2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Experience-Based Model Selection to Enable Long-Term, Safe Control for Repetitive Tasks Under Changing Conditions

Christopher D. McKinnon and Angela P. Schoellig

Abstract— **Learning approaches have enabled significant performance improvements in robotic control allowing robots to execute motions that were previously impossible. The majority of the work to date, however, assumes that the parts to be learned are static or slowly changing, which limits their applicability in realistic scenarios with rapid changes in the conditions. This paper presents a method to extend an existing single-mode safe learning controller based on Gaussian Process Regression to learn an increasing number of non-linear models for the robot dynamics. We show that this approach enables a robot to re-use past experiences from a large number of previously visited operating conditions, and to safely adapt when a new and distinct operating condition is encountered. This allows the robot to achieve safety and high performance in a large number of operating conditions that do not have to be specified ahead of time. Our approach runs independently from the controller, imposing no additional computation time on the control loop regardless of the number of previous operating conditions considered. We demonstrate the effectiveness of our approach in experiment on a 900 kg ground robot with both physical and artificial changes to its dynamics. All of our experiments are conducted using vision for localization.**

I. INTRODUCTION

At the core of most control algorithms is a model that captures the relationship between the state, input, and dynamics of a robotic system. During tasks such as repetitive path following, which are common in mining, agriculture, and logistics, a robot continuously gathers data which can be used to refine this model [1]. Over time, this enables the controller to exploit well-known, high-reward actions [2]. An accurate assessment of the risk associated with taking a control action, especially if it has not been taken before, is important during this process [3], [4]. Using an assessment of this risk to ensure safety is known as safe learning.

This paper presents a safe learning approach (see Fig. 1) for the case when the operating conditions can change over time, which is a significant extension over previous work. We demonstrate the effectiveness of our approach in experiment.

Safe learning methods generally incorporate an approximate initial guess for the system dynamics with some bounds on the modelling error incurred in the approximation [5], [6]. A learning term then refines the initial guess over time using experience data to better approximate the true dynamics. The goal is to guarantee that the system does not violate safety constraints (e.g., limits on the control input or path tracking error) while achieving the control objective (e.g., following a path) and at the same time improving the model of the system and, consequently, its task performance over time. Most learning algorithms learn a single model for the system dynamics or use multiple models that are trained ahead of time based on appropriate training data from operating the robot in all relevant conditions [5], [7]–[11]. This presents a challenge for robots that are deployed into a wide range of operating conditions which may not all be known ahead of time.

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

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under the grant RGPIN-2014-04634

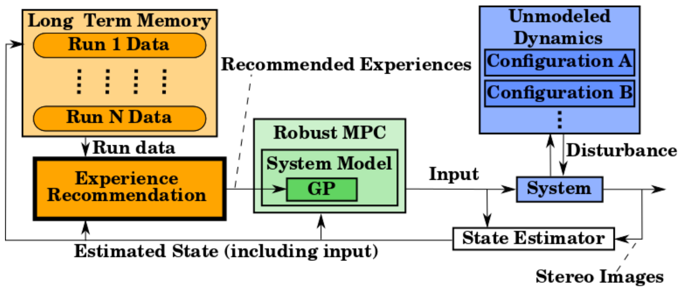


Fig. 1: Block diagram showing the proposed experience recommendation in closed-loop with a safe controller. The system dynamics can change from one run to another. The proposed experience recommendation chooses the experiences from previous runs that best match the current dynamics. These experiences are used in a Gaussian Process (GP) to model the system dynamics. The robust Model Predictive Controller (MPC) is the controller from [5].

This paper builds on work in [5], which proposes a robust, learning-based Model Predicitve Controller (MPC) for repetitive path following using a vision-based localization algorithm [12]. The controller developed in [5] uses local Gaussian Process (GP) regression to construct a model for the dynamics at each time step. These models are valid over a small section of the path around the vehicle's current position. They are based on a fixed number of training points, or experiences, so have fixed computational cost. Our contribution is to generalize this approach to long-term safe learning with large and small, repeated changes in the dynamics that depend on the operating conditions or physical configuration of the robot (see Fig. 1). Examples are weather, terrain conditions [13], or payload configuration [14].

The proposed method builds local GPs of the robot dynamics on each previous run to select experiences from the run with the most similar dynamics. These experiences are used to construct the GP used for control. This allows the GP used for control to leverage knowledge related to changes in dynamics that are both sudden and gradual as long as a similar change has been observed in the past. We use the same hyper-parameters for all GP models. Accordingly, we introduce no additional model parameters and are able to assess how likely it is that the GP constructed from previous experiences will satisfy the assumptions of the safe controller, namely that the 3σ bounds on uncertainty are an accurate upper bound on model error. The local GPs used by our method are inexpensive to compute enabling the robot to learn over and leverage data from a large number of runs without having to 'forget' previous experiences. This is a significant advantage over previous methods.

II. RELATED WORK

Learning control has received a great amount of attention in recent years, most notably in the case of single-mode learning control. This is the broad class of learning methods that assumes the true (but initially unknown) mapping between the state at time k, $\mathbf{x}_k$, and the input $\mathbf{u}_k$, and the next state, $\mathbf{x}_{k+1}$, is one-to-one, or at least normally distributed according to some underlying process, $\mathbf{x}_{k+1} \sim \mathcal{N}(f(\mathbf{x}_k, \mathbf{u}_k), \boldsymbol{\sigma}_k)$. Recent developments have contributed controller designs with safety guarantees [6] and demonstrated impressive results in improved path following [15].

Multimodal safe control and path planning has also received a growing amount of attention. Applications include safely gliding a parafoil under a variety of wind conditions [9] and planning safe paths among uncertain agents such as pedestrians or automobiles [11]. The assumption and challenge in these cases is that the environment or obstacles in the environment have hidden states that change the dynamics but cannot be measured directly. Similar to our method, the algorithms try to infer this hidden state based on available observations and use this information for safe planning. An additional feature of our approach is that we attempt to build this model online and can deal with both discrete and continuous changes in dynamics.

Recent results in single-mode, safe learning control have taken great steps to improve performance while maintaining bounds on modelling error and therefore safety. Approaches in [5], [15]–[18] use GPs as corrective terms for approximate prior models and update them over time as more experience is gathered. A GP assumes a single underlying function with additive Gaussian noise. They are a perfect tool when there is only one mode for the dynamics or the mode can be measured directly. The inherent probabilistic bounds on the model error are used to allocate margin on safety constraints such that the system is robust to this model error. It is essential for safety that the bounds from the GP, usually some multiple of the standard deviation, bound the true model error with high probability. It is essential for high performance that the bounds are not unnecessarily conservative. A GP assumes a single underlying function with additive Gaussian noise, and is excellent when there is only one mode for the dynamics; however, if there are multiple dynamic modes (e.g., caused by driving both on snow and asphalt), the single GP must have overly conservative bounds to account for the dynamics in all modes (which may be significantly different), and may learn some combination of the dynamics in each mode which is sub-optimal in either mode. This is of particular concern to algorithms that update the GP online.

One approach that exhibits particularly good real-time performance and has been demonstrated in several real-world examples is presented in [5]. This approach continually reconstructs the GP disturbance model based on a fixed number of data points, to ensure the process model can be evaluated in constant time even if new experiences are added. Storing the data in first-in-first-out bins of fixed size allows the algorithm to update the data used in the GP in real time. If the mode changes, the model unlearns the existing mode by overwriting all of that data and relearns the new mode. During this process, it suffers from the same problems related to hyperparameters as mentioned above including either requiring over-conservative bounds to accommodate multiple modes, or having bounds that are realistic for a single-mode, but are unsafe while the model transitions between modes and is using data from more than one mode. Our method aims to overcome these limitations by only choosing data that is relevant to the current mode.

In addition to the single-mode, safe learning controllers, multimodal algorithms exist which identify a number of dynamic modes ahead of time using labelled or unlabelled training data and switch to the most likely model during operation [9], [11], [19]–[21]. This allows them to maintain persistent knowledge of a robot's dynamics across a wide range of operating conditions. Inferring the correct mode from measurements during operation allows them to maintain a high level of performance and robustness even when the mode is not directly observed. The method proposed in [22] for linear systems even infers the number of modes at training time. These approaches do, however, require that the number of modes and/or training data from each mode are available ahead of time, which can be a challenging task in real-world robotics applications. In contrast, our method does not require the number of modes or training data from each mode to be available ahead of time. Rather, it learns new dynamics as they arise during operation.

In our previous work, we presented an approach for learning multimodal dynamics by combining GPs and the Dirichlet Process, which is used in Bayesian non-parametric clustering models [14]. This allowed the robot to learn a new GP model for novel operating conditions and leverage an existing GP when the robot revisited an operating condition. The GPs represented the dynamics over the entire region of the state space and therefore required a large number of training points to be effective. These GPs could therefore not easily be used in the controller, which is limited to GPs with only a small number of training points due to computational constraints. The proposed method overcomes this by keeping the size of the GP used for control constant. In addition, the previous approach only used a relative measure of model quality. That is, it would only switch to the prior, safe mode if that described the current dynamics better than the existing set of GP experts. In this work, we add an explicit check to ensure that the assumptions made by the safe controller are valid with high probability.

In light of the current approaches and their limitations, the goal of this paper is to present a method for adapting to multiple dynamic modes over a long period of time with guarantees on safety using a realistic and computationally efficient representation of the system dynamics (including model uncertainty estimates). The aim is to design an algorithm for life-long model learning to achieve excellence in the relevant operating conditions regardless of whether they are known ahead of time.

III. PROBLEM STATEMENT

The goal of this work is to learn a model for the dynamics of a robot performing a repetitive, path-following task. The robot may be subjected to large changes in its dynamics due to factors such as payload, terrain, weather, or tyre pressure changes. We assume that these factors cannot be measured directly. The algorithm must scale to long-term operation and take advantage of repeated runs in the same operating conditions. The model should also include a reasonable estimate of the model uncertainty that acts as an upper bound on model error at all times.

Further assumptions can be summarized as follows:

- The mapping $(\mathbf{u}_k, \mathbf{x}_k) \rightarrow \mathbf{x}_{k+1}$ can be modelled as a GP for a single run in a local region along the path.
- The operating condition is constant over a short time horizon.
- The number of operating conditions and the mapping $(\mathbf{u}_k, \mathbf{x}_k) \rightarrow \mathbf{x}_{k+1}$ for each is not known ahead of time.

The system can be modelled by some nominal dynamics $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ with additive, initially unknown dynamics $\mathbf{g}^c(\mathbf{a}_k)$ that are specific to discrete or continuous operating conditions, c, and depend on features, $\mathbf{a}_k$, so

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{g}^c(\mathbf{a}_k). \tag{1}$$

The unknown dynamics are assumed to be a deterministic function with additive, zero-mean Gaussian noise,

$$\mathbf{g}^c(\mathbf{a}_k) = \mathbf{g}_0^c(\mathbf{a}_k) + \boldsymbol{\eta}^c, \tag{2}$$

where $\boldsymbol{\eta}^c \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\eta^c)$, and $\boldsymbol{\Sigma}_\eta^c$ is the measurement noise covariance.

IV. METHODOLOGY

In this section, we present our approach for long-term, safe learning control. Our approach makes extensive use of local GPs to model the robot dynamics.

A. Gaussian Process (GP) Disturbance Model

We model the unknown dynamics, $\mathbf{g}(\cdot)$, as a GP based on past observations. We drop the superscript $(\cdot)^c$ for notational convenience because we learn a GP for each operating condition separately. Since there are many good references on GPs, e.g. [23], here we provide only a high-level sketch. The learned model depends on previously gathered experiences, which are assembled from measurements of the state denoted by $\hat{\mathbf{x}}$ and the input $\mathbf{u}$ using (1), so that

$$\hat{\mathbf{g}}(\mathbf{a}_{k-1}) = \hat{\mathbf{x}}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}). \tag{3}$$

The resulting pair, $\{\mathbf{a}_{k-1}, \hat{\mathbf{g}}(\mathbf{a}_{k-1})\}$, forms an individual experience. For simplicity, we model each dimension of the disturbance using a separate GP. Below, we derive the equations for a single dimension of $\mathbf{g}(\cdot)$ denoted by $g(\cdot)$.

A GP is a distribution over functions given past experiences, $\mathcal{D} = \{\mathbf{a}_i, \hat{g}(\mathbf{a}_i)\}_{i=1}^m$, and kernel hyperparameters.

We assume the experiences are noisy observations of the true function $g(\mathbf{a}_k)$; that is, $\hat{g}(\mathbf{a}_k) = g(\mathbf{a}_k) + \eta$ where $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$. The posterior distribution is characterized by a mean and variance, which can be queried at any point $\mathbf{a}_*$ using

$$\mu(\mathbf{a}_*) = \mathbf{k}(\mathbf{a}_*)\mathbf{K}^{-1}\hat{\mathbf{g}}, \tag{4}$$

$$\sigma^2(\mathbf{a}_*) = \kappa(\mathbf{a}_*, \mathbf{a}_*) - \mathbf{k}(\mathbf{a}_*)\mathbf{K}^{-1}\mathbf{k}(\mathbf{a}_*)^T, \tag{5}$$

where $\hat{\mathbf{g}} = [\hat{g}(\mathbf{a}_1), ..., \hat{g}(\mathbf{a}_m)]^T$ is the vector of observed function values, the covariance matrix $\mathbf{K} \in \mathbb{R}^{m \times m}$ has entries $[\mathbf{K}(\mathbf{a}_i, \mathbf{a}_j)] = \kappa(\mathbf{a}_i, \mathbf{a}_j) + \sigma_\eta^2 \delta_{ij}$, where δ_{ij} is the Kronecker delta, and the vector $\mathbf{k}(\mathbf{a}_*) = [\kappa(\mathbf{a}_*, \mathbf{a}_1), ..., \kappa(\mathbf{a}_*, \mathbf{a}_m)]$ contains the covariances between the new test point $\mathbf{a}_*$ and the observed data points $\mathcal{D}$. For this work, we use the squared exponential kernel,

$$\kappa(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{L}^{-2}(\mathbf{a}_i - \mathbf{a}_j)\right) \tag{6}$$

because of its success in modelling robot dynamics [4], [5], [15], [17]. The hyperparameters are the diagonal matrix of length-scales, $\mathbf{L}$, which are inversely related to the importance of each element of $\mathbf{a}$, and the process noise variance, σ_f^2, which is the variance of the prior family of functions represented by $g(\cdot)$.

As training data is added to a particular GP, uncertainty is reduced and the posterior distribution of the GP specializes to a particular family of functions that represents the system dynamics in a particular operating condition. The job of the experience recommendation, which is the contribution of this work, is to choose training data that results in the GP specializing to the functions that represent the robot dynamics in the current operating condition.

B. Control Approach

The GP-basd model is used in the controller. We use the robust model predictive controller from [5] which solves an optimization problem at each time-step and computes an input sequence that minimizes a cost over the next few seconds given a GP-based model of system dynamics. We refer the reader to this paper for details.

C. Data Management

The purpose of our method is to construct the best possible model of the system dynamics for MPC. MPC uses the dynamics over the upcoming section of the path to compute the control input. We use data from the recently traversed part of the path to determine which past runs are relevant and can be used for the MPC prediction model. Referring to Fig. 2, for each previous run, $i = 1, ..., n-1$, we use data, $\mathcal{D}_i^-$, from the recently traversed section of the path to construct a local GP, $\hat{g}_i^-$. Each of these GPs is then used to generate

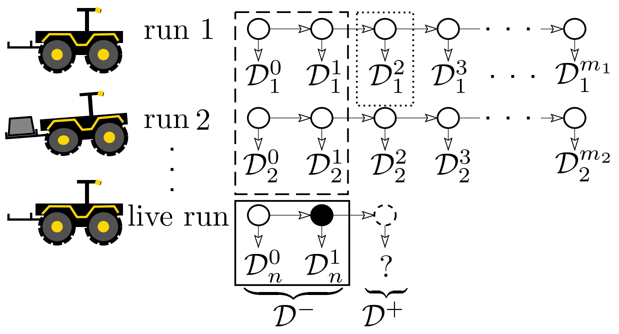

run 1
run 2
live run
$\mathcal{D}_1^0$ $\mathcal{D}_1^1$ $\mathcal{D}_1^2$ $\mathcal{D}_1^3$ $\mathcal{D}_1^{m_1}$
$\mathcal{D}_2^0$ $\mathcal{D}_2^1$ $\mathcal{D}_2^2$ $\mathcal{D}_2^3$ $\mathcal{D}_2^{m_2}$
$\mathcal{D}_n^0$ $\mathcal{D}_n^1$?
$\mathcal{D}^-$ $\mathcal{D}^+$

Fig. 2: Runs 1 and 2 represent previous autonomous traverses of the path and the live run represents the current traverse. The filled black circle indicates the current position of the vehicle and the dotted circle indicates the predicted position of the vehicle at the end of the MPC prediction horizon. Data, $\mathcal{D}_r^v$, for each run, r, is stored in small sets at each vertex, v, represented by a circle. The goal is to use recent data from the live run (solid box) to assess the similarity of the current dynamics to the dynamics in all previous runs along the same section of the path (dashed box). This is used to recommend experiences from a run with similar dynamics (dotted box) and construct a predictive model for the dynamics on the upcoming section of the path.

predictions for the mean and variance, $\{\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-\}_{j=1..m_n}$, at each of the m_n $\mathbf{a}_{n,j}$'s in recent experience from the current run, $\mathcal{D}_n^-$. These estimates are used to identify the most similar run to the current run and reject runs that are substantially different from the current run. Details are provided in the following subsections.

To update the control GP (see Fig. 1) with new points to model the dynamics on the upcoming section of the path, we randomly draw ten experiences (if available) from the most similar run in a window ahead of the vehicle overlapping with the MPC prediction horizon. These experiences are added to the set of experiences already in the control GP. Fifty experiences (if available) are then randomly chosen from this combined set to get a new set of experiences for the control GP [14]. If no runs are recommended, we randomly remove ten data points from the set in the GP used for control. If this happens several times in a row, the control GP will quickly revert to the prior which acts as a 'safe mode' since its uncertainty bounds are the most conservative. We only use experiences from the most similar run, however, we could easily make use of experiences from multiple runs if not enough experience was available from one run.

The control GP uses fifty points to allow the controller to run at 10 Hz with a 1.5 second look-ahead. Updating the GP incrementally helps the model change smoothly which in turn results in smoother control actions. We found this to be helpful during our experiments.

D. Run Rejection Criterion

Our first step is to eliminate runs where the dynamics are so different from the current dynamics that using data from these runs is likely to result in model errors that violate assumptions made by the safe controller. In particular, we must ensure that the 3σ bounds on the prediction from the GP are a reasonable upper bound on the model error [5].

To do this for candidate run i, we test whether the proportion of samples from $\mathcal{D}_n^-$ that lie further than $3\hat{\sigma}_{i,j}^-$ from the corresponding predicted $\hat{\mu}_{i,j}^-$ is significantly higher than would be expected by chance. We do this using the binomial test.

Let m_n be the number of input-output pairs in $\mathcal{D}_n^-$ and N_{out} be the number of outliers, points outside of the predicted 3σ bounds, according to the predictions from $\hat{g}_i^-$. The binomial distribution $B(x, m_n, p)$ describes the probability of drawing exactly x outliers from m_n independent samples where the probability of drawing an outlier is p. We calculate the probability of N_{out} or more outliers using

$$p(N_{\text{out}} \text{ or more}) = \sum_{x=N_{\text{out}}}^{m_n} B(x, m_n, p). \tag{7}$$

We reject the run if $p(N_{\text{out}} \text{ or more}) < \alpha$ where α is the significance level, or the probability of falsely rejecting a run. We chose a 5% significance level for our experiments. This may be reduced to avoid falsely rejecting runs, or increased to be more conservative. Since $3\hat{\sigma}_{i,j}^-$ is such a conservative bound, changing α only changes the allowed number of outliers by one or two for $m_n = 100$; that is, the algorithm is not very sensitive to this parameter.

Any runs that make it past this step are considered as candidates for drawing experiences to model the dynamics over the upcoming section of the path.

E. Run Similarity Measure

Our next step is to identify which runs are most similar to the current run given recent data from the live run, $\mathcal{D}_n^-$, and corresponding predictions, $\{\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-\}_{j=1..m_n}$, from the local GP for each candidate run, i.

We assume that the vehicle can be in a different operating condition for each run, and that one set of GP hyperparameters is sufficient to describe the robot dynamics in each operating condition when operating conditions are considered separately. We can then compute the posterior probability that the current dynamics are from the same operating condition, c, as run i using

$$p(c = i|\mathcal{D}_n^-, \hat{g}_i^-) \propto p(\mathcal{D}_n^-|\hat{g}_i^-)p(c = i). \tag{8}$$

The second term on the right is the prior, which we assume to be equal for all runs. The first term on the right is the probability of recent experiences if the operating conditions are the same as run i. Assuming each experience is independent, this is

$$p(\mathcal{D}_n^-|\hat{g}_i^-) = \prod_{j=1}^{m_n} p(\hat{g}(\mathbf{a}_{n,j})|\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-), \tag{9}$$

where $\hat{\mu}_{i,j}^-$ and $\hat{\sigma}_{i,j}^-$ are the predicted mean and variance of the GP for run i evaluated at point $\mathbf{a}_{n,j}$, which is in $\mathcal{D}_n^-$. Similar to our previous work [14], we reject any run that has lower probability than the GP prior of generating $\mathcal{D}_n^-$. This is to ensure that experience added to the GP for control is likely to improve the performance beyond what could be achieved with no experience at all.

In our implementation, we use the log-probability to avoid numerical issues and do not normalize the posterior since we are only interested in finding the most likely run and not the actual probability distribution. We denote the un-normalized log-probability for run i with L_i. The run with the largest L_i is chosen as the recommended run.

F. Overview of the Algorithm

Putting the components above together, we arrive at the experience recommendation algorithm Alg. 1.

Algorithm 1 Overview of the experience recommendation algorithm.

```
$n \leftarrow$ live run number
$\alpha \leftarrow$ significance level for binomial test
$\mathcal{D}^- \Leftarrow \left\{\mathcal{D}_i^-, i = 1..n-1\right\}$    $\triangleright$ data from candidate runs
$\hat{g}_i^- \leftarrow$ Fit a GP to each candidate run using data in $\mathcal{D}^-$
$scored\_runs = \{\}$
$\mathcal{A}_n^-, \mathcal{G}_n^- \leftarrow$ GP inputs from $\mathcal{D}_n^-$, GP outputs from $\mathcal{D}_n^-$
for $i = 1..n-1$ do
    $\left\{\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-\right\}_{j=1..m_n} \leftarrow \hat{g}_i^-(\mathcal{A}_n^-)$
    $N_{\text{out}} \leftarrow$ # outliers given $\left\{\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-\right\}_{j=1..m_n}, \mathcal{G}_n^-$
    $p_b \leftarrow$ Binomial test probability given $N_{\text{out}}$
    if $p_b < \alpha$ then
        continue
    end if
    Compute $L_i$ given $\left\{\hat{\mu}_{i,j}^-, \hat{\sigma}_{i,j}^-\right\}_{j=1..m_n}, \mathcal{G}_n^-$
    Append ($i$, $L_i$) to $scored\_runs$
end for
$\hat{\mathcal{D}}_n^+ \leftarrow$ experiences in the control GP
if $scored\_runs$ is empty then
    Remove 10 points from $\hat{\mathcal{D}}_n^+$
else
    $i^* \leftarrow$ run with largest $L_i$
    Randomly add 10 experiences from run $i^*$ to $\hat{\mathcal{D}}_n^+$
    Randomly remove 10 points from $\hat{\mathcal{D}}_n^+$
end if
Construct a new GP for control using $\hat{\mathcal{D}}_n^+$
```

V. Experiments

Experiments were conducted on a 900 kg Clearpath Grizzly skid-steer ground robot shown in Fig. 3. We tested our algorithm with the Grizzly in three configurations: *(i)* the *nominal* configuration with no changes to the vehicle; *(ii)* the *loaded* configuration with six bags of gravel, weighing approximately 30 kg each, in a cargo carrier mounted on the Grizzly (see Fig. 3); *(iii)* the *altered* configuration where the rotational rate commands were multiplied by 0.7. Compared to the *nominal* configuration, the *loaded* configuration results in oversteer and the *altered* configuration results in understeer.

Our first experiment was conducted in a parking lot on a 42 meter-long course. During this experiment, the configuration was switched between the *nominal* and *altered* configurations. We compare our proposed method to a baseline method, which uses only experiences from the most recent run. We conducted three runs in each configuration to allow the baseline method to converge to the dynamics in each configuration before switching. This serves as a simple example to demonstrate a few important features of our algorithm.

Our second experiment was also conducted in a parking lot on a similar course. However, we switched between all three configurations after only two runs in each over a total of 30 runs to compare our method to the baseline method during long-term operations. This was to demonstrate that the proposed method continues to learn during long-term operation and maintains high performance in all three configurations regardless of the order of configuration switches.

A. Implementation

Our algorithm was implemented in C++ and can process up to 300 runs in a single thread at 2 Hz on an Intel i7 2.70 GHz 8 core processor with 16 GB of RAM. This number is extrapolated based on the fact that the computational cost of the proposed method scales linearly with the number of runs. We consider the last three seconds of data (30 samples) from the live run for $\mathcal{D}_n^-$. The experience recommendation process runs in a separate thread from the controller. Therefore, it does not add any computation time to the control loop and can run at a different rate to process more runs. Our controller relies on a vision-based system, Visual Teach and Repeat [12], for localization.

B. System Model and Controller Parameters

Our process model is the unicycle model with an additive GP learning term [5],

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + dt \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k^{cmd} \\ \omega_k^{cmd} \end{bmatrix}}_{\text{unicycle model}} + dt \underbrace{\begin{bmatrix} g_x(\mathbf{a}_k) \\ g_y(\mathbf{a}_k) \\ g_\theta(\mathbf{a}_k) \end{bmatrix}}_{\text{learning term}}, \quad (10)$$

where v_k^{cmd} and ω_k^{cmd} are the commanded speed and turn rate, x and y are the position in the reference frame, θ_k is the orientation, and dt is the timestep of the controller. The cost function for MPC is a quadratic penalty on lateral error, heading error, ω_k^{cmd}, $(v_k^{cmd} - v_k^d)$ where v_k^d is the desired speed, $\dot{\omega}_k^{cmd}$, and $\dot{v}_k^{cmd}$. The respective weights are 500, 35, 5, 4, 1000, and 500. The desired speed was set at 1.5 m/s for all of our experiments.

C. Model Prediction Performance

In order to evaluate the quality of the models constructed using our learning method, we first compared the multi-step prediction performance of a GP constructed using our experience recommendation method to a GP constructed using experiences from the most recent run. We consider the rotational dynamics, because they differ the most between configurations.

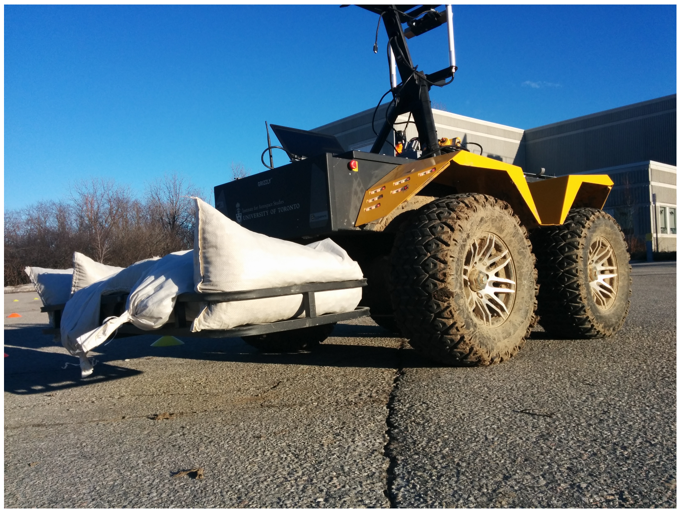


Fig. 3: Clearpath Grizzly in the *loaded* configuration with six bags of gravel in the cargo carrier. Each bag weighs approximately 30 kg for a total payload of 180 kg.

To measure the accuracy of the prediction of the mean, we use the Multi-Step RMS Error (M-RMSE) between the prediction made over the look-ahead horizon in MPC and the measured state at these times. To measure the accuracy of the error bound, we use the Multi-Step RMS Z-score (M-RMSZ) of the prediction at the future time-steps.

The M-RMSZ for a prediction of K time-steps is defined as

$$M - RMSZ_k = \sqrt{\frac{1}{K}\sum_{j=k+1}^{k+K}\frac{(\dot{\theta}_j^{true} - \mu_{\dot{\theta}}(\mathbf{a}_j))^2}{\sigma_{\dot{\theta}}(\mathbf{a}_j)^2}}, \quad (11)$$

where $\mu_{\dot{\theta}}(\mathbf{a}_j)$ and $\sigma_{\dot{\theta}}(\mathbf{a}_j)$ are the mean and standard deviation of the predicted rotational rate evaluated at the predicted GP input $\mathbf{a}_j$. The true measurements of angular velocity, $\dot{\theta}_j^{true}$, are used for comparison. An M-RMSZ of around one is ideal. A larger M-RMSZ around two indicates that the model is overconfident and M-RMSZ of less than one indicates that the model is conservative.

Figure 4 shows that the M-RMSE after a transition is up to 2.5 times higher when using experiences from the last run compared to the proposed method. The M-RMSZ shows a similar trend with the proposed method continually closer to one than the baseline method. We ignore run one for this comparison because the robot did not have any experience.

D. Closed-Loop Performance

To assess the impact of our method on closed-loop performance, we compare the speed, control cost, and tracking error using our method compared to when the GP is constructed using experiences from the last run.

Figure 5 shows that after transitions from the nominal configuration to the altered configuration, the proposed method significantly lowers the control cost compared to the baseline method. This cost comes from large lateral tracking errors due to understeering. We do not observe the same difference when transitioning to the nominal configuration because the

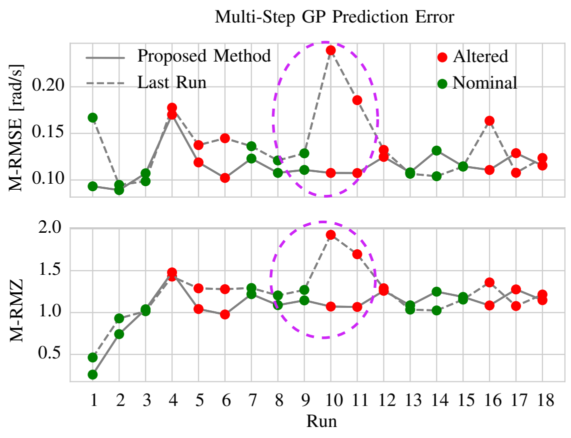


Fig. 4: This figure shows how the proposed method improves the GP multi-step prediction performance compared to using experiences from the last run, especially during runs 8-11 indicated by the purple dotted circle. The configuration is switched between the ***nominal*** configuration (green) and the ***altered*** configuration (red). The lower M-RMSZ and higher M-RMSE during run 16 for the Last Run method indicates that the controller exploited actions where the model was more uncertain for this run.

vehicle will tend to oversteer in this case, and the penalty function in the MPC penalizes the magnitude of the turn rate commands, which naturally prevents the vehicle from oversteering. Choosing a different control cost may result in an increased cost for switching the configuration either way.

E. Experience Recommendation by Configuration

Figure 6 shows that the proposed method prefers experiences from the same configuration as the vehicle's current configuration and relies heavily on experiences several runs in the past. This demonstrates that it is beneficial to store not only the most recent experiences, but also many experiences from the past in order to leverage experiences from the same configuration.

Some of the time, experiences are chosen from runs with a different configuration. Figure 7 shows that this is primarily along straight sections of the path where the vehicle is not turning and therefore the dynamics are similar. On all sections of the path with sharp corners where the dynamics are the most different between configurations, the proposed method prefers experiences from the same configuration.

The proportion of time that the proposed algorithm rejects all previous runs decreases rapidly over time. The highest proportion is during the first three runs (Nominal 1 in Fig. 6), where during the first run there are no previous runs to draw experiences from, and during the second and third run where the algorithm rejects all runs 17% and 11% of the time, respectively. After this initial phase of adaptation, the algorithm manages to find relevant experience over 89% of the time for each run.

F. Closed-Loop Performance, Long-Term Experiment

Our second experiment was to demonstrate the benefit of the proposed method during long-term operation with frequent changes in the operating conditions. The configuration

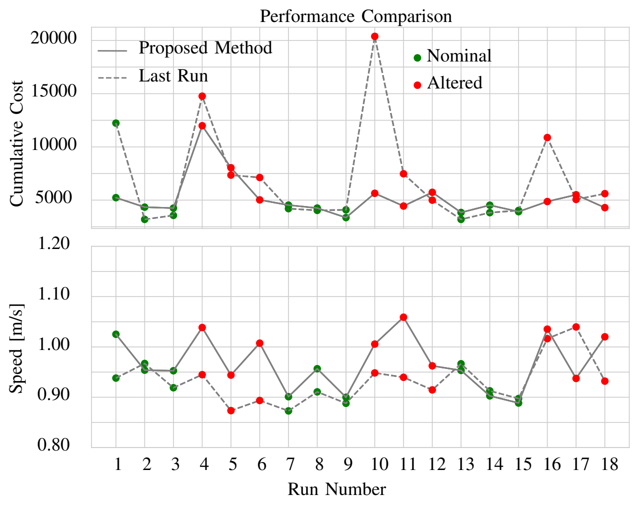
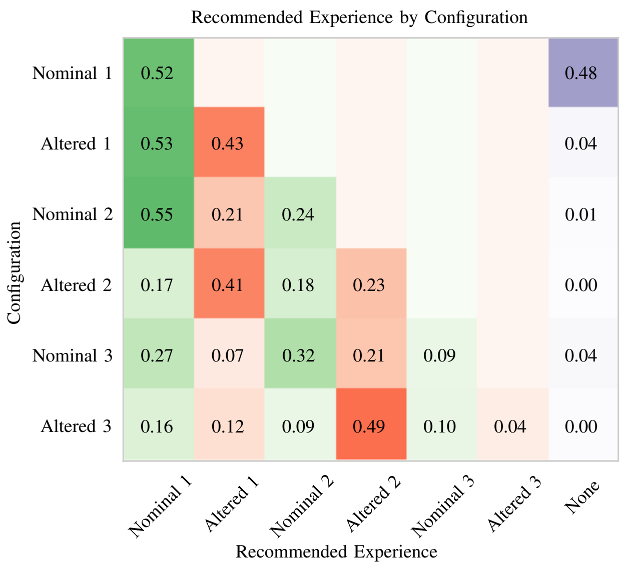


Fig. 5: The cumulative cost calculated using the MPC penalty function with the states and actions the system actually visited during a run, and the average speed during each run. Runs in the ***nominal*** configuration are green and runs in the ***altered*** configuration are red. The proposed method reduces the control cost compared to using experiences from the last run. The average speed when using the proposed method is slightly higher because the controller does not have to slow down to make large corrections as often.



Recommended Experience by Configuration
Configuration
Recommended Experience

Configuration	Nominal 1	Altered 1	Nominal 2	Altered 2	Nominal 3	Altered 3	None
Nominal 1	0.52						0.48
Altered 1	0.53	0.43					0.04
Nominal 2	0.55	0.21	0.24				0.01
Altered 2	0.17	0.41	0.18	0.23			0.00
Nominal 3	0.27	0.07	0.32	0.21	0.09		0.04
Altered 3	0.16	0.12	0.09	0.49	0.10	0.04	0.00

Fig. 6: This figure shows the distribution of recommended experiences by configuration when the vehicle was in each configuration. Each row corresponds to three runs conducted in the same configuration. Each column shows the proportion of experiences recommended from each previous set of runs. The color of the column indicates the vehicle configuration for those runs. Green is for the ***nominal*** configuration, red is for the ***altered*** configuration, and purple is for when ***none*** of the previous runs were recommended.

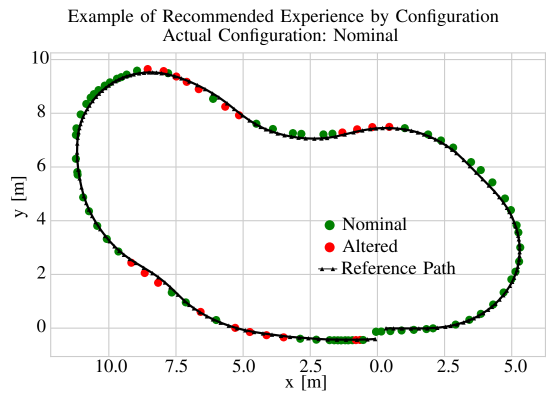


Fig. 7: A top-down view of the path showing the experiences recommended by configuration as the vehicle moves along the path while the vehicle is in the ***nominal*** configuration. Red and green markers indicate when experiences were recommended from the ***altered*** and ***nominal*** configurations, respectively. The reference path is shown in black. This shows that experiences are primarily recommended from a different configuration along straight sections of the path, which is when the dynamics in the two configurations are the most similar.

was switched every two runs giving the baseline method one run to adapt before changing the configuration again.

Figure 8 shows the cumulative control cost over thirty runs of a similar course to the first experiment. On average, the proposed method results in a 37% lower cumulative control cost compared to the baseline method primarily due to transitions to the *altered* configuration, which is the most distinct of the three.

In addition, the number of times the algorithm rejected all candidate runs decreased dramatically after the initial adaptation just like in the first experiment. For the first three pairs of runs, all runs were rejected 55%, 19% and 4% of the time, respectively. After this, the algorithm found matching experiences 97% of the time with the exception of the second pair of runs in the altered configuration, where it rejected all runs 9% of the time.

VI. Discussion

Choosing the best hyperparameters, kernel, and features is important for the GP to be a good model for the robot dynamics; however, this is beyond the scope of this paper. Our method can help a GP-based controller perform only as well as it would with training data from the correct operating condition. If the GP does not improve performance for a given operating condition when trained on the correct data, our method will not improve performance. We have seen evidence of this when driving in challenging off-road conditions. It is important to note that during these runs, the vehicle remained within path-tracking constraints; that is, model error did not jeopardize the safety of the vehicle, the GP just did not improve the performance.

In addition, our method requires that the dynamics over the previous section of the path are a good indicator of the

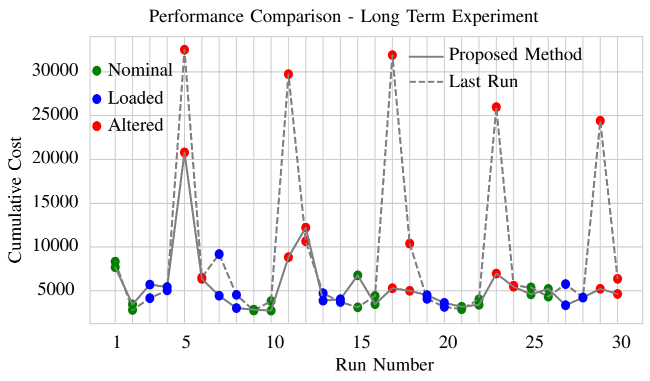


Fig. 8: This figure compares the sum of the control cost for the actual states/inputs over 30 runs in varying configurations when using the proposed method compared to the baseline method. Colored markers indicate the configuration for each run. The vehicle cycled between the ***nominal*** configuration (green), the ***loaded*** configuration (blue), and the ***altered*** configuration (red).

dynamics on the upcoming section of the path. The operating conditions are always sampled discretely (by run) but can be continuous. Although the method infers by run, which is discrete, it still works for continuous variables as long as they vary consistently between runs.

Our method also requires large changes in the dynamics to produce a noticeable improvement. The dynamics in the *loaded* and *nominal* configurations were quite similar; using data from one or the other did not produce noticeable differences in performance. If a new configuration with half the load was added, our method would automatically determine that it was similar to the loaded or nominal configurations and leverage data from runs in those configurations.

VII. Conclusion

In this paper, we presented a new, principled method for experience recommendation for long-term, GP-based, safe learning control. We demonstrated in closed-loop experiments how this method can be used to improve the performance of a controller conducting repeated traverses of a path when the dynamics switch between distinct configurations. This enables the controller to maintain high performance when revisiting operating conditions that have been seen before and safely learn new dynamics when new operating conditions are encountered.

References

[1] L. G. Dekker, J. A. Marshall, and J. Larsson. Industrial-Scale Autonomous Wheeled-Vehicle Path Following by Combining Iterative Learning Control with Feedback Linearization. In *In Proc. of the Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 2643–2648, Sept 2017.

[2] T. Moldovan, S. Levine, M. Jordan, and P. Abbeel. Optimism-Driven Exploration for Nonlinear Systems. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 3239–3246, 2015.

[3] F. Berkenkamp, A. P. Schoellig, and A. Krause. Safe Controller Optimization for Quadrotors with Gaussian Processes. In *Proc. of the Intl. Conference on Robotics and Automaiton (ICRA)*, pages 491–496, 2016.

[4] C. Ostafew, A. P. Schoellig, and T. Barfoot. Conservative to Confident: Treating Uncertainty Robustly Within Learning-Based Control. In *Proc of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 421–427, 2015.

[5] C. Ostafew, A. P. Schoellig, and T. Barfoot. Robust Constrained Learning-Based NMPC Enabling Reliable Mobile Robot Path Tracking. *Intl. Journal of Robotics Research (IJRR)*, 35(13):1547–1563, 2016.

[6] A. Aswani, H. Gonzalez, S. Sastry, and C. Tomlin. Provably Safe and Robust Learning-Based Model Predictive Control. *Automatica*, 49(5):1216–1226, 2013.

[7] C. Xie, S. Patil, T. Moldovan, S. Levine, and P. Abbeel. Model-Based Reinforcement Learning with Parametrized Physical Models and Optimism-Driven Exploration. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 504–511, 2016.

[8] 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)*, pages 1714–1721, 2017.

[9] B. Luders, I. Sugel, and J. How. Robust Trajectory Planning for Autonomous Parafoils Under Wind Uncertainty. In *Proc. of the AIAA Conference on Guidance, Navigation and Control*, 2013.

[10] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig. Deep Neural Networks for Improved, Impromptu Trajectory Tracking of Quadrotors. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 5183–5189, 2017.

[11] G. Aoude, B. Luders, J. Joseph, N. Roy, and J. How. Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[12] 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)*, 34(1):98–122, 2017.

[13] A. Angelova. *Visual Prediction of Rover Slip: Learning Algorithms and Field Experiments*. Phd thesis, California Institute of Technology, 2008.

[14] Christopher D. McKinnon and Angela P. Schoellig. Learning Multi-Modal Models for Robot Dynamics with a Mixture of Gaussian Process Experts. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2017.

[15] C. Ostafew, A. P. Schoellig, and T. Barfoot. Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 4029–4036, 2014.

[16] J. Gillula and C. Tomlin. Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning. In *Proc. of Robotics: Science and Systems (RSS)*, pages 81–88, 2012.

[17] P. Bouffard, A. Aswani, and C. Tomlin. Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, pages 279–284, 2012.

[18] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, K. Goldberg, and P. Abbeel. Learning Accurate Kinematic Control of Cable-Driven Surgical Robots using Data Cleaning and Gaussian Process Regression. In *Proc. of the Intl. Conf. on Automation Science and Engineering (CASE)*, pages 532–539, 2014.

[19] K. Jo, K. Chu, and M. Sunwoo. Interacting Multiple Model Filter-Based Sensor Fusion of GPS with In-Vehicle Sensors for Real-Time Vehicle Positioning. *Transactions on Intelligent Transportation Systems*, 13(1):329–343, 2012.

[20] R. Calandra, S. Ivaldi, M. Deisenroth, E. Rueckert, and J. Peters. Learning Inverse Dynamics Models with Contacts. In *Intl. Conf. on Robotics and Automation (ICRA)*, pages 3186–3191, 2015.

[21] R. Pautrat, K. Chatzilygeroudis, and J. Mouret. Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search. In *arXiv:1709.06919*, 2017.

[22] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Nonparametric Bayesian Learning of Switching Linear Dynamical Systems. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 457–464, 2009.

[23] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.