中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-22-001

Time-optimized velocity trajectory of a bounded-input double integrator with uncertainties: a solution based on PILCO

Hsuan-Cheng Liao, Wen-ChiehTung, Han-Jung Chou, Jing-Sin Liu

April 6, 2022 || Technical Report No. TR-IIS-22-001
https://www.iis.sinica.edu.tw/zh/page/library/TechReport/2022.html

Time-optimized velocity trajectory of a bounded-input double integrator with uncertainties: a solution based on PILCO

Hsuan-Cheng Liao, Wen-ChiehTung, Han-Jung Chou, Jing-Sin Liu

Institute of Information Science, Academia Sinica, Nangang, Taipei, Taiwan115, ROC
{brianhcliao,jie1202, r02221012}@gmail.com,
liu@iis.sinica.edu.tw

Abstract. Reinforcement learning(RL) is a promising framework for deeper investigation of robotics and control on account of challenges from uncertainties. In this paper, we document a simulation and experiment in applying an existing model-based RL framework, PILCO, to the problem of state-to-state time-optimal control with bounded input in the presence of uncertainties. In particular, Gaussian Process is employed to model dynamics, successfully reducing the effect of model biases. Evaluation of policy, which is implemented in Gaussian radial basis functions, is done through iterated prediction with Gaussian posteriors and deterministic approximate inference. Finally, analytic gradients are used for policy improvement. A simulation has shown a successful learning of a double integrator completing a rest-to-rest nearly time-optimal locomotion for a prespecified stopping distance along a linear track with uniform viscous friction. Time-optimality and data efficiency of the learning are demonstrated in the results. In addition, an experimental validation on an inexpensive robot car shows the generalization potential and consistency of the leveraging model-based RL to similar systems and similar tasks. Moreover, a rescaling transformation from the baseline learned triangle velocity profile to a set of safe trapezoid velocity profiles is presented, accommodating additional velocity limit.

Keywords: Model-Based Reinforcement Learning, Velocity Learning, Time-Optimal Trajectory, Data-driven Vehicle Control

1 Introduction

Optimal control based approaches are important for trajectory planning in a lot of robotics and autonomous driving applications, since the formulations are able to take into account more general constraints and objectives. To improve the working efficiency and productivity by completing tasks as fast as possible, especially for tasks involving repetitive state-to-state transfer trajectory execution, Time-Optimal Control Problems (TOCP) have been illustrated in a mass body of literature[1] related to robotics, autonomous driving or racing. The challenges of the problem lie in several aspects including complicated system dynamics, multi-dimensional state and control spaces, as well as various constraints on the robots and the environment and most importantly uncertainties. The derivation of optimal control and state trajectories are in general very computationally costly, so that efficiency and accuracy of computational methods

adopted to find the optimal state and control trajectories are usually a great concern, as most practical industrial or engineering systems are required to react to instantly changes of operating conditions and environments in a short time, or even in real time. A popular approach to obtain time-optimal motion for a system with known dynamic model and fixed boundary conditions under the safety and kino-dynamic constraints of a vehicle is via optimal control or model predictive controlformulations. This approach requires the derivation of optimality conditions, which is composed of coupled ordinary differential equations or partial differential equations with boundary conditions, and the help of numerical methods and optimization algorithms.

Learning-based control algorithms for a dynamical system, on the other hand, learn to generate the desired system behavior without a priori complicated system formalism and predefined controller parameters, thereby being likely more generalizable and platform-independent. One of the promising approaches in the context of intelligent planning and control is based on Reinforcement Learning (RL) [2], [3], which can be viewed as a class of optimal control. RL refers to the learning of a policy, defined as a mapping from state space to action space, by means of maximizing a reward the agent receives from the environments it interacts. There are several ways to categorize RL algorithms, such as either model-based or model-free. With the dynamical system modelled as a reward-maximizing RL agent and the desired behavior expressed as a utility function, it is possible to train the system for an optimal sequence of actions through its interactions with the environment. To stay data-efficient, the group of model-based methods that employ derived or learned system dynamic models are more preferable for diverse robot applications since fewer interactions between the agent and the environment are required to find better trajectories faster, as compared to the group of model-free approaches [2],[3]. In general, the intricate transition models can be derived from deterministic physics-based models or stochastic learning algorithms, and the constraints can then be incorporated within the interaction processes to enforce the system with desirable behaviors.

The effectiveness and performance of RL is task instance-specific, i.e. as a function of the transition and reward functions induced by the policy being evaluated. Therefore, to address the practical challenges in facilitating RL algorithms for a wide range of real-world decision-making problems such as the autonomous vehicles for diverse driving scenarios, it is generally believed that only specific applications of RL on concrete cases can better demonstrate related issues. With this aim of study, the velocity learning task of an autonomous car driving with bounded acceleration along a linear path is investigated empirically. The learning goal is to recover a time-optimal motion to complete a rest-to-rest linear locomotion along a linear track. The dynamics to be learned is assumed to obey a double integrator-like uncertain dynamics subject to symmetric input constraint with parametric uncertain parameters of mass and friction. The characteristics of the task is that the vehicle mass is uncertain and the environment characteristics such as friction is unknown, and both parameters affect the maximum speed along the path.

Our main contributions include:

1. In this paper, we perform both simulation and sim-to-real validation experiment results with an existing sample-efficient model-based policy search algorithm,

Probabilistic Inference for Learning Control (PILCO) [4]. We apply PILCO to time-optimal velocity planning problem of state-to-state steer task in simulation and a real-world sim-to-real experiment. It is empirically shown in simulation that the car, modeled as a constrained double-integrator with uncertain mass and viscous friction, successfully accomplishes the learning of near time-optimal triangle velocity profile with a single switching, while keeping the advantage of data efficiency.
2.Provide assessment and interpretations on the model-based RL planning results
3.Transform the learned triangle velocity profiles in accordance with additional speed constraints into single or double trapezoid velocity profiles.

The remainder of the article is structured as follows: In section 2 we give an overview of the approaches to the time-optimal control problem. The key elements of the PILCO algorithm are elaborated in Section 3, including dynamics modelling, trajectory prediction, policy evaluation and policy improvement. In Section4, we present the simulation results of time-optimal velocity learning for the autonomous vehicle with double integrator dynamics, and sim-to-real experimental validation on a simple car along with discussions. Transforming a triangle velocity to a trapezoid or double trapezoids velocity for longer traveling time is shown in Section 5. Lastly, we conclude with final remarks in Section 6.

2 Related Work

There is a wealth of literature with regard to different aspects of TOCP. In this section, we briefly present the common approaches developed in robotics and autonomous vehicles, followed by those based on reinforcement learning.

2.1 Approaches to TOCP with Known dynamics

Control and trajectory generation and their optimization for the mission to be completed are typically based on a model of the controlled system, thus platform-specific. Regarding time-optimality, there are two variations of TOCP, namely complete and decoupled. Complete approaches aim to solve the challenging problem in its entirety of determining the optimal state and input trajectories simultaneously, in which a number of direct or indirect numerical methods are developed [6], [12], [25]. On the contrary, the decoupled approach, or path-velocity decomposition approach, decomposes the trajectory generation and optimization into two subproblems: the first is the generation of a geometric path for connecting two configurations without violating geometric constraints such as obstacle avoidance or smoothness requirements, and the second is the design of a time-scaling function of state-to-state transfer along the planned path, while respecting the given kino-dynamic constraints and fixed boundary conditions (such as the initial and target positions and velocities are precisely specified). Given a path, the velocity and acceleration of the vehicle on it can be altered by the design of the time-scaling function compliance with the constraints including torque, acceleration, velocity [7-10]. The following three conventional methods have been commonly used for motion planning and optimization of a broad variety of robotic systems such as robotic manipulators or mobile robots in the robotic community.

Hamilton-Jacobi-Bellman (HJB) Equation. To address the optimal control problems in full generality with the use of optimal control theory such as Pontryagin Maximum Principle (PMP) or Dynamic Programming Principle (DPP)[11], necessary optimality conditions for the state trajectories and control policies are derived. This yields the Two-point Boundary Value Problem of HJB partial differential equations for time-optimization of trajectories. HJB approach is a practically useful approach in that many numerical solvers of HJB equations are developed. The advantage of generality is that more general state and input constraints and objective functions can be taken into account. For example, time-optimality can be traded off against energy to yield less aggressive control to move the robots slower but smoother.

Convex Optimization (CO). The Hamiltonian of TOCP for robotic manipulator is shown to be convex with respect to the control input. TOCP is transformed into a convex optimization problem with a single state through a nonlinear change of variables [13], where the acceleration and velocity at discretized locations on the path are the optimization variables. . Then, followed by [14], the work is further extended to meet speed dependent requirements. Such approach is simple and robust thanks to the existing convex optimization libraries, yet only convex objective functions can be concerned. However, the convex optimization program contains a large number of variables and inequality constraints, making it slow and less suitable for real-time applications.

Numerical Integration (NI). Since the vehicle velocity highly depends on the path to be followed, the decoupled approach splits the motion planning problems to two subproblems as aforementioned to manage the computational complexity. A kinematically feasible state-to-state transfer path parametrized by a scalar curvilinear abscissa coordinate s, usually the arc length, is first determined. The travel time is determined by the path velocity $\dot{s}$ along the path or the time scaling function s(t) that is solved by optimization tools to meet the imposed constraints. Then dynamics expressed as multi-dimensional state space equations is reduced to expressions in phase plane of $(s, \dot{s})$of parametrized path coordinate s, and parametric path velocity $\dot{s}$. By description of the dynamics and constraints along the path to be followed on the $(s, \dot{s})$phase plane, then this method generates the velocity limit curve on the phase plane from the velocity and acceleration bounds. The generation of minimum-time velocity profile along the given pathis greatly simplified to the determination of switching structure in the phase plane [1], [11]. Essentially, itsearches for switching points on the phase plane and establishes the velocity profile by integrating forward with the acceleration limits and backward with the deceleration limits pivoting from those points. In general, a model predictive control (MPC) framework can be used in the decoupled approach to generate the safe velocity profileand the input commands for following a given planned path in terms of known system dynamics [15].

2.2 Reinforcement learning frameworks

For an autonomous robot or vehicle to perform a mission, the system is composed of the agent and the real-world environment where it operates. The mission may very often involve some planning task with time and computational resource budget in real-time applications. A model therefore is a good informative basis for planning. However,

the model of the agent and the environment is uncertain and not accurate due to parametric uncertainties and unmodelled dynamics. Sources of uncertainties exist in the input and perception and agent-environment interaction. Dealing with uncertainties is one of the most challenging issues in planning and control of unmanned vehicles. The control should thus be robust to real situations encountered during execution of mission for safety and performance guarantee. For enhancing autonomous robotic system performance, instead of tuning the behaviors frequently and manually in practical situations, we may consider the integration of learning and planning due to the capabilities of tackling the issue of over-simplicity of modeling development due to unmodeled dynamics and uncertainties. This aim is likely to be alleviated with the advents of RL frameworks, especially the model-based family [27]. These approaches reformulate the problem as a Markov Decision Process for the autonomous agent, which maximizes the long-term rewards and does not necessarily need the transition dynamics beforehand. While the model-free group of approaches attracts the most scientific interest, the model-based group is recommended for real robots [2], [3], [28] since the employment of a learned model of the agent-environment interactions creates an internal simulation during learning process and reduces physical engagement substantially, decreasing potential hazards and mechanical wear of robots, while improving reliability and robustness in task execution. The suitability is, in addition to data efficiency and the advantage of less agent-environment interactions, because model-based RL learns a dynamics model and then the derived characteristics of a learned model is used for generating trajectories and learning the policy.

There are a surging number of researches related to model-based reinforcement learning over the past decade. Despite its faster convergence over the model-free frameworks, a severe issue is that any model bias greatly affects the learning performance. System transition modellingor model fitting techniques range from deterministic methods such as physics-based formulation, Receptive Field Weighted Regression(RFWR), to stochastic methods including Expectation Maximization (EM), Deep Neural Networks (DNN) and so on. Among which, Gaussian Process (GP) is the state-of-the-art practice that extracts the information from the sampled data with the highest data efficiency [4]. In contrast to other probabilistic models that maintain a distribution over random variables, GP builds one over functions. Therefore, it has no prior assumption on the function mapping current states and actions to future states. The fact makes it an effective tool, and is employed in this study.

RL for controlling non-linear dynamical models with continuous state and action space has two method families: value function or policy search.

Value Function. The most widely used one is the value function approaches. This class of methods estimates an optimal value function of future outcome $Q^{\pi}(s,a)$, usually written in the form of equation (1), given the state *s*of the agent and takes action*a*according to a policy $\pi(s,a)$ that results in a sequence of state and action.The policy that maximizes the long-term reward is consequentlyreconstructed from the sequence of optimal actions at each state

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma Q^{\pi}(s',a')] \quad (1)$$

This category is widely adopted in model-based RL and particularly implemented with dynamic programming techniques, which requires known system dynamics. However, it is not suitable for high dimensional or continuous state and action spaces as the scale of the value function will be infinite.

Policy Search. Policy search is deemed more natural for robotics learning [41]. Instead of deriving an optimal policy from the value function, this class of methods keeps a set of parameters for the policy and directly optimizes it by maximizing the cumulative reward. Two common techniques involved in policy search are gradient-based approaches and sampling-based approaches. We have chosen the former because the latter does not always guarantee a convergence to the optimal policy and the generated trajectory is sometimes far from the observed input-output data collected due to model biases [2]. More details will be given in the next section.

3 Model-based RL for Time-Optimal Vehicle Motion

The aim is to control the vehicle from an initial state to reach a target state in minimal time. The existence of time-optimal trajectories is guaranteed by PMP for a vehicle [16]. In order to find the time-optimal control policy, there are a number of choices of learning algorithms. An earlier work [17] used Q-learning for car-like vehicle motion planning. [18] considered the transfer learning of obstacle avoidance behaviors in similar environments with similar obstacle patterns, where the state of the environment is represented by the obstacle pattern. A recent model-free actor-critic RL algorithm is applied to time-optimal velocity planning along arbitrary path in [19].This work showed that incorporation of velocity computation by exploiting the vehicle dynamics model is feasible in practice to enhance the learning outcome to some extent.

The goal of data-driven velocity planning in the form of time scaling function we deal with is to recover or approximate the optimal time-scaling function for a vehicle without knowing its dynamic model. In this paper, for the specific aggressive velocity planning problem, we exploit and implement PILCO [4], a data- efficient model-based RL, first the simulation and then on a real experiment of toy car. PILCO is summarized in Algorithm 1. PILCO has been tested with success in benchmark tasks at low-dimensional state space such as control of the inverted pendulum and the cart-pole swing-up, demonstrating unprecedented performance of successful learning outcome. The method employs the state-of the art non-parametric Gaussian Process (GP) for model learning of the unknown dynamics and corresponding uncertainty estimates based on probabilistic dynamics modelling. It then uses approximate inference for system state trajectory predictions and policy evaluation. Finally, policy improvement is made with analytic policy gradients. The core elements of the PILCO framework, including dynamics modelling, trajectory prediction, policy evaluation and policy optimization, are briefly described in this section.

Algorithm 1 PILCO

1: *Define* parametrized policy:$\pi: z_t \times \theta \rightarrow u_t$

2: *Initialize* parameters θ randomly

3: *Execute* system and record data
4: **repeat**
5: *Learn* system dynamics model with GP
6:*Predict* system trajectories
7:*Evaluate* policy: $J^{\pi}(\theta) = \sum_{t=0}^{T} \gamma^{t} \mathbb{E}_{X}[cost(X_t)|\theta]$
8: *Update* parameters by gradients $\mathrm{d}J^{\pi}(\theta)/\mathrm{d}\theta$
9: *Execute* system and record data
10: **until** task completed

3.1 Dynamics modelling

In real world, model uncertainties and model errors are inevitable in the process of modeling a dynamic system. As mentioned, there are various methods to model and learn the unknown system dynamics [2]. It is important that the model learning algorithm can cope with the uncertainty and noise in collected data. PILCO adopts GP probabilistic modelling and inference to learn the transition dynamics of real-world agent (Algorithm 1, line 5) as a prediction model of the system to control. Therefore, it effectively handles the input uncertainties and reduces the effect of model errors or simplification for the system dynamics derived through non-trivial mathematics and physics equations, eliminating the common drawback of model-based frameworks.
The training inputs are data in the form of the state-action (*x,u*) pairs generated by the dynamics of agent-environment interaction:

$$\tilde{x}_t = \begin{bmatrix} x_t \\ u_t \end{bmatrix} \in \mathbb{R}^{D+F}, \tag{2}$$

and the target is the difference between consecutive states:

$$\Delta_t = x_{t+1} - x_t \in \mathbb{R}^{D}. \tag{3}$$

In this paper, a common choice of the mean and variance of a multivariate Gaussian is a zero mean function and a squared exponential covariance function defined as:

$$k(\tilde{x}_i, \tilde{x}_j) = \sigma_f^2 \exp(-\tfrac{1}{2}(\tilde{x}_i - \tilde{x}_j)^{\mathrm{T}} \Lambda^{-1} (\tilde{x}_i - \tilde{x}_j)), \tag{4}$$

where variance of the function σ_f^2 and $\Lambda \coloneqq \mathrm{diag}([l_1^2, l_2^2, \dots, l_{D+F}^2])$ depends on the length scales. With n training samples $\tilde{X} := [\tilde{x}_1, \dots, \tilde{x}_n]$ and y: $= [\Delta_1, \dots, \Delta_n]$, the posterior GP hyper-parameters are learned through evidence maximization and describes a one-step prediction model of state trajectory generation:
posterior state distribution

$$p(X_{t+1}|X_t, U_t) = \mathcal{N}(X_{t+1}|\mu_{t+1}, \sum_{t+1}), \tag{5}$$

mean

$$\mu_{t+1} = X_t + \mathbb{E}_f[\Delta_t], \tag{6}$$

variance

$$\sum_{t+1} = \mathrm{var}_f[\Delta_t], \tag{7}$$

where capitals represent random variables. In practice, the computationally tractable mean and variance are used for further decision making.

3.2 Deterministic Trajectory Prediction

With the model uncertainty handled by GP, PILCO employs model-based policy search for planning and uses analytic gradients of closed form solutions for optimization. For the later policy evaluation, PILCO first predicts long-term system trajectories with the learnt transition dynamics (Algorithm 1, line 6).To such end, the one-step prediction process is cascaded from $X_0$to X_1,X_1 to X_2, and up to X_T, generating a T-step-ahead prediction which forms a distribution over the system trajectories in which the uncertainties are handled by introducing the noise which contaminates the states. The distribution of state X_t at time t is assumed Gaussian with mean μ_t and variance Σ_t , $p(X_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$, and subsequently approximated by moment matching or linearization of the posterior mean and covariance for further computation during policy optimization.

$$\mu_{t+1} = \mu_t + \mu_\Delta, \tag{8}$$

$$\sum_{t+1} = \Sigma_t + \Sigma_\Delta + cov[X_t, \Delta_{t+1}] + cov[\Delta_{t+1}, X_t], \tag{9}$$

$$cov[X_t, \Delta_{t+1}] = cov[X_t, U_t]\Sigma_u^{-1} cov[U_t, \Delta_{t+1}]. \tag{10}$$

3.3 Policy evaluation

Having retrieved the predictive trajectories, it remains computing the expected long-term cost for policy evaluation. Given the entire experience time in each episode as a fixed length of horizon$\mathrm{T_{learn}}$, let $\Delta\mathrm{T} = \frac{\mathrm{T_{learn}}}{N}$be the sampling time so that [0,$\mathrm{T_{learn}}$] is uniformly sampled into equally-spaced N time steps $\{t\Delta\mathrm{T}, t = 0,1, \ldots, \mathrm{N}\}$. PILCO applies the discounted cost function (11) along the learned state trajectory $\{X_t, t = 0,1, \ldots, \mathrm{N}, X_0 \text{ given}\}$ for the sequence of actions generated by the control policy $\{a_t, t = 0,1, \ldots, \mathrm{N}-1\}$to state distributions at each time step

$$J(\theta) = \sum_{t=0}^{\mathrm{N}} \gamma^t \mathbb{E}_X[\mathrm{cost}(X_t)|\theta] \tag{11}$$

where γ is the future discount factor,

$$\mathbb{E}_X[\mathrm{cost}(X_t)|\theta] = \int \mathrm{cost}(X_t)\, \mathcal{N}(X_t|\mu_t, \sum_t)\mathrm{d}X_t, \tag{12}$$

and θ is the given policy parameter for the control policy. The reinforcement learning strategy is to iteratively refine a simulation-learned control policy that minimizes the fiexed learning horizon cumulative discounted cost (11)-(12) directly through the learned model as the model of the system is updated during learning episode when more information is acquired over eipsodes. In general, there are quite a number of cost functions possible for the reward of RL, yet the effectiveness depends on the applications indeed. For example, a parametric cost function [20] can be used to switch the cost from a quadratic cost function to a time-optimal cost during learning to come up with different feedback control in response to different events. Likewise, [21] utilizes a progress maximizing cost function, ensuring the learning agent drives as far as possible within every time step. Here, the cost function such that the integral in (12) can be analytically

calculated based on the trajectory (from the given initial state to the final state the system actually reaches) is suitable to avoid the pitfall.Note that a large γ will cause the accumulated cost (11) calculated at the end of episode to reduce more slowly in late time as time index t→ N. A small γ would thus be good for uniform convergence.

Remark. Since the learning reward (11)-(12) does not include the control regularization term (such as using L1, L2 or mixed L1- L2 norm regularization as additional sparsity-inducing cost for (11)-(12) [30]), it allows gradient computation for model-based optimization described in the following subsection.

3.4 Policy optimization

The policy is improved episodically through the gradient information of (11) (Algorithm 1, line 8): the update steps are in the gradient directions toward high rewards. For analytical tractability, it is required that the expected cost in (12) is differentiable about the state distribution moments, and that the moments of the controller distribution are differentiable with respect to the policy parameter θ. PILCO, a model-based RL, explicitly takes into account the model uncertainty for prediction and analytic policy gradient computation, which is available since the reward function and transition function are differentiable. Thorough analytic computation of the gradient $\mathrm{d}J^{\pi}(\theta)/\mathrm{d}\theta$, which involves several applications of chain rule, is documented in [4]. Finally, An advantage of PILCO is that thanks to the analytic expression of the policy gradient with respect to policy parameters, any standard gradient-based optimization method such as CG or BFGS can be implemented to search for the optimal policy parameter θ with thousands of parameters which minimizes the total cost $J^{\pi}(\theta)$ so as to obtain desired state trajectory.

4 Results and Discussions

The effectiveness and performance of a model-based reinforcement learning algorithm can be scenario (problem instance) dependent and platform-specific (in terms of dynamics model and parameters). The task demonstrated in this paper is to learn a velocity function for a vehicle with second order system dynamics so that the completion time of a rest-to-rest steer from origin to a state $\mathbf{x}_{\text{target}}$ along a linear path with unknown friction characteristics is as short as possible under the physical constraint of vehicle. The existence of a time-optimal velocity and input so that the target is achieved as fast as possible is guaranteed by PMP. The analytical or learned time-optimal velocity solution is both platform and path dependent. Control bounds, different boundary conditions or paths with different curvature profiles and lengths, and various choices of the physical parameters such as mass, friction, can produce different velocity solutions. As a result, different maximum velocities and different travel times are produced. A lower bound of the travel time can be calculated considering the path is a straight line as studied here.

For this task, both the state and the input can be observed to collect the data. Since the measured data x, $\dot{x}$ and the associated trajectory cost for a given policy have errors due

to model uncertainties and disturbances, model error is detrimental to task performance executed by a control policy [26]. Disturbances affect the stability of system to reach the target. Uncertainties, caused by lateral drift of the vehicle, physical properties such as inertia and friction etc., make the prediction of the sampled states on the linear track under the input ambiguous. The learning control algorithm should effectively mitigate the uncertainty propagation to accurately estimate the uncertainty of the dynamics when restricted for the specific state-to-state transfer task with given goal-reaching path to yield an accurate state-to-state time-optimal velocity solution along the linear track. In the following, to apply PILCO to time-optimal state-to-state transfer task along a linear path, we describe the test setup, simulation and experiment results, followed by some discussions.

The simulation consists in executing different control policies on the double integrator from the same rest state to reach a target state, and records the resulting state input pair and the cost at sampling times. We formulize the stage cost in (11) as related to an exponential function of the Euclidean distance to the target state $\mathbf{x}_{\text{target}}$ from the current state $X_t = \mathbf{x}(\text{t})$ at time t and tune the cost width with σ_c as shown in (13).

$$\text{cost}(X_t) = 1 - \exp\left(-\frac{1}{2\sigma_c^2}\left\|X_t - \mathbf{x}_{\text{target}}\right\|^2\right) \in [0,1] \tag{13}$$

This task-specific cost function measures how fast the vehicle progresses on the track to reach the target (subject to tolerance) within the learning horizon in each simulated episode. It has a shape of quadratic functions around the target state yet smoothens out at unity in distant states, and it is the only feedback information the vehicle receives during the vehicle-environment interaction. The parametrized policy is defined with a full state (position and velocity) feedback in the form of Gaussian Radial Basis Function (RBF) controller [4] as

$$\text{u}(\mathbf{x}, \theta) = \sum_{i=1}^{n_b} w_i \varphi_i(\mathbf{x}) \tag{14}$$

with

$$\varphi_i(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Delta^{-1}(\mathbf{x} - \mu_i), \tag{15}$$

where n_b is the total number of basis functions, and $\theta = [w_i, \mu_i, \Delta]$ represents the policy parameter vector of the weight, mean, and covariance of each Gaussian RBF. The choice of parametric controller (14)-(15) in the form of linear combination of Gaussian RBFs lies in its function approximation property and good generalization property, where the controller parameter values and the number of basis functions can be optimized using various methods. This implies robustness to vehicle parametric uncertainties and disturbances in learning. Since the optimal control is generally unknown, $n_b = 100$ is chosen to be sufficiently large, thus $\theta \in R^{403}$, to allow an accurate approximation.

4.1 Simulation

Setup. We take 0 as the initial time and rest state (0, 0) as the initial state. The final state to be reached is assumed to be a known target position at a distance L at rest. The environment is visualized in Fig. 1, where the autonomous car is drawn as the black box. The vehicle has a horizontal length of 30 centimeters and a mass of 0.5kg. In addition, in order to mimic a real vehicle on the road, a friction coefficient of $\mu = 0.1$is assumed between the ground and the car, and there is a symmetric bound of$\pm 4\text{m/s}^2$on

the acceleration control. It can move forward and backward. The simulation is on MATLAB with Intel i7 core and 16 GB RAM. For our simulation purpose, we use the double integrator to simulate behavior of real vehicle traveling along a straight line on the flat ground with viscous friction. Essentially for a point moving on a planar path, it can be considered as a double integrator or a canonical spring-damper-mass system, therefore a one-dimensional motion planning problem along a curved path. The time-optimal control for double integrator is solved in [25] as the trajectory optimization via barrier function. Let x(t) denote the distance traveled by a point mass m on a frictional ground controlled by an applied throttle(proportional to forward acceleration) u with symmetric constraint $u \in U_{ad} = [-u_{max}, u_{max}]$ where $U_{ad}$is an admissible control set (a convex polytope). We assume that u_{max} is not less than the static friction force, whereby the initial velocity is zero. To avoid slipping, the acceleration bound should satisfy $u_{max} \leq \mu g$, g=9.8m/s^2 is the gravitational acceleration. Defining $\mathbf{x}=[x, \dot{x}]^T$ as the state vector of simulated vehicle, the state equations for the vehicle dynamics can be written as

$$\dot{\mathbf{x}} = \mathbf{A}x + \mathbf{b}u, \mathbf{x}(0) = 0 \tag{16}$$

where$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{c}{m} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$,m is the vehicle mass, $c \geq 0$ is the viscous friction coefficient, $u = u(\mathbf{x}(t), \theta(t))$.

The action space for the vehicle is the convex and compact set U_{ad}. Since the vehicle (16) is controllable, the existence of a control input with at most one switching to steer the vehicle from a rest state $\mathbf{x}(0) = 0$ at time zero to a neighborhood of another state $\mathbf{x}(t)$ in a given time t is ensured by controllability; in our case, to a target state $\mathbf{x}_{target} = [5\ 0]^T$in minimum time T is ensured by PMP. Under nominal operating conditions with no constraints imposed on the motion profiles and no disturbance, the state response of (16) for an arbitrary piecewise continuous input function u(t) is precisely:

$\mathbf{x}(t)=\int_0^t e^{\mathbf{A}\tau}\mathbf{b}u(t-\tau)d\tau$
$=\mathbf{C}\mathbf{z}(t)$ (17)
where

$$e^{\mathbf{A}t} = \alpha_0(t)I + \alpha_1(t)\mathbf{A},$$

$\mathbf{z}(t) =[z_1 z_2]^T$, $z_i(t)=\int_0^t \alpha_i(t)u(t-\tau)d\tau$, i=1,2

$\mathbf{C}$= [**b Ab**]=controllability matrix

There exist a lot of candidate control algorithms that realize the required state-to-state steering task under nominal and perturbed operating conditions. Each requires the control algorithm to tune the corresponding controller parameters according to different system parameters or operating conditions. In this study, the learning-based control policy tries to find the policy that decreases the accumulated expected cost (13) of pairwise distance between the vehicle achieved progress at each sampling time and the target. Here an episode is considered where the vehicle is returned to the same initial state after executing a policy. The motion starts from rest at the origin, moves along a linear track for a fixed duration $T_{learn} = 4$ by applying a policy. Each episode lasts a fixed duration T_{learn} via repeated traversal along the linear track from the same initial

state while reaching finally a position by applying a policy, as shown in Fig. 1. It is desired that $x(T) = L$, $\dot{x}(T) = 0$ for a prescribed distance L=5 with $T \leq T_{learn}$ as small as possible, i.e. the policy determines the maximum $\dot{x}$ for each point x on the prespecified path and a T for task completion. The choice of the learning time $T_{learn} = 4$ in one episode for the simulations reported here is a tradeoff of the amount of data collected (computational load) and performance, where too small/largeT_{learn} results in aggressive/relaxed learning.

Results.

Dataset preparation. We use trajectory planning for data collection over $[0, T_{learn} = 4]$. Applying a given policy in the form of () to drive on the linear track from the rest produces specific sequence of states (positions and velocities). The collected dataset for learning the dynamics model holds only for the given path and task. In running each episode, the vehicle starts from the same initial state (zero state at rest) at t=0. We collect in one control cycle for each maneuver policy a batch of trajectory data recorded for vehicle motion based on simple point mass vehicle model given by (16) respecting the acceleration limits at sampling times $0.1, 0.2, \cdots, 3.9, 4$ over the time horizon $[0,4]$. Therefore, a total of 16 episodes (including an initial random-policy round) are executed, each of which has a total of N=40 samples of state-control pairs. Suppose we have a sequence of 40 admissible control inputs satisfying the control bound U_{ad} at the jth learning episode

$$u_1^{(j)} = \pi(\mathbf{x}_1^{(j-1)}, \theta^{(j-1)}), \cdots, u_{40}^{(j)} = \pi(\mathbf{x}_{40}^{(j-1)}, \theta^{(j-1)}) \quad (18)$$

where $u_0^{(j)}$ is an initial input for arbitrary initial exploration. The constant size dataset D^i is composed of a sequence of precisely 40 state-action pairs with the corresponding state being visited by the vehicle and the corresponding control input

$$D^j = \left(\left(\mathbf{x}_1^{(j)}, u_1^{(j)}\right), \cdots, \left(\mathbf{x}_{N_{target}}^{(j)}, u_{N_{target}}^{(j)}\right), \cdots, \left(\mathbf{x}_{40}^{(j)}, u_{40}^{(j)}\right)\right), \quad (19)$$

where $\mathbf{x}_0^{(j)} = \mathbf{x}_0$. Here $N_{target} \leq N = 40$ denote the first sampling time at which the car passes through the target region. The state transition rule of (16) that maps current state and action to the next state

$$\mathbf{x}_{k+1}^{(j)} = \mathbf{x}^{(j)}(0.1(k+1)) = \mathbf{x}^{(j)}(0.1k) + 0.1\mathbf{C}\mathbf{z}^{(j)}(0.1k) = \mathbf{x}_k^{(j)} + 0.1\,\mathbf{C}\mathbf{z}_k^{(j)},\ k=0, \cdots, 39,\ (\mathbf{x}_0^{(j)} = \mathbf{x}_0\ , u_0^{(j)}) \text{ given},$$

defines the set of reachable state from a given initial state. These 40 state-control pairs collected at different sampling times are correlated via state transition rules. The j-th episode learning data are to be used as demonstration via the reward in its next (j+1)-th learning cycle so that the policy is revised with online estimation of vehicle model

based on the batch of collected data (for the whole episode completed) during the learning process.

Performance. The learning performance was evaluated by computing the trajectory cost function (11)-(13) over the entirety of trial state and control trajectories in each trial. The dataset D^i of 40 data in each trial records how the vehicle modifies its input at each sampled point of the path in accordance with the trajectory cost function as the feedback the vehicle receives. The outcome of learning shown in Fig. 2 is encouraging. Given the trajectory cost function, though not monotonic over episodes, the state and control trajectories are improved by PILCO, which learns to utilize more force to reach the maximum velocity faster and the target faster as the model is updated when more information is acquired. The learned control policy exhibits desirable property of minimizing the hitting time N_{target}: the car drives forwards at the maximum acceleration from the start at rest to reach a suitable maximum speed depending on the boundary conditions of the path and knows when to slow down with the maximum deceleration for speed reduction to reach the target at rest. The input is always on values in proximity to the boundary of admissible control set U_{ad} , i.e. bang-bang control. Though the maneuver time minimization is not directly involved in the cost (13), by interacting with the environment during each episode (Algorithm 1, line 3/9), the agent in every possible state performs aggressive actions of bang-bang control according to a policy that achieves the target as maximum progress as possible. The resulting velocity profile forms a triangle with one switching point whose height and position are, respectively, the maximum allowed velocity and half-travel time from the initial to the target under symmetric acceleration constraint.

In summary, the simulation illustrates that PILCO, a model-based reinforcement learning that explicitly takes into account model uncertainty, is a viable approach to learning the desired property of time-optimal state and input trajectories of linear uncertain second- order systems from scratch.

Discussions and Analyses. We provide further discussions and analyses in the following.

Verifying Time-Optimality. Time-optimal control solution to double integrator with input constraint and/or velocity constraint can be found in the literature, where time-to-go function (the minimum time) is calculated through the application of PMP [6] or numerical optimization [25]. More specifically, given a priori rest-to-rest triangle velocity profile, we have zero velocity at both ends of motion. There are in general two unknowns that can be used to parametrize a triangle velocity profile: the total motion duration T and the single switching time T_{sw}. Considering the symmetric acceleration bounds and rest-to-rest boundary conditions, the only unknown for triangle velocity is the switching time $T_{sw} = T/2$. The switching point in this case also determines the bottom edge length (the minimal travel time T) of velocity-time plot uniquely. Theoretically, in ideal situations of no external disturbances and no model errors, by taking into account the input constraints, the triangle velocity profile is the intersection of forward integration along maximum acceleration curve from the initial and backward integration along the maximum deceleration curve from the target. That is, it is the velocity trajectory obtained by integrating the equation of a double integrator or second-order

mass-spring-damper system when the time-optimal input is a bang-bang control [11] (i.e. it is a piecewise constant $\pm u_{max}$ at all time) with one sign change.

Given that the area L of triangle is equal to the travel distance between the initial and the target, there is a switch from acceleration to deceleration for a triangle velocity profile, i.e. the intersection of the line of positive acceleration forward from the initial state and the line of negative acceleration backward from the target state in the phase plane of (position, velocity). Denote by time T_{sw} and by velocity (height) h at the switch point of the triangle. Let T denote the travel time (the bottom length), then T (thus $T_{sw} = T/2$) and h of switch point can be analytically computed (see Appendix). Equivalently, the time-optimal control needs to compute the distance needed to accelerate and T decelerate along the path. Note that in the case of double-integrator (14) with m=0.5, c=0, $u_{max} = 4$, it is a second order system without damping. The position trajectory is oscillating. It requires a minimum of

$$T = 2\sqrt{\frac{mL}{u_{max}}} = \sqrt{2.5} = 1.58$$

of time for traveling L=5 along a straight line with maximum acceleration/deceleration of 4 that achieves a highest speed of h= 6.32 in our case (see Appendix, Problem 1), as Fig. 3(b) shows. The switching from maximum acceleration to maximum deceleration is at half the total motion duration due to the symmetry of acceleration bounds. The switching time T_{sw} is computed as the time from highest velocity to zero or equivalently from zero to highest velocity:

$h/(u_{max}/m)$=6.32/8= 1.58/2=T/2=T_{sw},

as claimed. This requires $N_{target} = 16$ for a sampling time of 0.1. The characteristics of exact solution are learned: the learned triangle velocity profile nearly coincides with the time-optimal triangle profile.

PILCO Algorithm for model learning. PILCO uses GPs for model learning of the unknown dynamics (mass and friction coefficient). In our task, the GPs only learn from the collected dataset only valid for the given path and task, while prior knowledge about the dynamics based on physics, which have a general validity and is consistent across the tasks, is not available. The specific state response is derived from measured data for applying a given policy in the parametric form of (14)-(15) to the simulated model. In the second episode, nearly correct state response is generated without being concerned with the true parameters c/m and 1/m. The friction coefficient c can't be estimated independently of the mass m from the trajectory. The GP model is then used to form an estimator for the (approximate) posterior for optimal policy search. This makes the algorithm perform efficiently in velocity learning task for a specific platform, and the resulting policy encodes the desirable spatial and temporal correlations. However, it is also observed that sometimes the algorithm, in spite of the natural exploitation-exploration characteristics of the saturating cost function, is not guaranteed to be globally optimal. It is stuck in local optimum since the optimization problem is not convex [4, 18].

*1.Efficiency.*We further investigate the efficiency of the algorithm. As shown in Fig. 4, the agent can indeed be brought to rest at the destination from the initially rest condition within 2 episodes by an optimal action which achieves the transfer in the smallest

amount of time with practically a time-optimal trajectory. We can see two very different trajectory behaviors. The first trajectory is counter-intuitive in that it drives in reverse to the destination. This unusual behavior has a high trajectory cost due to divergence from the target, causing the rest of the trajectories across the second episode to the last one switching direction of motion by traveling toward the target from the start. The same implication has been drawn from the cost vs. episode plot of Fig. 5, where we observed that the learning converges after 2 episodes. The learning is successful as validated by the decreasing of state trajectory cost function value with the number of episodes shown in Fig. 5. After a certain number of learning episodes, the learned velocity trajectories are the same. In the first episode, the total cost is 40 on account of the unity saturating function, yet at the second episode, a new speed profile is found that can steer the car to the target with reduced cost and consistently with the reduced motion duration. The learning behavior from second episode until the final one is relatively identical with small steady-state oscillation. When overshoot occurs, turn-back is observed during learning. The policy is updated and exploration is terminated to maintain the cost at its lowest throughout the remaining learning episodes. Fig. 6 are the learning curves in the position-velocity phase plane, showing the maximization of reward with respect to time complexity in terms of the number of episodes.

*2.Cost Function.*Time-optimality is not imposed explicitly in the cost function (11) - (13).In the initial stages of learning, the predictive state mean is far from the target and the distribution is looser due to propagation of model uncertainties, so the cost function guides the agent towards the less unexplored regions. As the learned model that contains uncertainty becomes more accurate with the mean approaching to the target and the distribution tightened, the agent exploits the local regions for policy optimization. In fact, via minimization of (11) - (13) over the entire horizon $[0,T_{learn}]$ of learning episode, the maneuver or control policy tends to achieve a maximum progress per sampling time so that higher immediate reward is encouraged during the learning. This is because decaying factor to the power of $t\gamma^t$ in the time-accumulated trajectory cost sum (11) favors the immediate distance cost (13) to drive toward the target, encouraging the goal- reaching to be achieved with large progression so as to deliver a lower total trajectory cost. A metric of the error caused by the uncertainty in the learned model is comparing the predicted state response based on learned model with the state response of simulated system. The performance is gradually improved over episode as the speed of target reaching is increased by the cost optimization, validated in Fig. 5.

*3.Constraints***.**Apart from hard control boundaries, equality or inequality (possibly nonlinear, nonconvex) state constraints such as the limit on the travel distance or upper and lower limits on the vehicle position, obstacle avoidance or maximum velocity limits are very common to autonomous systems for safe operation. These requirements are not taken into consideration by PILCO in current form, in particular goal reaching is not imposed as the boundary constraint, ensuring parameter perturbations do not violate constraints during trajectory learning. Due to the objective function does not enforce the vehicle to stop at the target after the target is reached (without imposing equality constraint of reaching the target or boundary conditions at the target), in the simulation the vehicle has recovered a near time-optimal rest-to-rest linear locomotion. However, the vehicle did not converge precisely to the target state, but oscillated around the

neighborhood of the target state, since there is no terminal cost imposed on the cost function. A few trials are carried out to implement a maximum velocity on the autonomous vehicle through some tuning on the cost function. It tends to compromise to some extent and balance the cost generated from the target state and the penalties given for violating the constraints because it looks at the full horizon for policy evaluation. It shows that current method is more restricted and can only handle a limited set of objective functions and constraints.

4.2 Sim-to-Real Policy Transfer Experiment

In order to test the learned policy in a real-world environment, this section presents a sim-to-real validation experiment we conduct with a low-cost Raspberry Pi-controlled small car, AlphaBot, as shown in Fig. 7. The simple car is equipped with photo interrupters and ultrasonic sensors for state measurements. It can be controlled to accelerate and brake. We assume that the model of car belongs to a similar second-order dynamics with slightly changed model parameters, which is universal due to its base on first principles or physics. The learning task is the same time-optimal rest-to-rest steering along a linear track. This allows the reuse of the learning control obtained from the simulation. The low-cost vehicle has uncertainties in inertial and friction parameters, suitable for testing the uncertainty handling capability of learning control algorithm. In contrast to simulation scenario setup, there is an additional velocity limit for experimental car, which is a state constraint. The velocity limit stems from the electronic voltage constraints on board, and therefore is not directly handled by the control signal.

Setup. The small car is designated to begin its route at a distance of 180 cm from the wall and end at 50 cm. Its heading is fixed at zero angle of forward-looking for straight line motion. In the experiment, prior knowledge about the dynamics model (a detailed description of the dynamics that contain uncertainties) is not available or is not required. Instead, these are learnt through the Gaussian Processes. We reuse the same RBF controller that was used in the simulation in Sec. 4.1 to provide a feedback from actual vehicle state to a control input to the actual vehicle. When applied to a real vehicle, the learned policy from simulation runs thus can be viewed as an optimal demonstration of time-optimal control solution to double integrator via PMP. The control signal generated from the RL algorithm ranges from -2 to 2, and is translated into change rate of duty cycles of the motor PWM signal on board. In addition, the learning controller is data-driven in that it can directly process the sensory data to produce a maneuvering action for the vehicle motion.

Results. The experimental result is shown in Fig. 8. The car moves forward until the target is arrived. The vehicle drives at each point on the path with its allowable highest velocity, very similar to that of simulation. Fig. 8(a) shows that the real model car under the same control policy obtained by simulation reaches the desired target in the real-world experiment. The end position, despite having no overshooting and fluctuation, is slightly off the targeted 50 cm partly because of modeling uncertainties and localization errors due to lateral tracking error and sensor inaccuracies. It is observed that the vehicle follows a nearly a time-optimal velocity along the track as the desired control goal to attain. The velocity limit stems from the electronic voltage constraints on the robot, and therefore is not directly handled by the robot control signal. The learning curve is illustrated in Fig. 8(b). The total cost starts from 40 in the beginning trial of the task for

the saturating cost function. It attains a lowest cost of about 16 and a travel time of about 2.2 seconds at the seventh episode with a total experience time of 28 seconds. This is deemed very efficient since a very small number of trials is executed for successful learning. In Fig. 8(c), immediate cost at every time step in various episodes is plotted, showing the learning process and the arrival time being reduced over episodes. The decrease is, however, not monotonic, so the intermediate trajectories before the learning is complete may not be acceptable until a nearly time-optimal control input is finally obtained at convergence. It is clear that the design of reward will affect significantly the outcome of policy learning. In our case, the decrease of distance to target is directly related to reduction of travel time. The data-driven learned maneuver effectively mitigates the uncertainty propagation to accurately estimate the uncertainty of the dynamics. Accordingly, the control policy automatically tune the parameters from the recorded data accurately when restricted for the specific state-to-state transfer task with given goal-reaching path to yield an accurate state-to-state time-optimal velocity solution along the linear track. The learning achieves the goal of reaching the target as soon as possible.

Discussions and Analyses.

Differences between Simulation and Experiment. The learning task of nearly time-optimal state-to-state transfer in a scenario of short-distance driving on a linear path is the same for both simulation and experiment. The control (14)-(15) obtained from simulation of linear system is shown also effective to steer the real vehicle as fast as possible for state-to-state transfer. Both simulation and experiment show the learning reduces the accumulated distance to the target until no significant reduction of the accumulated distance, thus contributing to reduce the travel time. However, the convergence in the simulation is faster, which possibly stems from the following major differences.

The first important difference is that the learning experiment is on the basis of more complicated vehicle dynamics. In fact, in the presence of the inherent factors confronted during real world experiment that cannot be neglected oraccounted for in a simple physical model (16) pose, some extent of unknown nonlinearities and uncertainties to the actual system, the near time-optimality is much harder to ensure. In real world experiment, a more general uncertain nonlinear model is not easily available or too complicated for the design of state-to-state steer maneuvers and in the presence of parameter uncertainties as well as sensor measurement errors, or unknown external forces inherent in the vehicle-terrain interaction. In a vehicle hardware, there is an additional velocity limit (the electronic voltage bounds) to assure that fast motion does not cause harmful effect or instability on the vehicle in each trial. Other causes of discrepancies between simulation and experiment are introduced by external disturbances as a result of the inaccuracy of sensor measurements, motor characteristics and torque disturbances and variation of environment interactions such as uneven ground, drag force, complicated friction characteristics (such as Coulomb friction proportional to $\text{sign}(\dot{x})$ and aerodynamic drag force proportional to $\dot{x}^2$, in addition to viscous friction proportional to $\dot{x}$) and wheel sideslip.All the aforementioned factors cause the deviation of state trajectory in real experiment from that of the simulation under the same control. Since the task characteristics (system dynamics along a linear track on a frictional plane) and the environment for learning in simulation and experiment are very similar, a sim-to-real

transfer experiment that attempts to use the same data-driven control learned from simulation that works well as the motion policy is performed on a low-cost car. This validates the slight generalization capability (robustness and stability) of the model-based RL algorithm with Gaussian radial basis function kernel encountered for similar dynamics and task constraints.
Furthermore, due to the low-cost robot units, low PWM duty cycles translated from the control signals might not be able to drive the car, affecting the complexity of the task as well.In the experiment performed, we ascertain that the robot hasenough power available to travel a distance of L along the linear track to reach the target. Finally, we remark that the steady-state error can be corrected by switching from the learning control to a linear stabilizing controller in a neighborhood of the target position.

We see that the learning control method to realize the time-optimal state-to-state steer control in a simulated model is a good approximated time-optimal control of actual system with similar dynamics and task characteristics. Instead of providing an analytical and thorough proof of the control performance demonstrated by simulation and experiment, we give some interpretations or explanations.

Interpretation 1. A simple interpretation that follows the formulation of time-optimal point-to-point control with nonlinear model predictive control [29] that complements the experimentis as follows. Let ($\mathbf{x}$(t), u(t)=u($\mathbf{x}$(t))) be the time-optimal state-input trajectories of simulation model (16). The demonstration by the simulated system (16) provides a good understanding of the time-optimal vehicle motion with only input constraint under no disturbance and no state constraint. For successful sim-to-real experiment, we assume that the simulation model (16) can be extended to the actual system (20) in real experiment

$$\dot{\mathbf{x}}_a = \mathbf{A}\mathbf{x}_a + \mathbf{b}u_a + \mathbf{d}(\mathbf{x}_a, t), \mathbf{x}_a(0) = 0, \dot{\mathbf{x}}_a \le v_{max}, u \in U_{ad} = [-u_{max}, u_{max}] \quad (20)$$

where $\mathbf{x}_a(t)$ is the actual state, $u_a(t) = u(\mathbf{x}_a(t), \theta_a(t))$ is the same time-optimal control (14) leaned from (16), $\mathbf{d}(\mathbf{x}_a, t) \in \mathbf{D}$ is a bounded additive disturbance represented as a priori unknown function with $\mathbf{D}$ a bounded set, v_{max} is the maximum velocity limitation (or state constraint) due to hardware setting. The system (20) is assumed a slight perturbation of (16) due to small disturbance or unmodeled dynamics. To simplified analysis and gain some intuition, we can view $\mathbf{d}(t) = \mathbf{d}(\mathbf{x}_a, t)$ in (20) as the lumped model-mismatch effect of the unmodeled dynamics with respect to the simulated system model (16) used for optimal demonstration. Thus, $\mathbf{x}_a(t)$ is the actual state trajectory in the presence of adverse disturbance effect of $\mathbf{d}(t)$ as the same nominal state feedback control (16)-(17) used for (16) is also used for (20). From (20),we have

$$\begin{aligned}\mathbf{x}_a(t) &= \mathbf{C}\mathbf{z}_a(\mathbf{t}) + \int_0^t e^{\mathbf{A}\tau}\,\mathbf{d}(\tau)d\tau \\ &= \mathbf{C}\mathbf{z}(\mathbf{t}) + \int_0^t e^{\mathbf{A}\tau}(b\delta u(\tau) + \mathbf{d}(\tau))d\tau \quad (21)\end{aligned}$$

where $\delta u(t) = u_a(\mathbf{x}_a(t), \theta_a(t)) - u(\mathbf{x}(t), \theta(t))$. Given that the actual system is a slight perturbation of simulated system, the consistency or similarity of the task characteristics assumes that the Lipschitz condition

$$|\delta u(t)| \leq k_x \|\mathbf{x}_a(t) - \mathbf{x}(t)\| + k_\theta \|\theta_a(t) - \theta(t)\| \quad (22)$$

holds for some constants $k_x > 0,\ k_\theta > 0$. This is a reasonable assumption that small deviation in state yields a small fluctuation of control action. As seen from (21),(22), since the matrix **A** is stable, the difference between $\mathbf{x}(t)$ and $\mathbf{x}_a(t)$ is thus bounded by the discrepancy $\mathbf{d}(t)$between the model and actual system, the variation of controller parameters and how much the differential state-feedback control $\delta u(t)$ can attenuate the disturbance, i.e. when the disturbance is large/small, the counterbalancing control magnitude is also large/small correspondingly. Theoretically, if we assume that disturbance is bounded during motion, the system (20) can be controlled to steer to the target region using the nominal control $u(t)$learned from simulated vehicle dynamics (16) in near minimum time even in the presence of external disturbance. That is, the deviation of experimental state trajectory (20) from nominal time-optimal state trajectory of (16) is such that $\|\mathbf{x}_a(t) - \mathbf{x}(t)\| \leq \epsilon$ and $\mathbf{x}_a(T_{a,target}) \approx \mathbf{x}(T_{target}) \approx$ a given target at rest in the reachable set with the first target hitting time$T_{a,target}$of (20) largerthan $T_{target}$of (18).The deviation ϵenters the cost function (11) for the actual vehicle dynamics (20), thus it also depends on the sensitivity of the cost function on the discount factor. We can choose a discount factor $\gamma(0 < \gamma < \gamma^*)$ of cost function that guarantees the state trajectory of (20) exhibits small perturbation from time-optimal trajectory of model (16) under the same nominal time-optimal control [29] if there exists a γ^*such that the gradient of the cost (11) with respect to the deviation ϵ is upper bounded. Validity of the above consistency/similarity property that guarantees the actual system performance using the same control obtained from simulation is supported by comparing Fig. 9 and Fig. 10. Therefore, it allows the transfer of command input learned on the basis of simulated model to a similar, real kinematic vehicle driving on a linear short-distance path in real environment.

Interpretation 2. The parametric model uncertainty and external uncertainty causes the actual trajectory executed by the vehicle not exactly predictable. The equation of motion (16) with unknown but bounded additive disturbance (20) can be discretized using the Euler method as

$$v_i = \frac{x_i - x_{i-1}}{h}, \quad (23)$$

$$a_i = \frac{v_i - v_{i-1}}{h} = -\frac{c}{m} v_i + \frac{1}{m} u_i + \frac{1}{m} d_i \quad (24)$$

where v, a denote the velocity and acceleration of the simulated second-order system, h is the step size, i=1,…,N, $x_0 = 0, v_0 = 0, a_0 = 0$.The optimality condition of bang-bang control requires that the acceleration input after substituting (23) into (24)

$$\frac{x_i - 2x_{i-1} + x_{i-2}}{h^2} = \pm u_{max} \quad (25)$$

or

$$\frac{x_i - 2x_{i-1} + x_{i-2}}{h^2} = -\frac{c}{m} \frac{x_i - x_{i-1}}{h} \pm \frac{1}{m} u_{max} + \frac{1}{m} d_i \quad (26)$$

holds for the uncertain system (20) moving along a linear track to reach $\mathbf{x}_{N_{target}} \approx \mathbf{x}_{target}$. This is the optimality relation in the form

$$v_{i+1} = f(x_i, v_i, u_i\,(x_i, v_i) = \pm u_{max}, m, c, d_i) \quad (27)$$

Note that though the function f is valid and consistent across the tasks [27]. The model-based RL needs to learn the function f via (27) with optimality guarantee, i.e. the one that achieves the time-optimal control performance, from the dataset. In addition, we construct the aggressive policy which robustly steers the uncertain system from the initial state to the target as fast as possible along the trajectory predicted by the learned model f with errors caused by the uncertainty and disturbance to accurately match the time-optimal trajectories $u_i = \pm u_{max}$, with x_i confined to an interval on the linear one-dimensional line. The validity of GP prediction model of vehicle-terrain interaction sticks to the data points restricted to a linear path, while prior knowledge about the uncertain dynamics of a simple double-integrator based on physics is not available. With updated policy (state feedback), a new velocity is generated, we see in Fig. that the modeling error is reduced in subsequent episodes in which the learned state response (new velocity) from the initial to the final state is updated and coincides nearly with the time-optimal state response of simulated system. Therefore, the learned model is sufficiently accurate to capture the properties of the vehicle for specific path following task.

5. Rescaling the Velocity Profile as Expert Knowledge

Adaptability in the task of speed regulation, which is complex and varied, according to comfort, safety concerns and the path properties such as length and maximum curvature is a key feature of human driving behavior that has robustness to uncertainties. In planning a new safe velocity profile, the first step is to generate time-optimal velocity profile for the system as the reference velocity profile with the learning techniques, since it is well-known that the travel time affects the motion profile. It is necessary to know a priori the lower bound on travel time required by a given state-to-state steering task. Having attained a time-optimal velocity profile along a specified path, to trade off the time for smoothness thus energy consumption, there are a variety of safe speed profiles meeting the most restrictive a priori upper bound of speed and acceleration for each point on the path with less aggressive use of the admissible input that can be planned to reduce other cost such as energy consumption. Nevertheless, an increase in travel time is caused by a potential reduction in the achievable highest speed due to additional velocity or smoothness constraints. The new velocity profile makes the vehicle motion slower but likely smoother and energy-saving, as classical kinematics suggests.

By rescaling or regularization in time subject to the dynamic constraints, we can obtain a nearly time optimal velocity profile [22-24].Suppose the velocity has only cruise and constant acceleration sections, and we consider only triangle and trapezoid velocity profiles. A trapezoidal velocity profile （T profile) for transition between two boundary velocities consists of two ramp phases with constant acceleration/ deceleration and one cruise phase with constant speed.Triangle is a special case of trapezoid. Applying the rescaling approach to reduce the complexity of learning new motion profiles, we are particularly interested in how the learned triangle speed profile that has no constant

speed section along a linear path can be transformed to an arbitrary speed profile composed by single trapezoid or double trapezoids with cruising phases, denoted by T-profile and T^2-profile, respectively. In the whole state-to-state transfer motion duration, it is known that T-profiles are commonly used velocity schedules that result from the solution to the optimization of cost function defined by mixed time and energy, so that the resulting analytical velocity profile is a smooth trapezoid-like U shape under the condition of only acceleration constraint without velocity limit, or approximation to time-optimal velocity profiles with arbitrary boundary conditions, other than rest-to-rest [23]. Therefore, driving with a T-profile will take a longer travel time than with a triangle velocity for the same travel distance. T-profile consists of three phases: acceleration, cruising and deceleration. The acceleration can reach an appropriate cruise speed equal to or less than allowable maximum speed, requested by the user or limited by vehicle characteristics, that depends on the travel time or distance. Then, it is followed by a tunable duration of cruising constant speed, and a deceleration stage respecting the acceleration bounds. As another example, the triangle can be transformed into double trapezoids, so that a ladder-type trapezoid velocity profile (T^2- profile) is produced. T^2-profile enables the use of different cruise speeds, which lets the vehicle to select multiple durations of different constant transition speeds that are varied, and there is an additional acceleration in the middle of movement. Using an algebraic transformation (provided in Appendix), we can transform a triangle to a T-, T^2- profile under different situations for a given distance and a corresponding triangle velocity profile covering the distance without maximum velocity constraint. These families of velocity profiles based on T- or T^2- profiles are parametrized by a finite number of user-defined parameters that affect the highest speed, durations of constant speed and overall travel time. A few examples of T-, T^2- profiles are depicted in the Fig. 11, covering the same travel distance of the corresponding triangle velocity profile.

We point out that T- profile could be easily generated from this parametric speed family by assigning its respective defining parameter values complying with the specifications. The specifications can be designed by the user. In many applications, the motion execution time T_{ex} can be imposed, for example as multiple times of the minimal time traveled by the triangle velocity on the given path with fixed distance. The imposition of trajectory duration results in distinct velocity profiles guaranteeing that the travel time of the same distance on a given path is exactly T_{ex}. Another alternative is to set the highest velocity of T- profile a ratio of the maximum velocity bound, such as limited by traffic rules or on different terrain. Both have the effect of rescaling a given triangle velocity profile to make the motion slower. Similar reasoning can be applied to design a T^2- profile from a triangle or a T- profile. A detailed derivation is provided in Appendix. Rescaling examples following these alternatives for speed scheduling of the double integrator are depicted in Fig. 12 and Fig. 13.

In summary, instead of learning a new safe speed profile from scratch, the aforementioned easy-to-implement rescaling approach exploiting generated safe velocity profiles or human driver demonstrations as expert knowledge could accelerate the learning process for a wide range of safe speed profiles or may yield better learning results for complicated, high-dimensional vehicle dynamics in challenging scenarios such as slippery road. From our study here, the family of trapezoidal velocity profiles at the

expense of longer travel time can be deduced from the learned triangle velocity that is obtained from the maximum progress cost of reaching the goal independent of velocity. This eliminates the need to design new cost function and the gradient computation, and restart the whole learning episodes. Having obtained the learned minimum-time velocity profile, the transformations mentioned above are the expert knowledge that can be exploited to obtain a new safe speed profile.

6. Conclusion

Minimal-time velocity profile along a prespecified curve, as a subclass of time-optimal control problems subject to hard control constraints resulting from input saturation, state constraints and external disturbance, finds applications in a variety of autonomous systems such as autonomous driving and robotics. The imperfect modeling of system dynamics and perception of environment with significant noise makes machine learning a powerful approach to the practical, near minimum-time velocity planning for autonomous systems that do not rely on heavy dynamic model-based computations. An important aspect of this paper is adopting PILCO, an existing model-based RL which holds the state-of-the-art data efficiency, to learn a near time-optimal velocity for rest-to-rest state-to-state steer along a linear path by a vehicle with acceleration or velocity limits. The policy learned from simulation is implemented on a sim-to-real experiment with a similar vehicle path following task to illustrate the consistency of learned velocity profile closer to that obtained by time-optimal control. Our case study expands the scope of problems that can be successfully solved by model-based RL (such as PILCO) from scratch (without identification of physical parameters of vehicle motion, a priori task environment characteristics human demonstration and deriving the optimality conditions), and shows the capability of accounting for and compensating uncertainties and external disturbances. We illustrate that the safe velocity learning on different road topology and traffic flow is feasible for the challenging applications of RL algorithms, serving as a robust adaptive optimal control algorithm.

The study inspires several future researches. Firstly, one potential framework is to extend the predicted system trajectory in PILCO into Model Predictive Control and utilize Sequential Quadratic Programming to effectively deal with the constraints. Despite the one-dimensional problem we tackle here, the framework is to be examined in high-dimensional complex systems with sophisticated switching structures, similar to what NI methods do for the high-dimensional systems with complex known dynamics. Secondly, the algorithm now performs policy evaluation and optimization offline. It is crucial for future approaches to take these online since most real-world applications encourage real-time operation. Thirdly, time-optimality is forced by the long-term saturating cost in our experiment. It is suggested that future studies inspect the possibility of incorporating Pontryagin Maximum Principle into policy learning, which is a principled method with more theoretical supports.

References

1. Q. Pham, “A general, fast, and robust implementation of the time-optimalpath parameterization algorithm,” *IEEE Transactions on Robotics*, vol. 30,no. 6, pp. 1533-1540, 2014.
2. Polydoros, A. S., &Nalpantidis, L. (2017). Survey of Model-Based Reinforcement Learning: Applications on Robotics. *Journal of Intelligent & Robotic Systems,86*(2), 153-173.
3. Kober, J., Bagnell, J. A., & Peters, J. Reinforcement learninginrobotics: A survey. *The International Journal of Robotics Research*,32(11), pp.1238-1274, 2013.
4. M. Deisenroth, Carl Rasmussen. “PILCO: A Model-Based and Data-Efficient Approach to Policy Search” *Proceedings of the International Conference on Machine Learning*, 2011.
5. Ostafew, C. J., Schoellig, A. P., Barfoot, T. D., & Collier, J. Speed daemon: experience-based mobile robot speed scheduling. *Canadian Conference on Computer and Robot Vision*, pp. 56-62, 2014.
6. O. Stryk, R. Bulirsch, “Direct and indirect methods for trajectory optimization,” *Annals of Operation Research*, vol. 37, no. 1, pp. 357-373, 1992.
7. J. Bobrow, S. Dubowsky, J. Gibson, “Time-optimal control of roboticmanipulators along specified paths,” *International Journal of RoboticsResearch*, vol. 4, no. 3, pp. 3-17, 1985.
8. T. Kunz, M. Stilman, “Time-optimal trajectory generation for path followingwith bounded acceleration and velocity,” *Proceedings of Robotics:Science and Systems*, 2012.
9. Jond, H. Barghi, V. V. Nabiyev, A. Akbarimajd. "Planning of mobile robots under limited velocity and acceleration." *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, 2014.
10. Bianco, C. G. L., & Romano, M. (2007, April). Optimal velocity planning for autonomous vehicles considering curvature constraints. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 2706-2711). IEEE.
11. Liberzon, D. Calculus of variations and optimal control theory: a concise introduction. Princeton University Press, 2011.
12. Rao, A. V. (2014). Trajectory optimization: a survey. In *Optimization and optimal control in automotive systems* (pp. 3-21). Springer, Cham.
13. D.Verscheure, B. Demeulenaere, J. Swevers, J.DeSchutter, M. Diehl,“Time-optimal path tracking for robots:Aconvex optimization approach,”*IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.
14. ArdeshiriTohid, M. Norrlöf, J.Löfberg, A. Hansson. "Convex optimization approach for time-optimal path tracking of robots with speed dependent constraints." *IFAC Proceedings Volumes,* vol. 44,no. 1, pp. 14648-14653, 2011.
15. Qian, X., Navarro, I., de La Fortelle, A., &Moutarde, F. Motion planning for urban autonomous driving using Bézier curves and MPC. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 826-833, 2016.
16. Ozatay, E., Ozguner, U., &Filev, D. (2017). Velocity profile optimization of on road vehicles: Pontryagin's Maximum Principle based approach. *Control Engineering Practice*, *61*, 244-254.
17. Martinez-Marin, T. Learning optimal motion planning for car-like vehicles. IEEE International Conference on Computational Intelligence for Modelling, Control and Automation pp. 601-612, 2005.
18. Saha, O., Dasgupta, P., &Woosley, B. (2019). Real-time robot path planning from simple to complex obstacle patterns via transfer learning of options. *Autonomous Robots*, 1-23.
19. Hartman, G., Shiller, Z., & Azaria, A. (2018). Deep Reinforcement Learning for Time Optimal Velocity Control using Prior Knowledge. *arXiv preprint arXiv:1811.11615*.

20. Sprague, C. I., Izzo, D., &Ögren, P. (2019). Learning a Family of Optimal State Feedback Controllers. *arXiv preprint arXiv:1902.10139*.
21. J. Kabzan, L. Hewing, A. Liniger and M. N. Zeilinger, Learning-Based Model Predictive Control for Autonomous Racing.In *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363-3370, Oct. 2019, doi: 10.1109/LRA.2019.2926677.
22. Shin, K., & McKay, N. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, *31*(6), pp.501-511, 1986.
23. Kim, J., & Croft, E. A. Online near time-optimal trajectory planning for industrial robots. *Robotics and Computer-Integrated Manufacturing*, *58*, 158-171, 2019.
24. Wigstrom, O., Lennartson, B., Vergnano, A., &Breitholtz, C. High-level scheduling of energy optimal trajectories. *IEEE Transactions on Automation Science and Engineering*, *10*(1), pp.57-64, 2013.
25. Hauser, J., & Saccon, A. (2006, December). A barrier function method for the optimization of trajectory functionals with constraints. In *Proceedings of the 45th IEEE Conference on Decision and Control* (pp. 864-869). IEEE.
26. Nair, S., Savarese, S., & Finn, C. (2020). Goal-Aware Prediction: Learning to Model What Matters. arXiv preprint arXiv:2007.07170.
27. Song, C., & Boularias, A. (2020). Identifying Mechanical Models through Differentiable Simulations. *arXiv preprint arXiv:2005.05410*.
28. Moerland, T. M., Broekens, J., & Jonker, C. M. (2020). Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*.
29. Verschueren, R., Ferreau, H. J., Zanarini, A., Mercangöz, M., & Diehl, M.. A stabilizing nonlinear model predictive control scheme for time-optimal point-to-point motions. *56th IEEE Conference on Decision and Control,* pp. pp.2525-2530, 2017.
30. Dinev, T., Merkt, W., Ivan, V., Havoutis, I., & Vijayakumar, S. (2020). Sparsity-Inducing Optimal Control via Differential Dynamic Programming. *arXiv preprint arXiv:2011.07325*.

Appendix

Appendix contains two Euclidean geometry problems related to transforming an isosceles triangle to a single trapezoid or double trapezoids with the same area.

Problem 1: An isosceles triangle with given area L and slope u of isosceles, find the bottom length *T*.

Denote the height of switch point from acceleration to deceleration (or height of the triangle) by h. Then $h=\frac{T}{2}u$. Thus,

$$L = \frac{1}{2}T \times \frac{T}{2}u ,$$

or

$$T^2 = 4\frac{L}{u} \quad \text{(A1)}$$

Thus,

$$T = 2\sqrt{\frac{L}{u}} \quad \text{(A2)}$$

The total traversal time T increases with the travel distance L and decreases with the acceleration u. Note that (A1) can be solved iteratively by Newton method

$$T(n+1) = \frac{1}{2} * \left(T(n) + \frac{4L}{uT(n)}\right)$$

Problem 2: Transformation of a triangle to a single or double trapezoids

Anisosceles triangle with given bottom length *T*and height *h*, and an isosceles trapezoid with the same base angle (That is to say, the isosceles triangle and isosceles trapezoid have the same slope of side). We want to calculate the upper line length and the lower line length of the trapezoid necessary to make its area $r_A$times as the area of the isosceles triangle.

Two cases are considered.
Case 1. trapezoid's height is given
(a) Transform to single trapezoid (Fig. 14)
Refer to Fig. 14. Assume the ratio of the height ratio of trapezoid to the triangle is r_h, and the upper line length of the trapezoid isα. Let the area of the trapezoid be r_A times the area of triangle. For our purpose, $r_A = 1$.
The bottom length of the triangle in thetrapezoid is $r_h h \times \frac{T}{h} = r_h T$, and the lower line length of the trapezoid= $\alpha + r_h T$
The area of the trapezoid= $\frac{(\alpha + (\alpha + r_h T)) \times r_h h}{2}$

$= r_A \times$ The area of the triangle $= r_A \times \frac{1}{2} T h$

By multiplying$\frac{2}{h}$on both side, we get$r_h(2\alpha + r_h T) = r_A T$, $2\alpha = \frac{r_A T}{r_h} - r_h T$,$\alpha = \frac{1}{2}(\frac{r_A}{r_h} - r_h)T$
Therefore, the upper line length of the trapezoid $= \alpha = \frac{1}{2}(\frac{r_A}{r_h} - r_h)T$, and the lowerline length= $\alpha + r_h T = \frac{1}{2}(\frac{r_A}{r_h} + r_h)T$

(b) Transform to double trapezoids

Refer to Fig. 15. Similar to Fig. 14, denote the defining ratios $r_A, r_h$of single trapezoid as r_{A_1}, r_{h_1} for the upper trapezoid andr_{A_2}, r_{h_2} for the lower trapezoid.There are four constraints:

1. $\alpha_1 \geq 0$
2. The lower line length of the upper trapezoid ≤ The upper line length of the lower trapezoid→ $\frac{1}{2}\left(\frac{r_{A_1}}{r_{h_1}} + r_{h_1}\right)\alpha \leq \frac{1}{2}\left(\frac{r_{A_2}}{r_{h_2}} - r_{h_2}\right)\alpha$
3. $r_{A_1} + r_{A_2} = 1$
4. $h_1 + h_2 \leq V_{max} \longrightarrow r_{h_1} + r_{h_2} \leq \frac{V_{max}}{h}$

We can generate a solution by guessing ar_{A_1}first, and $r_{A_2} = 1 - r_{A_1}$. Then for such r_{A_1}, r_{A_2}, check if there is any feasible r_{h_1}, r_{h_2}satisfying other three constraints listed above. This process is iterated until r_{h_1}, r_{h_2} are found.

We can formulate the following optimization problem for the transformations.

i)Minimizing the lower line (bottom) length

$\frac{r_A}{r_h} - r_h > 0 \longrightarrow r_h \leq \sqrt{r_A}$. $\frac{r_A}{r_h} + r_h$ is a decreasingfunction when $r_h < \sqrt{r_A}$ with its minima $= 2\sqrt{r_A}$ happens when $r_h = \sqrt{r_A}$. Therefore, this problem is equivalent to maximizingr_h.

$\frac{r_{A_1}}{r_{h_1}} + r_{h_1} \leq \frac{r_{A_2}}{r_{h_2}} - r_{h_2}$, if $r_{A_1}, r_{A_2}, r_{h_1} + r_{h_2} := H = \frac{V_{max}}{b}$ is given, we can rewrite the inequality as $r_{h_1} + r_{h_2} = H \leq \frac{r_{A_2}}{r_{h_2}} - \frac{r_{A_1}}{r_{h_1}} = \frac{r_{A_2} r_{h_1} - r_{A_1} r_{h_2}}{r_{h_1} r_{h_2}}$

$\longrightarrow H r_{h_1} r_{h_2} = H(H - r_{h_2}) r_{h_2} \leq r_{A_2} r_{h_1} - r_{A_1} r_{h_2} = r_{A_2}(H - r_{h_2}) - r_{A_1} r_{h_2}$

$\longrightarrow H^2 r_{h_2} - H {r_{h_2}}^2 \leq H r_{A_2} - (r_{A_1} + r_{A_2}) r_{h_2}$

$\longrightarrow {r_{h_2}}^2 - \left(H + \frac{r_{A_1} + r_{A_2}}{H}\right) r_{h_2} + r_{A_2} \geq 0$(A3)

We can solve the equation to get one constraint on the range of r_{h_2} ($r_{h_2} \geq$ the larger root or$r_{h_2} \leq$ the smaller root of (A3) with equality), along with $r_{h_2} \leq \sqrt{r_{A_2}}$ and $r_{h_1} = H - r_{h_2} \leq \sqrt{r_{A_1}} \longrightarrow r_{h_2} \geq H - \sqrt{r_{A_1}}$, we know the feasible range for r_{h_2}. To minimize the lower line length, we have to pick the largestr_{h_2}in the feasible range.

ii)Minimizing the lower line length

$\frac{r_A}{r_h} - r_h > 0 \longrightarrow r_h \leq \sqrt{r_A}$. $\frac{r_A}{r_h} + r_h$ is a decreasingfunction when $r_h < \sqrt{r_A}$ with its minima $= 2\sqrt{r_A}$ happens when $r_h = \sqrt{r_A}$. Therefore, this problem is equivalent to maximizingr_h.

$\frac{r_{A_1}}{r_{h_1}} + r_{h_1} \leq \frac{r_{A_2}}{r_{h_2}} - r_{h_2}$, if $r_{A_1}, r_{A_2}, r_{h_1} + r_{h_2} := H = \frac{V_{max}}{h}$ is given, we can rewrite the inequality as $r_{h_1} + r_{h_2} = H \leq \frac{r_{A_2}}{r_{h_2}} - \frac{r_{A_1}}{r_{h_1}} = \frac{r_{A_2} r_{h_1} - r_{A_1} r_{h_2}}{r_{h_1} r_{h_2}}$

$\longrightarrow H r_{h_1} r_{h_2} = H(H - r_{h_2}) r_{h_2} \leq r_{A_2} r_{h_1} - r_{A_1} r_{h_2} = r_{A_2}(H - r_{h_2}) - r_{A_1} r_{h_2}$

$\longrightarrow H^2 r_{h_2} - H {r_{h_2}}^2 \leq H r_{A_2} - (r_{A_1} + r_{A_2}) r_{h_2}$

$\longrightarrow {r_{h_2}}^2 - \left(H + \frac{r_{A_1} + r_{A_2}}{H}\right) r_{h_2} + r_{A_2} \geq 0$

We can solve the equation to get one constraint on the range of r_{h_2} ($r_{h_2} \geq$ the larger root or$r_{h_2} \leq$ the smaller root), along with $r_{h_2} \leq \sqrt{r_{A_2}}$ and $r_{h_1} = H - r_{h_2} \leq \sqrt{r_{A_1}} \longrightarrow r_{h_2} \geq H - \sqrt{r_{A_1}}$, we know the feasible range forr_{h_2}. To minimize the lower line length, we have to pick the largest r_{h_2}in the feasible range.

For anisosceles triangle with given slope of legs *m* and area *A* and let its bottom length to be x. From $\frac{1}{2} \times x \times (\frac{x}{2} \times m) = \frac{mx^2}{4} = A, x = 2\sqrt{\frac{A}{m}}$, we can choose a close initial value x_0, then use Newton-Raphson method$x_{n+1} = \frac{1}{2}(x_n + \frac{4A}{mx_n})$ to update its value iteratively to find a better approximation to$x$.Suppose we have an approximation to x.

Single trapezoid

If $r_A = 1$, and an approximation to x is available

The upper line length of the trapezoid $= \frac{1}{2}(\frac{1}{r_h} - r_h)x$

The lower line length $= \frac{1}{2}(\frac{1}{r_h} + r_h)x$

Ladder (double) trapezoids

The upper line length of the upper trapezoid = $\frac{1}{2}\left(\frac{r_{A_1}}{r_{h_1}} - r_{h_1}\right)x$

The lower line length of the upper trapezoid = $\frac{1}{2}\left(\frac{r_{A_1}}{r_{h_1}} + r_{h_1}\right)x$

The upper line length of the lower trapezoid = $\frac{1}{2}\left(\frac{r_{A_2}}{r_{h_2}} - r_{h_2}\right)x$

The lower line length of the lowertrapezoid = $\frac{1}{2}\left(\frac{r_{A_2}}{r_{h_2}} + r_{h_2}\right)x$

where $r_{A_1}, r_{A_2}, r_{h_1}, r_{h_2}$must satisfy the constraints

Case 2. Trapezoid's bottom is given. Let T be the bottom length of triangle.

For single trapezoid, suppose the lower line length is given byλT. We have $\frac{1}{2}(\frac{r_A}{r_h} + r_h)T = \lambda T$, i.e.${r_h}^2 - 2\lambda r_h + r_A = 0$.Let the discriminant = $4\lambda^2 - 4r_A \geq 0$, or$\lambda \geq \sqrt{r_A}$ for existence of real solutions. Solving the quadratic equation, we get $r_h = \lambda \pm \sqrt{\lambda^2 - r_A}$. For the upper line length = $\frac{1}{2}\left(\frac{r_A}{r_h} - r_h\right)T \geq 0$,${r_h}^2 \leq r_A \leq \lambda^2$,so the $\lambda + \sqrt{\lambda^2 - r_A}$solution is not possible. Therefore, we have $r_h = \lambda - \sqrt{\lambda^2 - r_A}$. Therefore, we obtain the trapezoid

Height = $r_h h = \left(\lambda - \sqrt{\lambda^2 - r_A}\right)h$

The upper line length = $\frac{1}{2}\left(\frac{r_A}{r_h} - r_h\right)T = \frac{1}{2}\left((\lambda + \sqrt{\lambda^2 - r_A}) - (\lambda - \sqrt{\lambda^2 - r_A})\right)T = \sqrt{\lambda^2 - r_A}T$

For T^2 profile, if the lower bottom length of the upper trapezoid is λ_1T, and the lower bottom length of the lower trapezoid is λ_2T, we can get result analogous to the single trapezoid case:

Upper trapezoid:

$\lambda_1 \geq \sqrt{r_{A_1}}$,Height = $\left(\lambda_1 - \sqrt{{\lambda_1}^2 - r_{A_1}}\right)h$,The upper line length = $\sqrt{{\lambda_1}^2 - r_{A_1}}T$

Lower trapezoid:

$\lambda_2 \geq \sqrt{r_{A_2}}$,Height = $\left(\lambda_2 - \sqrt{{\lambda_2}^2 - r_{A_2}}\right)h$,The upper line length = $\sqrt{{\lambda_2}^2 - r_{A_2}}T$

Since the lower bottom length of the upper trapezoid≤The upper line length of the lower trapezoid, wehave an additional constraint between $\lambda_1$and λ_2:

$\lambda_1 \leq \sqrt{{\lambda_2}^2 - r_{A_2}}$.

Figures

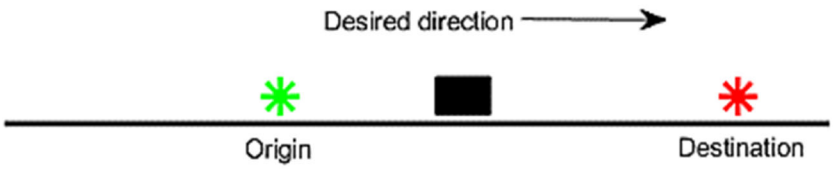


Fig. 1. Setup of one-dimensional state-to-state transfer task. The black box depicts the car, which is modeled using a point-mass double integrator. The car begins moving from the origin (green star) at rest along a straight line to reach the target (red star) along a rough plane. The task involved the execution of different acceleration control policies on the double integrator with embedded uncertainties from the same rest state to reach a target state. The resulting state–input pair and cost at each sampling time point were recorded.

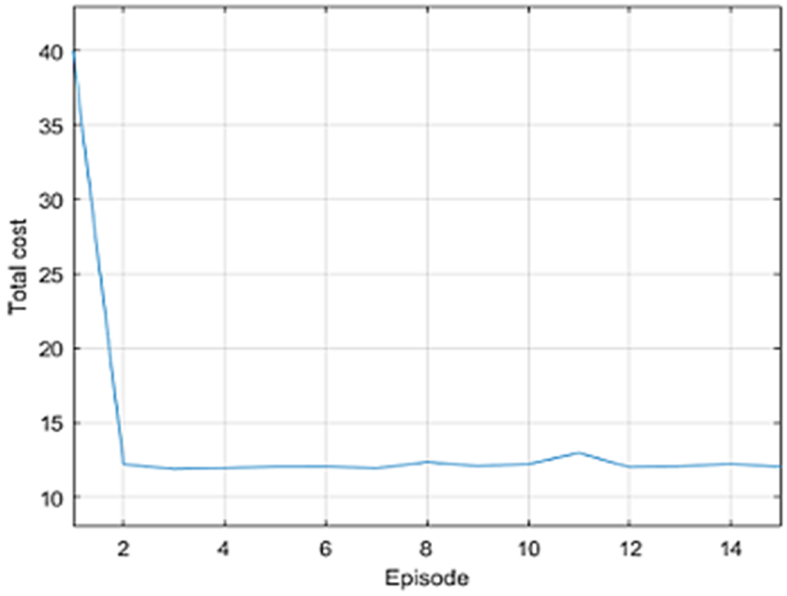
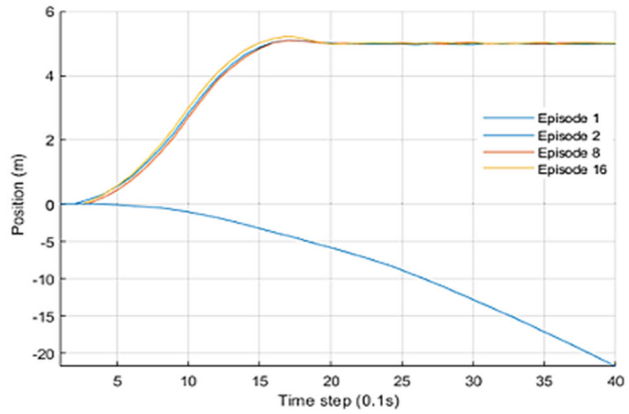
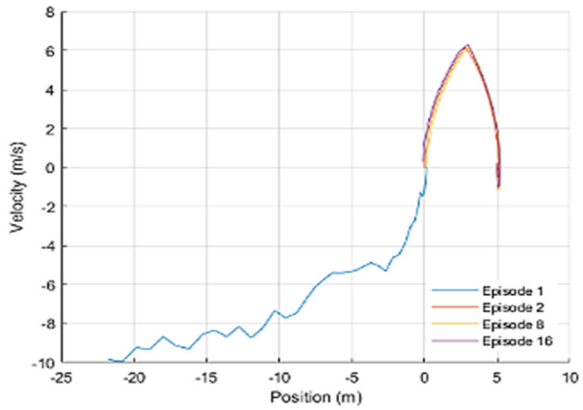


Fig. 2.Total cost vs. episode. The cost is reduced until (near) convergence is achieved at the second episode, and is stable after convergence.

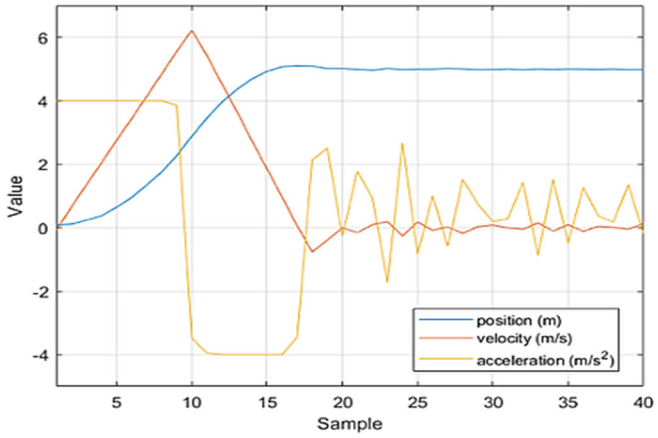


(a)

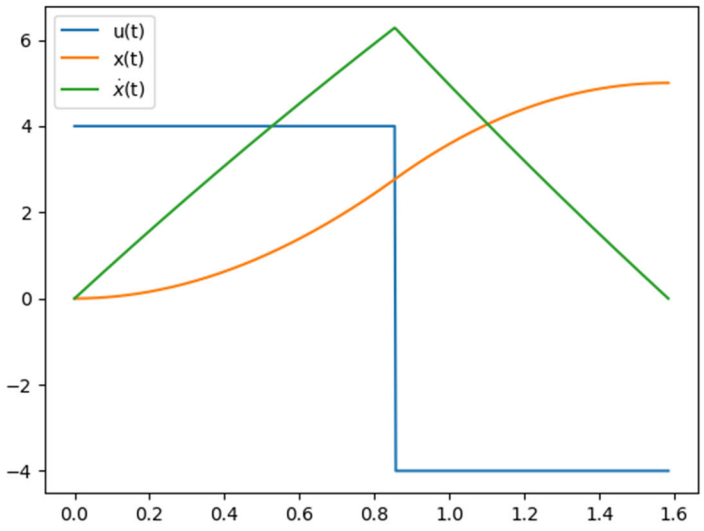


(b)

Fig.3. Learned response in time and phase plane. Episode 1 features the car reversing. In Episode 2, a correct, nearly time-optimal motion toward the target was produced. (a) Vehicle position at each episode over 40 sampling time steps spanning a time horizon of 4 s. (b) Predicted position–velocity trajectories in the phase plane through application of data-driven control on learned model for the goal-reaching task along a linear track. The state trajectories became more accurate (time-optimal) with respect to the predicted velocity trajectory as the model was updated during learning. The time-optimal velocity trajectory upon convergence with one instance of switching is shown.



(a)



(b)

Fig. 4.(a)Converged learning outcome of state (position, velocity) and input (acceleration) trajectories of the vehicle. After reaching the neighborhood of target, the vehicle tries to stay at the target at rest. There is little steady- state oscillation around the target since the vehicle can't decelerate fast enough to rest at the target. (b) The analytical solution with minimum time 1.58443.We see that the temporal characteristics of time-optimal trajectories is learned: the velocity profile is a triangle and the acceleration input exhibit the bang-bang control. The discrepancy is due to the mismatch between the objective and time optimality.

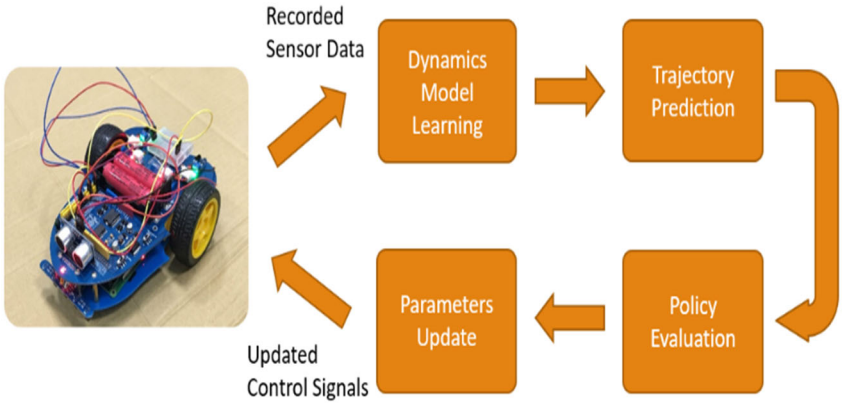
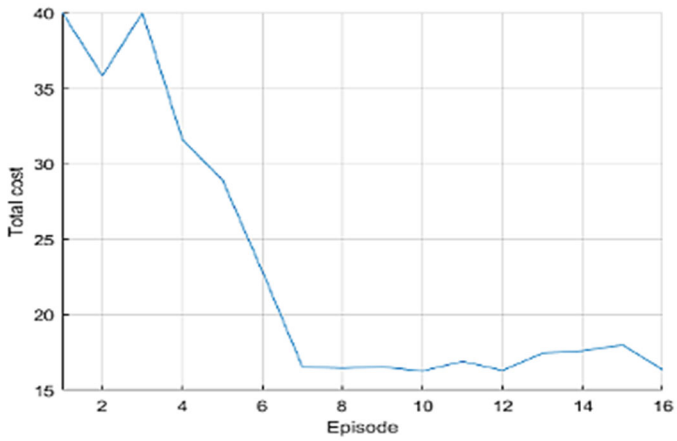
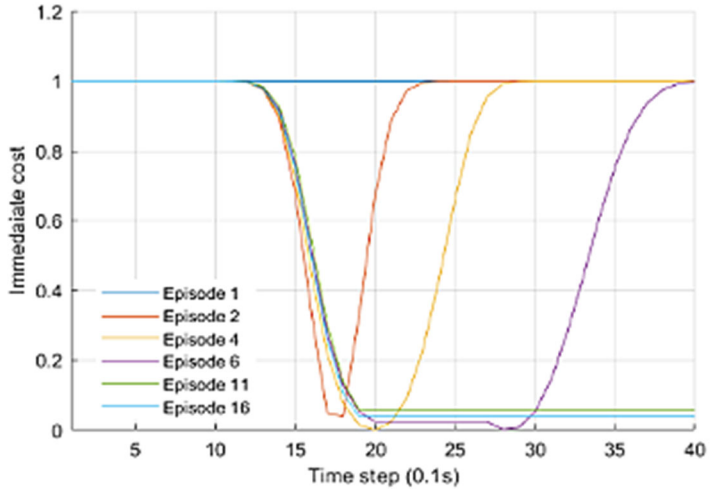


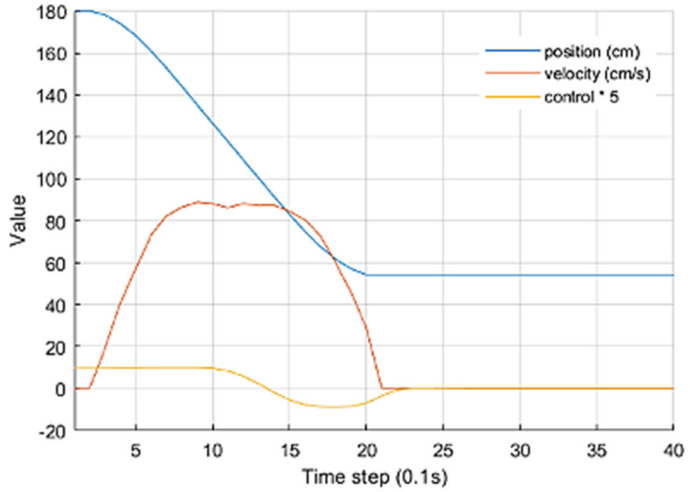
Fig. 5. The low-cost model car AlphaBot for experiment (left photo) and the experimental set-up of driving a AlphaBot along a linear track.



(a)

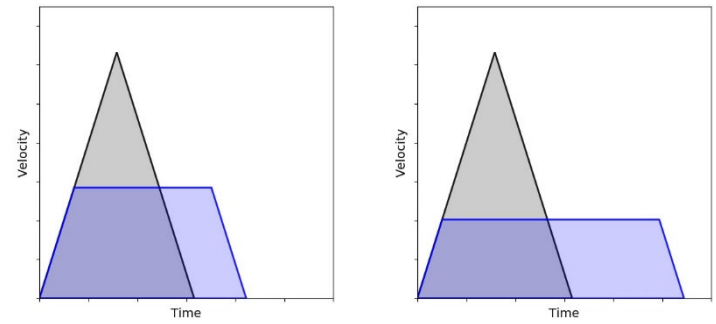


(b)

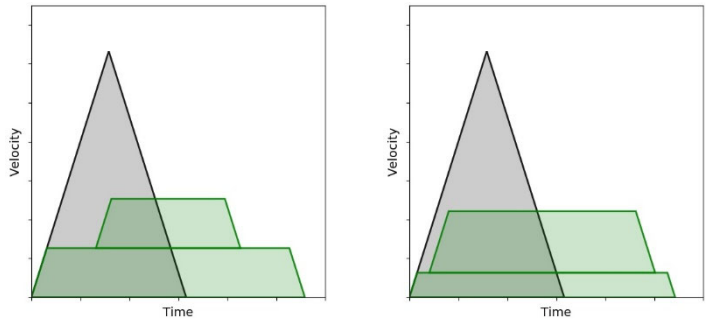


(c)

Fig.6. Experimental results. (a) Graph of total cost against episodes.The trajectory cost decreased after some transients until convergence at the 7th episode where a travel time of approximately 2s was achieved. (b)Intermediate costs in selected episodes.(c) Outcome with respect to learning state and control trajectories.

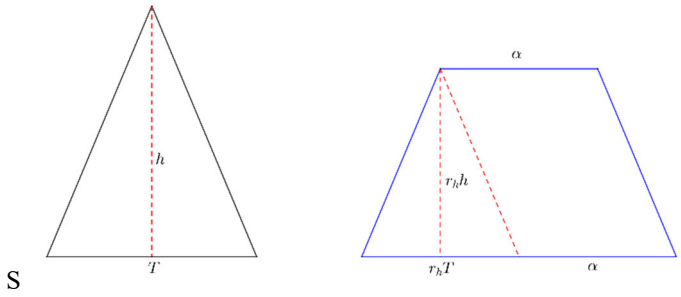


(a)



(b)

Fig. 7. Transformation of a triangular speed profile with symmetric acceleration bound to different trapezoid speed profiles with an additional velocity limit in the velocity–time plane considered. (a) Examples of T (trapezoidal) profile: trapezoid with one span of time of constant speed at a height equal to the maximum velocity and with sides whose slopes are equal to the maximum acceleration and deceleration. (b) Examples of a T^2 (double trapezoidal) profile: a ladder-type velocity profile with spans of time of constant speed and with sides whose slopes are equal to the maximum acceleration and deceleration. The total distances traveled by the new velocity profiles are the same as those of their triangular counterparts (for the travel distance L). The constant velocity section in T velocity profiles must be lower than the maximum velocity.



S

Fig. 8 (left) A triangle and (right) a trapezoid of the same area or known area ratio

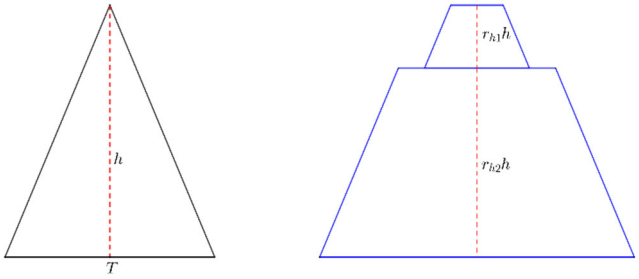


Fig. 9 A triangle (left) and double trapezoid (right) of the same area or known area ratio

Table 1. shows the switching time t_{sw}, minimum time T^* of the triangle profile in the presence of $\mp 4\%$ deviation of both mass and friction coefficient simultaneously with accurate c=0.1, m=0.5 for given L=5 and $u_{max} = 4$. For comparison, $t_{sw} = 0.79057, T^* = 2t_{sw} = 1.58114$ for the case of m=0.5, c=0. It indicates the friction clearly slows down the fastest motion.

m, c / Time	m=0.5 c=0	m=0.5 c=0.1	m=0.5 c=0.096	m=0.5 c=0.104	m=0.48 c =0.1	m=0.52 c=0.1
t_{sw}	0.79057	0.854717	0.852088	0.857352	0.836147	0.872974
T^*	1.58114	1.58443	1.58418	1.5847	1.55229	1.61595