Learning Symbolic Forward Models for Robotic Motion Planning and Control

Hirotaka Moriguchi[1,2] and Hod Lipson[2]

[1]Department of Computer Science, the University of Tokyo, Tokyo, Japan
[2]Creative Machines Lab, Cornell University, Ithaca, NY, USA
hmori@nii.ac.jp, hod.lipson@cornell.edu

Abstract

Physiological studies suggest that humans have internal dynamics models for both themselves as well as their environment, which are integral components in motion planning and control. Although robotic systems rely on similar models, a primary constraint for robotic applications is how such models are acquired and developed. Traditionally human engineers derive the dynamics models for robots; this approach is not scalable for increasingly complex designs. As a result, there is growing interest in model inference methods, which automate the modeling process and extends the design range of robots. This paper proposes a novel method that infers dynamics models as mathematical expressions via Symbolic Regression and applies them for robotic motion planning and control tasks. The advantage of this expression is not only the accuracy but also the computational efficiency. Experimental results on underpowered pendulum domains validate that our inferred models enable fast motion planning and real-time control based on rapid re-planning, with significantly superior results over Support Vector Regression and Gaussian Process Regression.

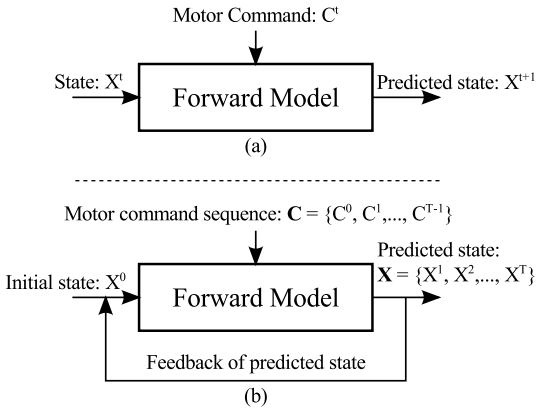


Figure 1: Diagrams of forward modeling. (a) One-step state prediction with forward model, and (b) iterative state prediction with internal feedback for simulating command sequences.

Introduction

Recent advancement in robotics has resulted in multitude of morphologically diverse robots, ranging from joint-based, legged robots to soft, continuous robots. The traditional approach to designing robot controllers requires that human engineers derive a dynamics model using first principles and prior knowledge about robots. However, as these robots increase in complexity, obtaining the dynamics model using analytical methods becomes significantly more difficult. Instead, inferring a dynamics model via machine learning approaches is a promising alternative.

Physiological evidences suggest that humans also acquire dynamics models that are vital for motion planning and control (Wolpert et al., 1995). Such dynamics models are classified into two types: *forward* and *inverse models* (Kawato, 1999). Forward models predict the consequence of motor commands, while inverse models determine the necessary motor commands to achieve a desired state transition.

The goal of this work is to infer a robot's forward model and to effectively apply it in motion planning and control tasks. In robotics, forward models take current state and motor command as input, and predict the next state without actually executing the command (Fig. 1(a)). Furthermore, they can also simulate command sequences of arbitrary length by iterating one-step predictions with internal feedback loop (Fig. 1(b)). The latter provides an infrastructure for subsequent command optimization, which generally takes the form of motion or trajectory planning.

For applications in robotic motion planning, forward models should be accurate as well as computationally efficient. Model accuracy is essential as faulty prediction may lead to misleading optimization. Since simulating command sequences requires iterative use of forward models, even minor errors in individual predictions can accumulate, resulting in significant discrepancies over the course of the simulation. Real-time re-planning is an effective remedy to this problem; however this requires that the model is computationally lightweight for rapid evaluations under a strict time constraint.

This paper introduces a novel method to infer the forward model of an arbitrary robot and apply them for mo-

Table 1: Comparison of autonomous modeling methods for robotic motion planning and control.

Authors	Type of models	Algorithm	Usage
Sturm et al. (2008)	forward and inverse	GPR	feedback motion control
Nguyen-Tuong and Peters (2008)	forward and inverse	GPR	feedback motion control
Dearden and Demiris (2005)	forward	GPR	state prediction
Bongard et al. (2006)	morphological	EA	motion planning (offline)
Ours	forward	SR	motion planning (offline, real-time)

tion planning and control problems. Our method uses Symbolic Regression (SR) (Koza, 1992) for model inference. Models inferred via SR are mathematical expressions that accurately explain robot's dynamics and are computationally lightweight. Experiments on underpowered pendulums show that the accuracy of our models are comparable to those learned with Gaussian Process Regression (GPR), while being superior to those with Support Vector Regression (SVR). Another advantage of SR models is that they can be evaluated for prediction at least three orders of magnitude faster than GPR and SVR models. This allows for fast motion planning and real-time control based on rapid re-planning. For motion planning, our method can find a more desirable plan with smaller computational effort compared to GPR and SVR-based methods. Furthermore, our method can achieve large performance gain via real-time re-planning, while methods based on GPR or SVR models cannot meet strict time constraints.

Background and Related Work

There are a wide range of approaches and applications for autonomous modeling of robots' dynamics. This section provides a brief survey of previous studies, summarized in Table 1.

Sturm et al. (2008) investigated an autonomous modeling approach that used Gaussian processes to infer the dynamics model of robot arms. Their models inferred the relationship between the motor targets of all joints and resulting pose of the arm. In contrast, our formulation relates the actuated torque or force with resulting state transition. Our approach allows for better generalization and applications in underpowered control domains such as legged locomotion.

Nguyen-Tuong and Peters (2008) proposed similar approach for autonomously modeling the dynamics of robot arms. Their models inferred the inverse kinematics of robot arms and are used in real-time feedback control. They inferred the model using Local GPR (LGPR) that can be inferred and evaluated efficiently. While we also focus on computational efficiency, we extend the use of such efficient models from feedback control to rapid motion planning and real-time re-planning.

Dearden and Demiris (2005) proposed a forward modeling approach based on Gaussian processes to model two motored arms. Their model relates motor commands to the resulting arm motions. However the motor command in their work is binary, while our experiments investigate continuous motor command.

Bongard et al. (2006) inferred the morphology of robots autonomously via an Evolutionary Algorithm (EA) and modeled them in a 3D simulator. The simulated model was used as a surrogate for the real robot and was sufficiently accurate to develop gait. An advantage of their approach was resilience against unexpected damage. However, this work relied on the accuracy of the 3D simulator to develop the model. The design of such 3D physics simulations raises the same fundamental issue of requiring laborious derivations from human engineers. Moreover, since their 3D models require heavy computation, they are not suitable for real-time control.

For goal-oriented control tasks, behavior-based control approaches, such as Reinforcement Learning (Sutton and Barto, 1998) and Neuroevolution (Yao, 1999), are widely studied as alternatives to model-based approaches. These approaches do not rely on models, but instead try to optimize sensorimotor mappings to achieve predefined goals. Although they are successful for achieving given goals, such as inverted cart-pole tasks, they lack the ability to generalize the learned knowledge to other tasks.

Learning Forward Models

Forward Modeling

In this work, we seek to find a *forward model* that explains the dynamic relationship between given commands and robot's state transition. We assume discrete time dynamics in which a robot's state at time t is represented as a set of m sensored values $X^t = \{x_1^t, ..., x_m^t\}$. A forward model for such a dynamics is a function that predicts the state X^{t+1} at time $t+1$ as

$$X^{t+1} \approx f(X^t, C^t), \tag{1}$$

where $C^t = \{\tau_1^t, ..., \tau_n^t\}$ is a set of n command signals at time t.

An advantage of this formulation is that it can extrapolate the resulting motion of an arbitrary length of motor commands. That is, given initial state X^0 and a command sequence $\boldsymbol{C} = \{C^0, ..., C^{T-1}\}$, resulting state at time

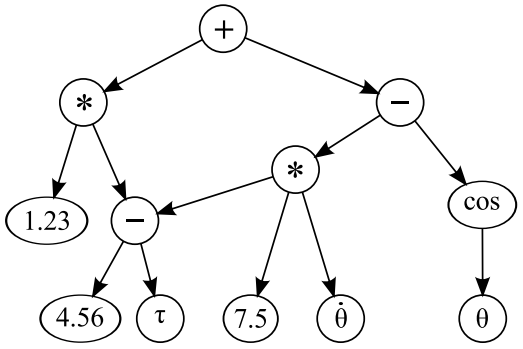


Figure 2: A sample SR model. The DAG representation of a function:
$f(\tau, \theta, \dot{\theta}) = 1.23 * (4.56 - \tau) + (4.56 - \tau) * 7.5 * \dot{\theta} - cos(\theta)$

$t \in [1:T]$ can be predicted as

$$\begin{aligned} X^t &\approx f(X^{t-1}, C^{t-1}) \\ &\approx f(f(X^{t-2}, C^{t-2}), C^{t-1}) \\ &\approx f(f(...f(X^0, C^0)...), C^{t-1}). \end{aligned} \quad (2)$$

In this study, all state variables are continuous. The modeling problem is simplified by explicitly predicting second-order differentials of these state variables. We assume that state variables X can be decomposed as $X = \langle q, \dot{q} \rangle$, where q is a set of linear and angular position variables, and $\dot{q}$ is a set of their first-order differentials (i.e. velocity variables). We formulate the modeling problem as predicting the function:

$$\ddot{q}^{t+1} \approx g(q^t, \dot{q}^t, C^t),$$

instead of directly inferring function f in Eq. 1. We calculate state values $\dot{q}^{t+1}$ and q^{t+1} via following integration:

$$\begin{aligned} \dot{q}^{t+1} &\approx \dot{q}^t + \ddot{q}^{t+1}\Delta t \\ q^{t+1} &\approx q^t + \dot{q}^{t+1}\Delta t. \end{aligned}$$

The robot's state is represented as a set of angular and linear parameters in most robotic systems. By using generalized state variables, this approach of inferring second-order differential systems can be readily adapted to arbitrary robotic systems. To generate training data set for model inference, random commands are sent to robot's actuators, and $\langle \ddot{q}^{t+1}, \dot{q}^t, q^t, C^t \rangle$ at each time step is collected as a training data point.

Learning Models with Symbolic Regression

An SR uses evolutionary algorithm that searches mathematical expressions to explain a given data set. SR has been successfully applied to infer non-linear dynamics, such as conserved laws of nature, accurately (Schmidt and Lipson, 2009). Our work uses SR to search for mathematical expressions that explain the relationship that exists in the training data. The fundamental idea of SR is to use Genetic Programming (GP) to evolve populations of expressions and selectively generate populations of lower error. Mathematical expressions are represented as Directed Acyclic Graphs (DAGs). GP searches the space of possible graphs and minimizes error by applying genetic operators, such as mutation and crossover. A sample mathematical expression and its DAG representation are shown in Fig. 2. We used Eureqa (Schmidt and Lipson, 2009) as SR implementation in our experiments.

Motion Planning and Control

Offline Motion Planning

For motion planning, we propose an approach that directly searches the command space via a hill-climbing heuristic, which searches for an optimal command sequence to maximizes a target function. Inferred forward models are used for simulating candidate command sequences in such a function. Optimization algorithm is sketched in Algorithm 1.

Algorithm 1 Motion planning with forward models using a hill-climbing heuristic

```
if Offline planning then
    C_best ← random commands
    X^0 ← initial state
else if Real-time planning then
    C_best ← current motion plan
    X^0 ← observed state
end if
E_tmp = target(C_best, X^0)
repeat
    for all C in neighbor(C_best) do
        if target(C, X^0) > E_tmp then
            C_best = C
            E_tmp = target(C, X^0)
        end if
    end for
until Goal condition satisfied or allotted iterations expire
return C_best
```

Note the evaluation of command sequences is completely dependent on the predicted state transition. Thus, the optimality of the plan depends highly on the accuracy of the predictive model. On the other hand, since the command space is high-dimensional and continuous, the search for the optimal command requires numerous iterations. Therefore, rapid evaluation of target function is vital. Since mathematical expressions are evaluated extremely efficiently on modern computers, SR models can evaluate the function quickly.

Real-time Motion Control by Re-planning

In practice, the fundamental issue of predicting state transition with forward models is the accumulation of errors

Table 2: Specification of the experiments.

Arm size (H×W×D)	2×0.4×0.2m
Arm mass	1kg
Gravity	10m·s^{-2}
Maximum torque for P_1	$\tau_{\max}$: 1.25kg·m
Maximum torque for P_2	$\tau_{\max}$: 4kg·m
Time step resolution	60Hz
State variables	$X = \{\theta_1, \dot{\theta}_1\}$ (P_1) $X = \{\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2\}$ (P_2) $\theta_1, \theta_2 \in [-\pi : \pi]$
# of neighbors in Algorithm 1	10

over iterative use of prediction as seen in Eq. 2. We resolve this problem using real-time re-planning to adapt accordingly to the errors. As the robot obtains real-time sensor values, it is able to ground the predicted state with the recorded observations. This is implemented by modifying offline planning in accordance to the new observations. In case of re-planning, the hill-climbing algorithm takes the observed state as the initial state and current motion plan as initial command sequence. Since typical robotic systems require high frequency control and feedback, real-time planning must be equally constrained by such critical restrictions. SR models are sufficiently computationally efficient to allow for re-planning, while GRP and SVR models are not. Although re-planning was introduced primarily to adapt to cumulative error, it can start with random motion and plan the motion in purely online fashion.

Experiments

In this section, we present experimental results. We evaluate our method in the motored single and double pendulum problems, called P_1 and P_2 (Fig. 3). While robots in these problems are mechanically simple, motion planning in this domain remains a challenge for autonomous robotic controllers. Pendulums used in our experiments are underpowered, and thus, achieving most angular positions is a non-trivial task which often requires unexpected motions.

Experimental Settings

P_1 is composed of an arm that is hinged to a stationary point via a motored joint. P_2 has two arms: motor joints connect the first arm to stationary point, while connecting the second arm to the first arm. These pendulums are simulated with Bullet Physics Library (Coumans, 2010), a popular, open-source 3D physics simulator. Detailed specification of the experiment is provided in Table 2.

The goal of the control task in this domain is to move the only arm (of P_1) or the upper arm (of P_2) to the upright position within allotted time steps (i.e., 600 steps in P_1, and 1200 steps in P_2). Task start with the initial state

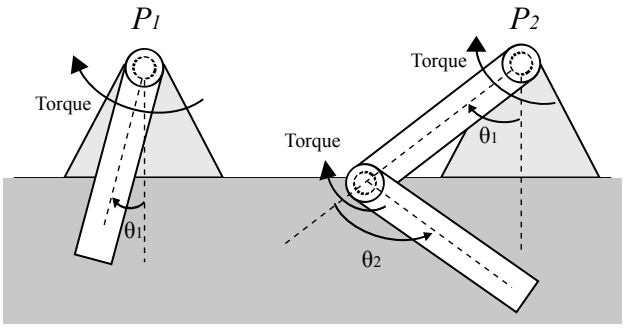


Figure 3: Single (P_1) and double (P_2) motored pendulum.

of $X^0 = O$, where the arms are in the stable equilibrium position. The torque output of each joint motor is limited so that controllers cannot reach to the upright position by simply applying maximum torque in a single direction. Instead, a successful motion requires that the pendulum be swung to accumulate sufficient momentum to eventually achieve the upright position. This additional complexity makes it difficult for existing automated motion controllers to achieve the goal, since evaluating long motor commands via forward models is vital to planning successful motion.

To generate training data set, we actuate a robot using random motor commands. For the pendulum domain, each motor is actuated with a randomly generated torque curve whose torque, $\tau(t)$, at time t is calculated as

$$\tau(t) = \frac{\tau_{\max}}{\sum_{i=1}^{N} a_i} \cdot \sum_{i=1}^{N} a_i \cdot \sin(b_i \cdot t + c_i), \qquad (3)$$

where $\tau_{\max}$ denotes maximum torque, and a_i, b_i, and c_i are random values drawn from uniform distribution under following constraints;

$$\begin{aligned} a_i &\in [0:1], \\ b_i &\in [\frac{1}{4} \cdot \frac{\pi}{60} : \frac{\pi}{60}], \\ c_i &\in [0:2\pi]. \end{aligned}$$

This formulation results in a composite wave of N individual sine waves. We set $N = 3$ throughout the experiments. Random actuation lasts for 1 minute, resulting in 3600 time steps. In the double pendulum domain, two distinct torque curves are generated with different random seeds.

Model Inference and Cross-validation Evaluation

Given the training data, we can learn forward models using off-the-shelf regression algorithms. We compare Symbolic Regression (SR) with Support Vector Regression (SVR) and Gaussian Process Regression (GPR). We used Eureqa (Schmidt and Lipson, 2009), libSVM (Chang and Lin, 2001), and Weka (Hall et al., 2009) libraries for SR, SVR, and GPR, respectively.

Table 3: Comparison of regression algorithms. Listed numbers denote correlation coefficients of inferred models. Computation time for inferring each model is shown in parenthesis.

Target	GPR	SVR	SR (short)	SR
$\ddot{\theta}_1$ in P_1	0.9996 (≈4min)	0.9733 (<1sec)	**1** (=4min)	**1** (=2hours)
$\ddot{\theta}_1$ in P_2	0.7063 (≈4min)	0.9588 (≈18sec)	0.9639 (=4min)	**0.9700** (=8hours)
$\ddot{\theta}_2$ in P_2	0.7455 (≈4min)	0.9286 (≈27sec)	0.9080 (=4min)	**0.9751** (=8hours)

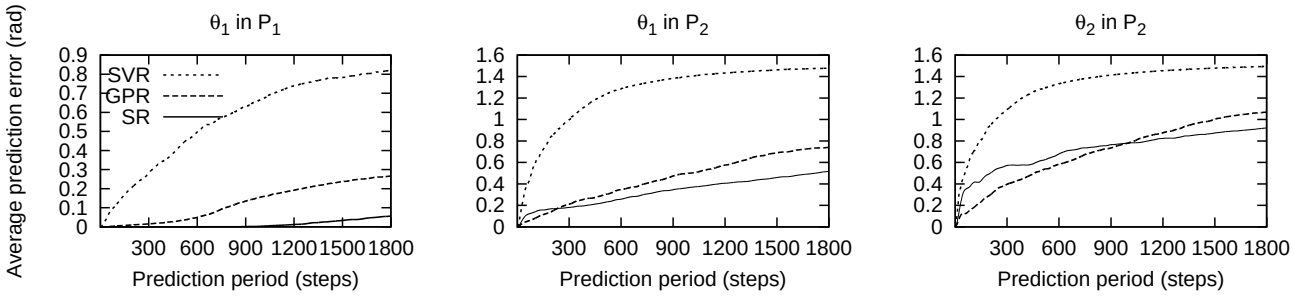


Figure 4: Average error on varying prediction periods.

The model inference results of all approaches were compared on cross-validation data sets and performance is measured with correlation coefficient. The results are summarized in Table 3. Computation time for inferring models on Intel Core2 Duo 3.06GHz are also listed in the table. Since SR is a stochastic process, longer training periods may yield better results, while GPR and SVM are deterministic algorithms that do not improve with additional time. We tested on short, being matched with the time GPR inference took, and long training period for SR. Since model inference is an offline process, time constraints are typically not strict.

The results indicate that SR is superior to both GPR and SVR, given long training time. SR inferred a virtually perfect model for $\ddot{\theta}_1$ in P_1. Even with shorter training period, SR models are comparable to those inferred with GPR or SVR. In the following experiments, we use SR models inferred with longer training period.

Model Accuracy on Novel Command Sequences

Since cross-validation results on training data sets do not necessarily reflect generalization performance of inferred models on novel data sets, we used additional tests to inspect and analyze differences in these models. To evaluate models on novel data sets, we generated 20 random torque curves using Eq. 3 with different random seeds and predicted resulting motion using the inferred models. Their prediction error was evaluated as the difference from actual motion. Since one of our concerns is the effect of cumulative errors, we vary the prediction period from 1 to 1800 steps to see how each model behaves during iterative predictions. Average predictive error on each joint angle over the prediction period is shown in Fig. 4.

We can readily see that average prediction error tends to increase, as the prediction period gets longer. This implies that the cumulative error harms predictive performance over iterations. For predicting θ_1 in P_1, it is clear that the error is less pronounced in the SR models, suggesting that the symbolic representation has a more consistent model representation. SVR did poor job on novel data sets, in spite of good performance on cross-validation. The differences of performance among three models are statistically significant ($p < 0.05$) for prediction periods of 60, 120, 600, and 1200 steps.

In P_2, SVR again marked poor performance, for predicting both θ_1 and θ_2. GPR and SR performed comparably. While GPR models outperformed SR models for short periods (i.e., lower than 233 and 997 steps for θ_1 and θ_2, respectively), SR was superior for long periods. This implies that SR models would be more robust to cumulative error over iterative predictions. Another implication is that cross-validation results, in which GPR models are evaluated badly, do not reflect the models' generalization performance appropriately.

Evaluation of Offline Motion Planning

Given inferred dynamics models, the motion planning is formulated as an optimization problem. The target function is defined as follows:

$$\mathrm{target}(\boldsymbol{\theta_1}) = \max_{\theta_1^t \in \boldsymbol{\theta_1}} |\theta_1^t|$$

where $\boldsymbol{\theta_1} = \{\theta_1^0, ..., \theta_1^T\}$, and θ_1^t is the angle of the target joint at time t. Only the initial state is provided to the models. Given candidate command sequence $\boldsymbol{C}$ and initial state variables X^0, the algorithm simulate entire state transition over T steps. Our method plans the motion by searching for

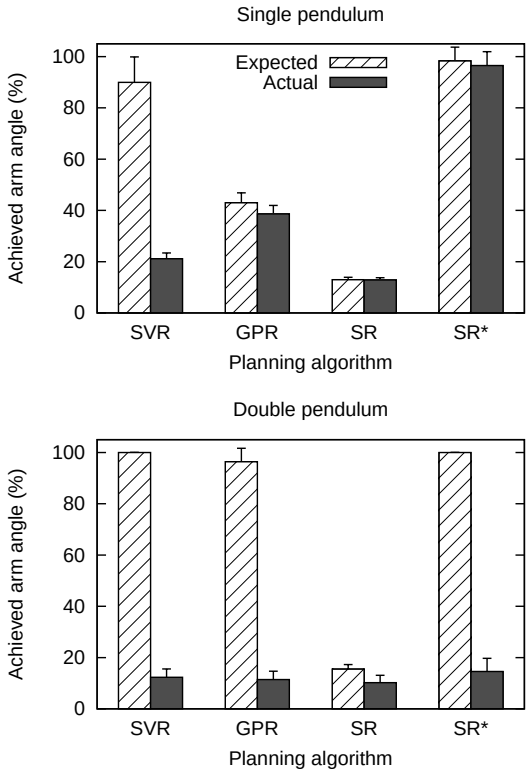


Figure 5: The performance on motion planning tasks.

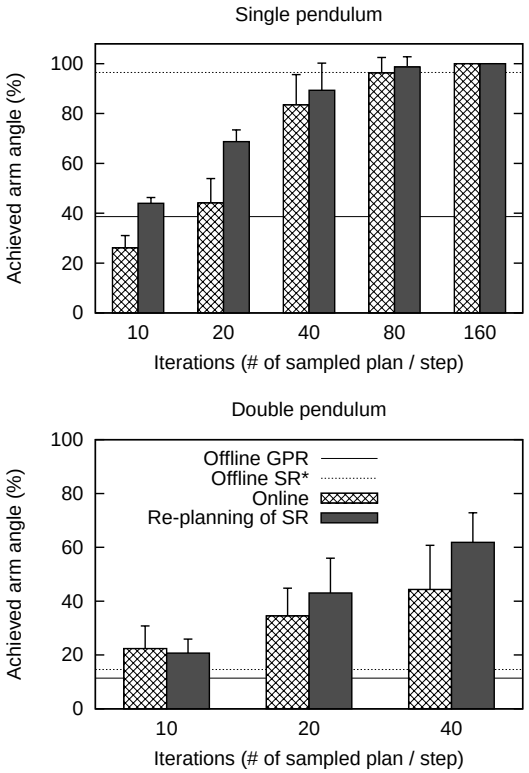


Figure 6: The performance of real-time planning. The performances of offline methods are given as baselines.

a command sequence whose simulated motion maximizes this function.

To evaluate motion planning performance of each model, the hill-climbing approach, described in Algorithm 1, was applied for 20 runs per each. The planned commands were executed on actual robot, and their actual performance were evaluated based on Eq. 4. We compared the actual performance with the expected performance to analyze the discrepancy between predicted and actual performance.

In initial tests, the hill-climbing approach for all models was executed for 500 iterations. However, there is a significant difference in the computational efficiency of SR, GPR and SVR, with SR executing at least three orders of magnitude faster than GPR and SVR in both P_1 and P_2 experiments. To provide comparable results with respect to computational effort, an additional SR, called SR*, experiment was implemented with 500000 iterations.

The performance of the SVR, GPR, SR and SR* motion planning is shown in Fig. 5. It is clear that SVR and GPR tends to overestimate its capability, often allowing for highly optimized plans that do not translate to real world results. In comparison, SR provides a truer prediction of the real world even if the hill-climbing problem becomes subsequently harder. When normalized for computational effort, SR* does a significantly better job, compared to SVR and GPR.

In the double pendulum problem, SVR and GPR predicted a nearly optimal solution with certainty, but in fact, the actual performance was poor. SR achieved similar performance, but had a more realistic prediction. Although SR* had the best actual performance, its performance was far poorer than expected.

Evaluation of Real-time Motion Control

This section provides the comparison for the real-time re-planning motion control. We experimented two re-planning approaches: first approach started with an initially optimized plan (i.e., re-planning of the plan generated by SR that is described in the last section); while second one started with a random plan, resulting in purely online planning and control. These two are compared with the offline GPR planning and SR* planning. For the re-planning approaches, the number of iterations per step (IPS) was doubled until the real time constraint could not be achieved. This limit was resulted 160 IPS and 40 IPS for the single and double pendulum, respectively. Since SVR and GPR models could not achieve even 1 IPS for both problems, they are not applied to re-planning. The results are shown in Figure 6.

The performance of online and "re-planning of SR" approaches improve with increased IPS, significantly outperforming offline GPR and SR* ($p < 0.05$) with maximum

IPS (i.e., 160 for single and 40 for double pendulum). "Re-planning of SR" typically showed remarkable improvement over the purely online approach, resulting in the best performance among all tested approaches with maximum IPS. When the IPS is set to maximum, the total number of iterations required for "re-planning of SR" approach was 96500 iterations and 48500 iterations for single and double pendulum problems, respectively. While offline SR* consumed 500000 iterations, its performance is poorer than that of "re-planning of SR".

In summary, the benefits of SR modeling over GPR and SVR modeling are two-fold: first, it provides more accurate prediction that is vital to planning; and second, the reduced computational effort makes efficient real-time applications possible.

Conclusion and Future Work

In this paper, an SR-based method was proposed to model a robot's dynamics autonomously and the inferred model was used for motion planning and control tasks. The model is represented as a mathematical expression that accurately explains the robot's dynamics and allows for fast computation. These features enable fast motion planning and real-time control based on re-planning, resulting in significantly superior control performance over GPR and SVR-based methods.

Possible applications of our method include forward modeling and motion planning for the locomotion of legged robots or soft robots. Soft robots are an area of particular interest since deriving a dynamic model is a difficult and daunting task. However, forward models of these robots are considered significantly more complex than of pendulums. Since the difficulty of model inference via SR is known to increase dramatically as the complexity of the true model increases (Schmidt and Lipson, 2008), we will need to develop novel techniques to infer models of more complex dynamics.

This paper is an initial investigation of applying SR for robotic modeling and there is room for further optimization. Although this work used a simplistic hill-climbing heuristic approach for motion planning, more complex and superior algorithms can be applied instead. It will also be possible to use SR to infer inverse models and utilize them in feedback control. We believe that the coexistence of accuracy and efficiency in SR models will help a novel class of algorithms in robotics to emerge.

Acknowledgements

The authors thank Daniel Ly of the Creative Machines Laboratory at Cornell University for his invaluable comments on the present paper. This work was supported in part by NIH NIDA grant RC2 DA028981, NSF CDI Grant ECCS 0941561. The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of the sponsoring organizations. Hirotaka Moriguchi is supported by a scholarship from Funai Foundation for Information Technology.

References

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.

Chang, C. and Lin, C. (2001). *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Coumans, E. (2010). *Bullet 2.76 Physics SDK Manual*. Software available at http://bulletphysics.org/.

Dearden, A. and Demiris, Y. (2005). Learning forward models for robots. In *Proceedings of IJCAI*, pages 1440–1445.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection (Complex Adaptive Systems)*. The MIT Press.

Nguyen-Tuong, D. and Peters, J. (2008). Local gaussian process regression for real-time model-based robot control. In *Proceedings of IROS*, pages 380–385.

Schmidt, M. and Lipson, H. (2008). Coevolution of fitness predictors. *IEEE Transaction on Evolutionary Computation*, 12(6):736–749.

Schmidt, M. and Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85.

Sturm, J., Plagemann, C., and Burgard, W. (2008). Unsupervised body scheme learning through self-perception. In *Proceedings of ICRA*, pages 3328–3333.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: an introduction*. The MIT Press.

Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.