be a nonlinear transformation of the state) and the mixing weights, *(ii)* is achieved since each expert c can have a different variance, σ_c^2, and *(iii)* can be achieved through setting the prior $p(c)$ based on recent observations, which will be discussed in the next section. To account for the fact that cost prediction error varies over the prediction horizon (generally increasing further into the future as prediction error accumulates), we partition the horizon into multiple sections and fit a separate mixture model to each grouping of i.

D. Cost-based Mode Inference

We assume that $\delta\ell$ is dependent on $\mathbf{a}$ and i, which can be measured directly, and additional factors cannot be measured directly which we will call the mode. Let $\mathcal{D}_{\delta\ell}^- = \left\{\delta\ell_j^-, \mathbf{a}_j^-\right\}_{j=0}^n$ be the n most recent measurements of predicted cost error from the current run for a particular section of the horizon. We will follow [7] and make the run prior $p(c)$ dependent on recent data so it becomes:

$$p(c|\mathcal{D}_{\delta\ell}^-) \propto med\left(p(\delta\ell_j^-|\mathbf{a}_j^-, c)\right) \qquad j = 0...n \tag{10}$$

where $med(\cdot)$ is the median. Here, we have departed from the conventional assumption that data is i.i.d. (which would lead to $med(\cdot)$ being replaced by a product). Our main motivation was that this produced more consistent estimates of $p(c|\mathcal{D}_{\delta\ell}^-)$ in experiment.

E. Augmented MPC Cost Function

Now that we have a model for $\delta\ell(\cdot)$ we can augment the cost function used to compute the optimal control sequence in (1). This results in the following cost function for MPC:

$$V_f(\mathbf{s}_H) + \sum_{i=0}^{H-1} \ell(\mathbf{s}_{k+i}, \mathbf{u}_{k+i}) + \delta\ell^u(\mathbf{a}_{k+i}) \tag{11}$$

where $\delta\ell^u$ is an upper percentile of $p(\delta\ell|\mathbf{a})$. See Fig. 1 for a visualization of what the cost function may look like and how using different percentiles change the additional cost. Our main contribution in this paper is to learn $\delta\ell(\cdot)$ and demonstrate the effectiveness of our approach on a robot in changing conditions.

F. Model Learning

Here, we give a brief summary of the approach for learning robot dynamics used in this paper. For a detailed description, see [7]. We consider (2) to be of the form:

$$\mathbf{s}_{k+1} = \mathbf{s}_k + dt\mathbf{g}_k(\mathbf{s}_k) + \boldsymbol{\eta}_k, \qquad \boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_k^\eta), \tag{12}$$

where $\mathbf{g}_k(\cdot)$ is *partially* known ahead of time and the timestep dt is fixed and known. Each component $g_k(\cdot)$ of $\mathbf{g}_k(\cdot)$ that is *unknown* is learned independently using Bayesian linear regression (BLR):

$$g_k(\mathbf{s}_k) = \mathbf{w}_k^T\mathbf{s}_k + \eta_k, \tag{13}$$

where the joint distribution for $\mathbf{w}_k$ and σ_k^2 is a Normal-Inverse-Gamma distribution. Parameters $\mathbf{w}_k$ and σ_k^2 are determined given observations of g_k and $\mathbf{s}_k$. The prior for $\mathbf{w}_k$ and σ_k^2 is updated at each sampling time using data from the current run; the posterior is computed using data from previous runs weighted by the similarity of the robots recent dynamics to the robots dynamics in each previous run over the recent section of the path.

IV. EXPERIMENTAL SETUP

This section outlines how we apply our method to the Clearpath Grizzly shown in Fig. 2. See http://tiny.cc/cost-model-learning for a video of our experiments.

Fig. 2: The Clearpath Grizzly in **nominal** (left) and **loaded** configurations (right). The mass of the bags of gravel was 133 kg.

A. Robot Model

Let $\mathbf{s} = [x, y, \theta, v, \omega]^T$, be the 2D position, heading, forward speed and turn rate of the robot, and $\mathbf{u} = [v^{cmd}, \omega^{cmd}]^T$ be the commanded forward speed and turn rate of the robot. We model the dynamics of $\mathbf{s}$, $\mathbf{h}(\cdot)$, as a unicycle with first order dynamics for the forward speed and the turn rate:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ \omega_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ \omega_k \end{bmatrix} + dt \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \\ \omega_k \\ [v_k^{cmd}, v_k]\mathbf{w}_k^v + \eta_k^v \\ [\omega_k^{cmd}, \omega_k]\mathbf{w}_k^\omega + \eta_k^\omega \end{bmatrix}. \quad (14)$$

where $\eta_k^v \sim \mathcal{N}(0, \sigma_{v,k}^2)$ and $\eta_k^\omega \sim \mathcal{N}(0, \sigma_{\omega,k}^2)$. We learn $\mathbf{w}_k^v$, $\sigma_{v,k}^2$, $\mathbf{w}_k^\omega$, and $\sigma_{\omega,k}^2$ using the method covered in Sec. III-F.

B. Implementation

Our algorithm was implemented in C++ on an Intel i7 2.70 GHz 8 core processor with 16 GB of RAM. Our controller relies on a vision-based system, Visual Teach and Repeat [24], for localization, which runs on the same laptop. The controller runs at 10 Hz with a three second look-ahead discretized by 30 points. The optimization problem (1)-(4) is solved as a sequential quadratic program and re-linearized three times, taking an average of 45 ms. We use a quadratic penalty for $\ell(\cdot)$ where the weights on tangential, lateral, heading, speed, and turn rate error are 100, 200, 100, 2, and 2 respectively. The weights on deviation of commanded speed, turn rate, and reference speed from their references are 1, 1, and 40 respectively. The weights on rate of change of commands in the same order are 10, 15, and 5.

The proposed cost error model is updated at 5 Hz in a separate thread from the controller. This keeps the most computationally expensive step of the proposed approach, fitting the model, separate from the control loop. For all experiments, we use $\mathbf{a} = v^2$ because we expect similar cost prediction error driving forwards and backwards. We partition the horizon into two segments for all of our experiments.

V. EXPERIMENTS

To demonstrate the effectiveness of the proposed algorithm in experiment, we tested our algorithm on a Clearpath Grizzly driving in various outdoor environments. For all figures showing a distribution of a quantity, we show the 25th, 50th, and 75th percentiles.

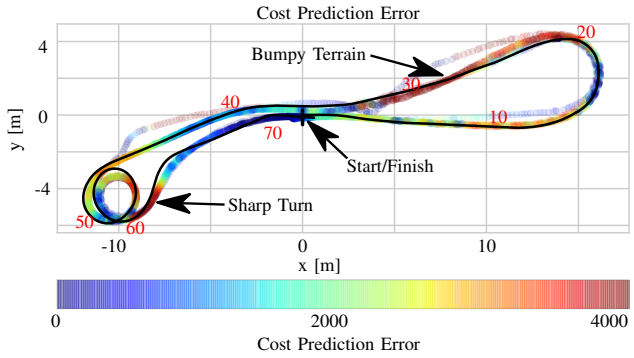


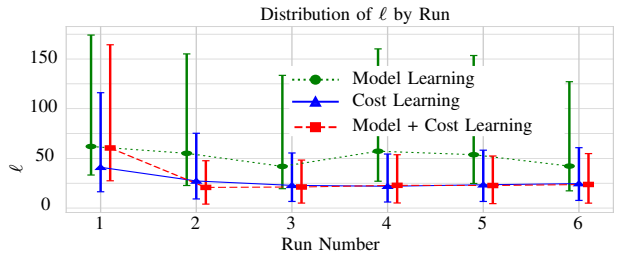
Fig. 3: Sum of cost prediction error over the horizon at each sampling time superimposed for 6 runs. Distance along the path in meters is marked in red. For reference, the actual cost over the horizon ranges from 300 to 8000 depending on the section of the path. The cost prediction error is consistently large over a bumpy section of the path and around a sharp turn near the end of the path. The vehicle has limited suspension so driving over the bumpy section of the path produces significant disturbances that are hard to model. See our video: http://tiny.cc/cost-model-learning.

A. Cost Learning vs. Model Learning

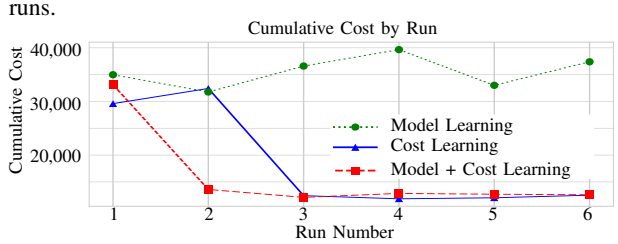
In this experiment, we compare the effectiveness of model learning and cost learning separately to the combination of both. Cost learning alone means that the vehicle uses a fixed model for the robot dynamics that is fit using previous data but learns $\delta\ell(\cdot)$. Model learning alone means that the controller uses the online model learning algorithm presented in [7] and $\delta\ell(\cdot) = 0$. The combination means that both the model and cost are learned online over the course of the run. For these experiments, we use the mean value of $\delta\ell(\cdot)$ to augment the MPC cost function (11). We drove the vehicle 6 times around a 73 m course (shown in Fig. 3). The two main sources of model error in this case are a bumpy section of the path, which causes significant but non-repeatable disturbances at high speed, and a sharp turn, which causes significant side-slip (not in the model) and skidding. The vehicle was in the nominal configuration (see Fig. 2) for these experiments.

To measure the effect of each component of the proposed algorithm, we first look at differences in the control cost along the path. In this experiment, the main effect of adding cost learning is for the vehicle to slow down over the section of the path with bumps and sharp turns. Fig. 4a shows that the distribution of ℓ over each run is significantly lower when using cost learning compared to model learning. Similarly, Fig. 4b shows a lower cumulative cost to traverse the path with cost learning compared to model learning. In both cases, the combination of cost learning and model learning convergences to low cost in the fewest runs. In this experiment, we used a good initial guess for the dynamics model–(14) with parameters identified using data from similar conditions–so cost learning alone achieved good results. For a different initial guess, the best performance could be arbitrarily bad.

To gain insight into why, we investigate the effect of cost learning on model accuracy. Intuitively, adding $\delta\ell(\cdot)$ should discourage the system from taking actions that result in high model error since this leads to high cost prediction error.



(a) The distribution of ℓ for each run. The combination of cost and model learning converges to the lowest ℓ in the fewest runs.



(b) Cumulative cost to traverse the path by run. The combination of cost learning and model achieves the lowest cost in the fewest run.

Fig. 4: A comparison of the cumulative cost and ℓ with **cost learning**, **model learning** and **cost learning + model learning** on the course shown in Fig. 3. Cost learning reduces the cost to traverse the path, both in the average cost and cumulative cost, after just three runs. When combined with model learning, it converges after just one run. Model learning on its own is not sufficient to achieve the same performance making little improvement in the average cost and no improvement in the cumulative cost.

We measure prediction accuracy of the mean states over the MPC horizon using Multi-step RMSE (M-RMSE) and of the uncertainty using multi-step RMS Z-score (M-RMSZ). For the speed v, the M-RMSE at sampling time k is calculated by comparing the predicted mean at each timestep i along the horizon $\bar{v}_{k,i}$ to the measured mean at the corresponding sample time $\bar{v}_{k+i,0}$:

$$\text{M-RMSE}_k = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(\bar{v}_{k,i} - \bar{v}_{k+i,0})^2}, \qquad (15)$$

and M-RMSZ is calculated the same way but each term in the sum is normalized by the predicted variance $\sigma^2_{v\ k,i}$. Ideally, M-RMSE would be low and M-RMSZ would be around one.

Figure 5 shows the M-RMSE and RMSZ for speed over the course of 6 runs with model learning vs. model and cost learning. The combination of cost and model learning reduces the average M-RMSE for speed and turn rate by 50% and 46% respectively and reduces the number of times that the M-RMSZ jumps significantly above one.

While using a richer model may improve prediction accuracy, cost correction provides a means of adapting the vehicle's behaviour in cases where the dynamics model accuracy varies along the path which is useful for driving in challenging off-road conditions.

B. Cost-based Mode Inference

In the previous section, we showed that cost learning reduces the cost of traversing a path when the dynamics were

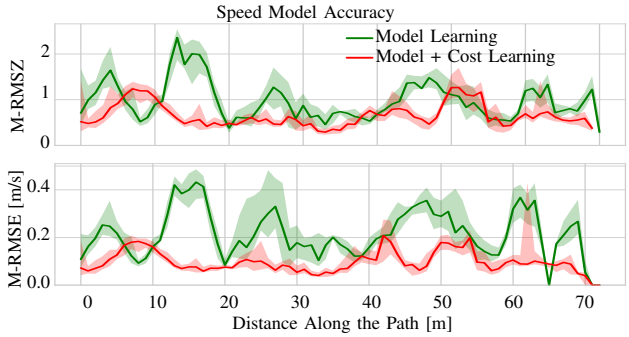
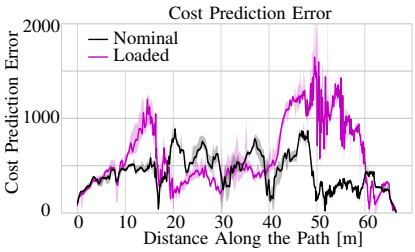


Fig. 5: A comparison of the multi-step RMS Error (M-RMSE) and Z-score (M-RMSZ) when using only **model learning** and **cost learning + model learning** on the course shown in Fig. 3. The addition of cost learning improves prediction accuracy (lower M-RMSE) and uncertainty estimates (M-RMSZ closer to 1.0).

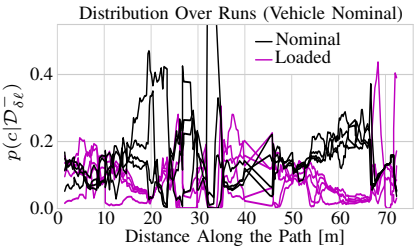
the same between runs. For the experiments in this section, we changed the robot dynamics between runs by adding or removing a payload of gravel bags (see Fig. 2). The path was similar to the one shown in Fig. 3. For these experiments, we used a fixed dynamics model and only vary whether or not mode inference (Sec III-D) is enabled to isolate the effect of this component of the algorithm. When mode inference is not enabled, $p(c|\mathcal{D}^-_{\delta\ell})$ is uniform.

First, we investigate whether the mode inference recommends runs where the vehicle was in a similar configuration. Figure 6a shows the predicted cost error for 15 traversals of the path coloured by vehicle configuration. Here, we see that adding payload increased $\delta\ell$ between 40 and 60 m when the vehicle was turning sharply. The proposed algorithm correctly infers that $\delta\ell$ is the most similar to runs when the vehicle is in the same configuration as the live run, especially between 45 and 65 m (e.g. see Fig. 6b for the estimated probability of each run during run 10 when the vehicle was in the nominal configuration). There is a slight delay because the algorithm uses a sliding window of previous experiences to estimate $p(c|\mathcal{D}^-_{\delta\ell})$. Finally, Fig. 6c shows that the algorithm consistently selects the majority of experiences from a run where the vehicle was in a similar configuration with the exception of run 6, which is the first run in the nominal configuration. Runs in a different configuration from the live run receive high $p(c|\mathcal{D}^-_{\delta\ell})$ some fraction of the time, however Fig. 6a and 6b show that this is over sections of the path when $\delta\ell$ is similar between configurations so this would have little impact on the model $\delta\ell(\cdot)$.

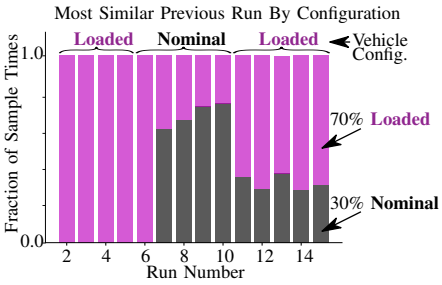
Second, we show that including mode inference makes a difference to the control performance over the section of the path where the cost prediction error is different between the two configurations (between 40 and 60 m along the path). Figure 7a shows that the distribution of ℓ incurred over the path is lower when mode inference is enabled after the algorithm has one run in each configuration. Figure 7b shows that the cumulative cost to traverse the path from 40-60 m is also reduced when mode inference is enabled after the vehicle has completed one run in each configuration.



(a) Distribution of the sum of $\delta\ell$ over the prediction horizon at each sampling time along the path for 15 runs coloured by vehicle configuration.

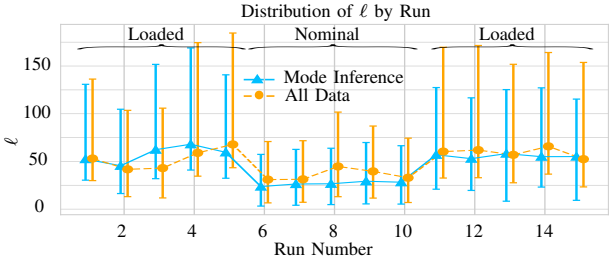


(b) Estimated $p(c|\mathcal{D}^-_{\delta\ell})$ for each previous run when the vehicle was in the **nominal** configuration, during run 10. Each line is coloured by the actual vehicle configuration during that previous run.

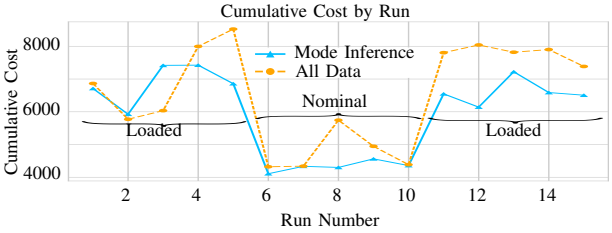


(c) For each run, the fraction of the time the previous run with the maximum $p(c|\mathcal{D}^-_{\delta\ell})$ was from each configuration.

Fig. 6: The vehicle was driven for 15 runs alternating between the **loaded** and **nominal** configuration. The sum of cost prediction error along the horizon at each sampling time, depicted in *(a)*, shows that the cost prediction error differs significantly between 40 and 60 m along the path, which corresponds to a sharp turn. Over this section of the path, *(b)* shows that during run 10, when the vehicle was in the nominal configuration, mode inference correctly identifies that runs in the nominal configuration are more similar since those runs are assigned a higher $p(c|\mathcal{D}^-_{\delta\ell})$. Finally, *(c)* shows that the run with the highest $p(c|\mathcal{D}^-_{\delta\ell})$ consistently came from a run with the vehicle in the same configuration as the live run with the exception of run 6 when the vehicle has no previous experience in the nominal configuration. Mixing experience from different configurations is acceptable when the cost prediction error is similar for both configurations since the resulting model $\delta\ell(\cdot)$ will therefore also be similar.



(a) The distribution of ℓ incurred over each traverse of the path with and without mode inference enabled. After run 6, when the vehicle has experience from both configurations, adding mode inference consistently lowers the step cost throughout the run.

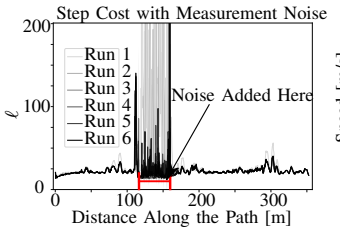


(b) The cumulative cost for the vehicle to go from 40-60 m. Including mode inference for learning the cost prediction error consistently improves performance after the algorithm has experience in each configuration.

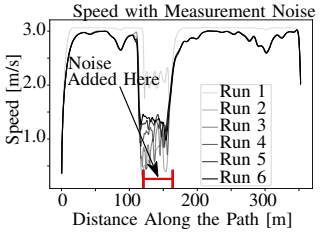
Fig. 7: A comparison of the distribution of ℓ and the cumulative cost to traverse the path shown with **mode inference** compared to using **all previous data** to construct $\delta\ell(\cdot)$. Adding experience recommendation consistently reduces the cost after run 6, when the vehicle has experience from both configurations.

C. State Dependent Measurement Noise in Simulation

In addition to model error, measurement noise can also contribute to cost prediction error since the measured state provides the initial condition for rolling out the sequence of control actions over the prediction horizon. Figure 8 shows how the proposed algorithm reduces the cost to traverse the path when we added zero-mean Gaussian noise with variance proportional to v^2 over a straight section of the path. The



(a) The cost ℓ along the path for each run when cost learning is enabled.



(b) The vehicle's forward speed with each run when cost learning is enabled.

Fig. 8: Results of enabling the cost learning in simulation with speed dependent noise. The system learns to drive slowly over the section of path with speed dependent noise to reduce the associated cost.

model for robot dynamics was otherwise perfect. We used the upper 2σ bound on the cost prediction error to augment the cost function to account for the random nature of the cost induced by the noise. The algorithm learns that there is a large additional cost associated with driving quickly over this section of the path so reduces the speed (Fig. 8b) and, consequently, the cost (Fig. 8a).

VI. Conclusion

In this paper, we proposed a new method combining cost and model learning for a ground robot performing a repetitive path-following task. We demonstrated in simulation and experiment that the combination of model and cost learning out-performed either component separately. While model learning is effective if the form of the dynamics is known or there is sufficient data to train a complex model, cost learning is useful when the dependence of model accuracy is known but the form of the dynamics is not or data is scarce. By combining both, we can improve performance when form of the dynamics is known (model learning) and avoid states that induce higher cost than predicted (cost learning). We encourage the reader to watch our video at `http://tiny.cc/cost-model-learning` showing the experiments conducted in this paper.

REFERENCES

[1] C. D. McKinnon and A. P. Schoellig, "Learning Multi-Modal Models for Robot Dynamics with a Mixture of Gaussian Process Experts," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 322–328.

[2] L. Hewing, A. Liniger, and M. Zeilinger, "Cautious NMPC with gaussian process dynamics for autonomous miniature race cars," in *Proc. of the European Control Conf. (ECC)*, 2018, pp. 1341–1348.

[3] C. D. McKinnon and A. P. Schoellig, "Experience-Based Model Selection to Enable Long-Term, Safe Control for Repetitive Tasks Under Changing Conditions," in *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 2977–2984.

[4] F. Meier and S. Schaal, "Drifting Gaussian Processes with Varying Neighborhood Sizes for Online Model Learning," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 264–269.

[5] B. Niekerk, A. Damianou, and B. Rosman, "Online Constrained Model-based Reinforcement Learning," in *Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[6] J. Kabzan, L. Hewing, A. Liniger, and M. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[7] C. D. McKinnon and A. P. Schoellig, "Learn Fast, Forget Slow: Safe Predictive Learning Control for Systems with Unknown and Changing Dynamics Performing Repetitive Tasks," *Robotics and Automation Letters*, 2019.

[8] L. Jamone, B. Damas, and J. Santos-Victor, "Incremental Learning of Context-dependent Dynamic Internal Models for Robot Control," in *Intl. Symp. on Intelligent Control (ISIC)*, 2014, pp. 1336–1341.

[9] V. Desaraju, A. Spitzer, and N. Michael, "Experience-driven Predictive Control with Robust Constraint Satisfaction under Time-Varying State Uncertainty," in *Proc. of the Robotics: Science and Systems Conf. (RSS)*, 2017.

[10] J. Ting, A. D'Souza, S. Vijayakumar, and S. Schaal, "A Bayesian Approach to Empirical Local Linearization for Robotics," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 2860–2865.

[11] C. D. McKinnon and A. P. Schoellig, "Learning Probabilistic Models for Safe Predictive Control in Unknown Environments," in *Proc. of the European. Conf. Control (ECC)*, 2019, in press.

[12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information Theoretic MPC for Model-Based Reinforcement Learning," in *Proc. of the International on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[13] N. Mohajerin and S. Waslander, "Multi-Step Prediction of Dynamic Systems with Recurrent Neural Networks," *Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2019.

[14] A. Venkatraman, B. Boots, M. Hebert, and J. Bagnell, "Data as Demonstrator with Applications to System Identification," in *ALR Workshop, NIPS*, 2014.

[15] A. Doerr, C. Daniel, D. Nguyen-Tuong, A. Marco, S. Schaal, T. Marc, and S. Trimpe, "Optimizing Long-term Predictions for Model-based Policy Search," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2017, pp. 227–238.

[16] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for Iterative Tasks. A Data-driven Control Framework," *Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

[17] U. Rosolia, X. Zhang, and F. Borrelli, "Robust Learning Model Predictive Control for Iterative Tasks: Learning from Experience," in *Proc. of the Conf. on Decision and Control (CDC)*, 2017, pp. 1157–1162.

[18] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan Online, Learn Offline: Efficient Learning and Exploration via Model-based Control," *arXiv preprint arXiv:1811.01848*, 2018.

[19] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel, "Learning from the Hindsight PlanEpisodic MPC Improvement," in *Proc of the Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 336–343.

[20] A. Ng, D. Harada, and S. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," in *Proc of the Intl. Conf on Machine Learning (ICML)*, vol. 99, 1999, pp. 278–287.

[21] J. Asmuth, M. Littman, and R. Zinkov, "Potential-based Shaping in Model-based Reinforcement Learning," in *AAAI*, 2008, pp. 604–609.

[22] M. Grześ, "Reward Shaping in Episodic Reinforcement Learning," in *Proc of the Conf. on Autonomous Agents and MultiAgent Systems*, 2017, pp. 565–573.

[23] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[24] M. Paton, F. Pomerleau, K. MacTavish, C. Ostafew, and T. Barfoot, "Expanding the Limits of Vision-based Localization for Long-term Route-following Autonomy," *Journal of Field Robotics (JFR)*, vol. 34, no. 1, pp. 98–122, 2017.