Control in the Reliable Region of a Statistical Model with Gaussian Process Regression

Youngmok Yun and Ashish D. Deshpande

***Abstract*— We present a novel statistical model-based control algorithm, called Control in the Reliable Region of a Statistical Model (CRROS). A statistical model is unreliable when its state passes into a region where training data is sparse. CRROS drives the state away from such an unreliable region while pursuing the desired output by taking advantage of the redundancy in the input-output relationships. We validated the performance of CRROS by a simulation with a redundant manipulator and experiments with a robot. In the experiments, a manipulator called the Flex-finger, for which it is challenging to build an analytical model, is controlled to demonstrate the practical effectiveness of the proposed method.**

I. Introduction

Using a statistical model is an attractive alternative to using an analytical model for controlling a robotic system. For many systems (e.g., soft robots), it is difficult to build a suitable analytical model. Even if a suitable analytical model exists, determination of its parameters (e.g., mass matrix of manipulator) is another difficult problem. To address this, many researchers have devised statistical modeling methods to *learn* the system model based on input-output trials (i.e., training data set). Methods such as neural networks (NN) [1], support vector regression (SVR) [2], and Gaussian process regression (GRP) [3] are commonly employed for statistical modeling. A number of researchers have controlled robotic systems using models developed with statistical modeling methods [4].

However, a crucial weakness of the statistical modeling methods is that the reliability of statistical models is highly dependent on the training data. Even if a learning algorithm successfully works with a given training data, it is almost impossible to reliably predict the behavior of nonlinear systems when the robot state passes into a region where no training data is available or where training data is sparse. A reliable statistical model can be obtained only when the states are near a region where many training data points have been collected.

We consider a tracking control problem for the following system.

$$\begin{aligned} \mathbf{y} &= g(\mathbf{x}) \\ \dim(\mathbf{x}) &> \dim(\mathbf{y}) \end{aligned} \tag{1}$$

Y. Yun is a PhD student of Mechanical Engineering, The University of Texas at Austin, 3.130, ETC, Austin, Texas, USA yunyoungmok at utexas.edu

A.D. Deshpande is a faculty of Mechanical Engineering, The University of Texas at Austin, 3.130, ETC, Austin, Texas, USA Ashish at austin.utexas.edu

where $\mathbf{y}$ is the output, $\mathbf{x}$ is the controllable state, and g is a nonlinear system. Since $\dim(\mathbf{x}) > \dim(\mathbf{y})$, multiple $\mathbf{x}$ can work as solution for a given $\mathbf{y}$. A kinematically redundant manipulator with joint angles as state and end-effector position as output is a well-known example of such a system [5]. Another example is a human-like tendon-driven finger [4]. Our goal is to design a control algorithm to determine values of $\mathbf{x}$ that lead to tracking of a desired trajectory of $\mathbf{y}$ with a statistical model, denoted as f. Here, for successful control, the statistical model f should reliably represent g.

We present a statistical model-based control strategy for the system (1), called CRROS (Control in the reliable region of a statistical model). The underlying idea is to drive $\mathbf{x}$ toward an area of reliable f by regulating redundant DOFs while pursuing a desired output trajectory. With a statistical modeling method, unreliable f is inevitable in sparse data regions, but CRROS is able to avoid these unreliable regions. The first step of CRROS is to develop a statistical model with the GPR algorithm. Since GPR is fully based on the Bayesian theorem, it provides not only the predicted output for a given input but also the uncertainty of the prediction. Here the prediction uncertainty is important because it indicates which region is reliable. The next step in CRROS is to track a desired output $\mathbf{y}$ by using the pseudoinverse of the Jacobian of the GPR model, and simultaneously regulate the null space to drive $\mathbf{x}$ toward a region of low uncertainty.

A statistical model, which has different features compared with an analytic model, demands different control strategies. After the introduction of the GPR, few researchers have developed feedfoward control strategies with nonlinear model predictive control (NMPC) [6] to address the model reliability issues. However, because the combination of GPR with NMPC requires considerable computational burden, the applications have been restricted to only chemical process control [7][8]. In contrast, CRROS is an iterative-optimization-free algorithm, and thus has the potential for implementation in high frequency control loop or large scale problems.

II. System Modeling with Gaussian Process Regression

In this section, we will build a model for a nonlinear system with the GPR algorithm [3][9]. An advantage of the GPR is that the user does not need to actively tune the complexity and parameters of the model, in contrast to other nonlinear regression modeling methods including NN [1][10] and SVR [2]. Once the Gaussian Process (GP)

is defined with a mean and a covariance function, the detailed model is determined by the database itself. Another merit is that its prediction output is a probabilistic distribution (i.e., Gaussian distribution), and the variance of the distribution can be regarded as the level of prediction uncertainty. This uncertainty plays a key role in the development of CRROS.

Let us assume that we have a training data set D for the system (1), which contains inputs $\mathbf{x}_i$ and outputs y_i : $D = \{(\mathbf{x}_i, y_i)|i = 1, 2, \ldots, n\}$, $X = [\mathbf{x}_1 \quad \mathbf{x}_2 \ldots \mathbf{x}_n]^\top$, $\mathbf{y} = [y_1 \quad y_2 \ldots y_n]^\top$. Given this data set, we wish to predict the probability density of y_* for a new query point $\mathbf{x}_*$. In other words, we want to find $p(y_*|\mathbf{x}_*, D, \Theta)$, where Θ is a hyperparameter set for the GPR model. If the system (1) has a multidimensional output (MIMO system), its output is predicted by multiple functions, $p(y_*^j|\mathbf{x}_*, D^j, \Theta^j)$, where j is the index of the output. In this section, for simplicity, we only explain $p(y_*|\mathbf{x}_*, D, \Theta)$. The next section describes the expansion into a multidimensional output.

A Gaussian process is a stochastic process, and it is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. It is completely determined with a mean function and a covariance function as:

$$f(\mathbf{x}) \equiv \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{2}$$

where mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, and covariance function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$. In our GP model, the mean function and the covariance function are determined as:

$$m(\mathbf{x}) = 0 \tag{3}$$

$$k(\mathbf{x}, \mathbf{x}') = \nu_f^2 \exp\left\{-\frac{(\delta\mathbf{x})^\top \Lambda (\delta\mathbf{x})}{2}\right\} + \nu_n^2 \varphi(i, j) \tag{4}$$

where $\delta\mathbf{x}$ is defined as $\mathbf{x} - \mathbf{x}'$. $\varphi(i, j)$ is the Kronecker delta function. Λ is a diagonal matrix whose elements are (squared inverse) length scales for input space. ν_f^2 specifies the overall vertical scale of variation of the latent values, ν_n^2 the latent noise variance. The hyperparameter set, denoted as Θ, is defined as a set of ν_f, ν_n and Λ, and determines the characteristics of a GP model.

The Gaussian process prior is updated with a training data set based on the Bayes theorem. The posterior is given as (For the derivation, refer to [3]):

$$p(y_*|\mathbf{x}_*, D, \Theta) = \mathcal{N}(\mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{y}, \kappa - \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{k}_*) \tag{5}$$

where $\mathbf{K}$ is a matrix whose elements K_{ij} are covariance function values of $k(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_1) \ldots k(\mathbf{x}_*, \mathbf{x}_n)]^T$. $\kappa = k(\mathbf{x}_*, \mathbf{x}_*)$. Fig. 1 helps in understanding the useful feature of GPR. The posterior of GP provides large variance for sparse input data regions.

The characteristics of a GP model are determined by the choice of the hyperparameter set, so selection of an appropriate hyperparameter set is critical for a good representation of a system. We select a hyperparameter set which maximizes the log-likelihood $p(\mathbf{y}|\mathbf{X}, \Theta)$. For the detail, refer [3].

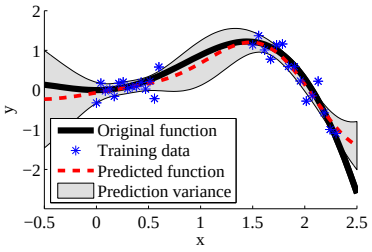


Fig. 1. An example of GPR. From a nonlinear function, $y = x^2\cos(x - 0.5)$, a training data set was sampled with noise. The Gaussian process posterior, updated with the training data, provides not only the predicted function but also its variance, which can be regarded as the level of prediction reliability. The prediction variance is large where training data is sparse or noisy.

III. Development of CRROS

As stated in Section I, a statistical model has different features compared to an analytical model, thus it demands a different control strategy. Specifically, if we apply a conventional control algorithm to a statistical model, it will output a path of $\mathbf{x}$ with no regard to the distribution of the training data. Since the reliability of the model is highly dependent on the density of data, this may result in failure of control. One example of this situation is presented in Section IV. To address this problem, which all the statistical models suffer from, we propose a novel control strategy, called CRROS that takes advantage of the system redundancy to drive the system states to reliable regions.

The basic idea of CRROS is to first track a desired output with the pseudoinverse of the Jacobian of the GPR model, then push the state of the system in a direction which minimizes the prediction uncertainty on the null space associated with the Jacobian. This method is inspired by the solutions to the obstacle avoidance problem in kinematically redundant robot [5]. In CRROS, the obstacle is the unreliable region of a statistical model. CRROS is able to achieve three goals: to track a desired output, to drive the state toward a region of low uncertainty, and to execute the algorithm at a high speed, which is essential for robot control. Since CRROS takes advantages of the Jacobian operation, which provides a continuous local optimum solution, it does not need any iterations, and thus allows for high speed control loop. Other possible algorithms for avoiding sparse data regions, such as ILC (iterative learning control) [11] or NMPC [6], involve iterative optimization procedure, and would require considerable computational time.

A. Preliminary work

To introduce CRROS, we first need to define a number of mathematical terms which will be used in the next subsection. The GP in (5) provides two values: predicted output and variance of the prediction, which can be rewritten with two functions. The first function, $f_\mu(\mathbf{x}_*)$ in (6), is for the predicted output at a query point $\mathbf{x}_*$, and the other function, $f_\Sigma(\mathbf{x}^*)$ in (7), is for the variance (uncertainty) of

the prediction.

$$f_\mu(\mathbf{x}_*) = \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{y} \tag{6}$$

$$f_\Sigma(\mathbf{x}_*) = \kappa - \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{k}_* \tag{7}$$

Next, their Jacobians are calculated as:

$$\begin{aligned} J_\mu(\mathbf{x}_*) &= \frac{\partial f_\mu(\mathbf{x}_*)}{\partial \mathbf{x}_*} \\ &= (\mathbf{K}^{-1}\mathbf{y})^\top \frac{\partial \mathbf{k}_*}{\partial \mathbf{x}_*} \end{aligned} \tag{8}$$

$$\begin{aligned} J_\Sigma(\mathbf{x}_*) &= \frac{\partial f_\Sigma(\mathbf{x}_*)}{\partial \mathbf{x}_*} \\ &= -2\mathbf{k}_*^\top \mathbf{K}^{-1} \frac{\partial \mathbf{k}_*}{\partial \mathbf{x}_*} \end{aligned} \tag{9}$$

The size of both matrixes J_μ and J_Σ are $1 \times d$ where $d = \dim(\mathbf{x})$. The term $\partial \mathbf{k}_* / \partial \mathbf{x}_*$, commonly used in (8) and (9), is defined as:

$$\frac{\partial \mathbf{k}_*}{\partial \mathbf{x}_*} = \begin{bmatrix} \frac{\partial(k(\mathbf{x}_*, \mathbf{x_1}))}{\partial \mathbf{x}_*[1]} & \cdots & \frac{\partial(k(\mathbf{x}_*, \mathbf{x_1}))}{\partial \mathbf{x}_*[d]} \\ \vdots & \ddots & \vdots \\ \frac{\partial(k(\mathbf{x}_*, \mathbf{x_n}))}{\partial \mathbf{x}_*[1]} & \cdots & \frac{\partial(k(\mathbf{x}_*, \mathbf{x_n}))}{\partial \mathbf{x}_*[d]} \end{bmatrix} \tag{10}$$

where $\mathbf{x}_*[i]$ indicates the i-th element in the vector $\mathbf{x}_*$. In our case, since the covariance function of GP uses the squared exponential kernel, each element is defined as:

$$\frac{\partial(k(\mathbf{x}_*, \mathbf{x}))}{\partial \mathbf{x}_*[i]} = -\Lambda_{ii}(\mathbf{x}_*[i] - \mathbf{x}[i])\nu_f^2 \exp\left\{-\frac{(\delta\mathbf{x})^\top \Lambda (\delta\mathbf{x})}{2}\right\} \tag{11}$$

So far, we have been limited to an one-dimensional output. Let us expand this problem into a multidimensional output problem, $\dim(\mathbf{y}) = q > 1$. We assume that we do not have any knowledge of relation between the outputs, a priori, thus the multi-dimensional output functions are simply stacked with the one dimensional functions. That is:

$$\mathbf{f}_\mu(\mathbf{x}_*) = [f_\mu^1(\mathbf{x}_*) \quad f_\mu^2(\mathbf{x}_*) \quad \dots \quad f_\mu^q(\mathbf{x}_*)]^\top \tag{12}$$

$$\mathbf{f}_\Sigma(\mathbf{x}_*) = [f_\Sigma^1(\mathbf{x}_*) \quad f_\Sigma^2(\mathbf{x}_*) \quad \dots \quad f_\Sigma^q(\mathbf{x}_*)]^\top \tag{13}$$

f_μ^i and f_Σ^i are for individual outputs, built in (6) and (7) . Both $\mathbf{f}_\mu$ and $\mathbf{f}_\Sigma$ are q-dimensional vectors. In the same way, $q \times d$ dimensional jacobian matrixes $\mathbf{J}_\mu$ and $\mathbf{J}_\Sigma$ are built as:

$$\mathbf{J}_\mu(\mathbf{x}_*) = \begin{bmatrix} J_\mu^1(\mathbf{x}_*) \\ J_\mu^2(\mathbf{x}_*) \\ \vdots \\ J_\mu^q(\mathbf{x}_*) \end{bmatrix} \tag{14}$$

$$\mathbf{J}_\Sigma(\mathbf{x}_*) = \begin{bmatrix} J_\Sigma^1(\mathbf{x}_*) \\ J_\Sigma^2(\mathbf{x}_*) \\ \vdots \\ J_\Sigma^q(\mathbf{x}_*) \end{bmatrix} \tag{15}$$

Since we want to solve a control problem, we are more interested in the sum of variances of multiple outputs rather than individual variances. Therefore, another function and its gradient are defined as:

$$f_\Xi(\mathbf{x}_*) = \sum_{i=1}^{q} f_\Sigma^i(\mathbf{x}_*) \tag{16}$$

$$\nabla f_\Xi(\mathbf{x}_*) = \left(\sum_{i=1}^{q} J_\Sigma^i(\mathbf{x}_*)\right)^\top \tag{17}$$

where f_Ξ is a scalar function providing the sum of prediction variances; the variances are already squared values of standard deviations, thus the sum operation can represent its total variance. $\nabla f_\Xi(\mathbf{x}_*)$ is a d-dimensional gradient vector which indicates the direction of the linearized slope of f_Ξ at $\mathbf{x}_*$ point.

B. Algorithm of CRROS

The system equation g in (1) is modeled with the GP mean function $\mathbf{f}_\mu$, and it can be linearized as (18). It can be rewritten as (19). In the view point of the feedback control, (19) can be rewritten as (20).

$$\Delta\mathbf{y} = \mathbf{J}_\mu \Delta\mathbf{x} \tag{18}$$

$$\Delta\mathbf{x} = \mathbf{J}_\mu^+ \Delta\mathbf{y} + (\mathbf{I}_d - \mathbf{J}_\mu^+ \mathbf{J}_\mu)\mathbf{v} \tag{19}$$

$$\tilde{\mathbf{x}}_{t+1} = \xi\left(\mathbf{J}_\mu^+(\mathbf{y}_{t+1} - \hat{\mathbf{y}}_t) + (\mathbf{I}_d - \mathbf{J}_\mu^+ \mathbf{J}_\mu)\mathbf{v}\right) + \hat{\mathbf{x}}_t \tag{20}$$

where $\tilde{\mathbf{x}}_{t+1}$ is the next target state to be controlled, $\hat{\mathbf{x}}_t$ is the observed current state. $\mathbf{y}_{t+1}$ is the next desired output. $\hat{\mathbf{y}}_t$ is the observed current output. From here, since we are only interested in the prediction at the point $\hat{\mathbf{x}}_t$, in other words $\mathbf{x}_* = \hat{\mathbf{x}}_t$, we will use $\mathbf{J}_\mu$ instead of $\mathbf{J}_\mu(\hat{\mathbf{x}}_t)$. Other notations will follow this omission rule as well. $\mathbf{J}_\mu$ is the Jacobian matrix for $\mathbf{f}_\mu$. $\mathbf{J}_\mu^+$ is the Moore Penrose pseudoinverse of $\mathbf{J}_\mu$ and ξ is a scalar gain. $(\mathbf{I}_d - \mathbf{J}_\mu^+ \mathbf{J}_\mu)$ is a matrix projecting a vector on the null space associated with $\mathbf{J}_\mu$ where $\mathbf{I}_d$ is a $d \times d$ identity matrix. $\mathbf{v}$ is an arbitrary d-dimensional vector. Here, the remarkable point is that we can select any $\mathbf{v}$ because the $\mathbf{v}$ is projected on the null space which does not affect $\Delta\mathbf{y}$, but $\mathbf{v}$ can affect $\Delta\mathbf{x}$. In many robotics applications, $\mathbf{v}$ is set to be a zero vector because it minimizes $||\tilde{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t||$, in other words, it is the minimum norm solution. In our case, $\mathbf{v}$ is set in a different way.

In CRROS, we wish to minimize the sum of prediction variances f_Ξ while tracking a desired path. To reduce f_Ξ, we need to consider a direction of $\Delta\mathbf{x}$ to minimize Δf_Ξ. The direction of $\Delta\mathbf{x}$ is defined with a direction vector $\mathbf{v}_d$:

$$\mathbf{v}_d = \underset{\mathbf{u}}{\operatorname{argmin}} \left(f_\Xi(\hat{\mathbf{x}}_t + \varepsilon\mathbf{u}) - f_\Xi(\hat{\mathbf{x}}_t)\right) = -\frac{\nabla f_\Xi}{||\nabla f_\Xi||} \tag{21}$$

where ε is a small positive real number. (21) is a widely used theorem in gradient-based optimization algorithms [12]. The change of f_Ξ, caused by $\mathbf{v}_d$, is defined with $\mathbf{v}_m$ as (22):

$$\mathbf{v}_m = f_\Xi(\hat{\mathbf{x}}_t + \mathbf{v}_d) - f_\Xi(\hat{\mathbf{x}}_t) \simeq \nabla f_\Xi^\top \mathbf{v}_d = -||\nabla f_\Xi|| \tag{22}$$

"$\simeq$" is caused by linearization, or the gradient operation. Now, let us build CRROS's $\mathbf{v}$ whose direction is $\mathbf{v}_d$ and magnitude is $\beta||\mathbf{v}_m||$. Using (21) and (22), $\mathbf{v} = -\beta\nabla f_\Xi$ where β is a control gain for reducing the prediction variance. $\mathbf{v}$ points toward a reliable region in the GPR statistical model, and its magnitude is proportional to the slope of f_Ξ. Finally, (20) is rewritten with the new $\mathbf{v}$ as:

$$\mathbf{x}_{t+1} = \xi\left(\mathbf{J}_\mu^+(\mathbf{y}_{t+1} - \hat{\mathbf{y}}_t) - \beta(\mathbf{I}_d - \mathbf{J}_\mu^+ \mathbf{J}_\mu)\nabla f_\Xi\right) + \hat{\mathbf{x}}_t \tag{23}$$

Because the primary task is to track a desired trajectory, $\mathbf{J}^+$ first uses a part of $\mathbf{x}$ space, and then the remaining space (i.e.,

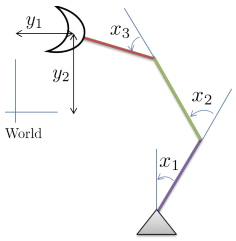


Fig. 2. The configuration of 3-DOF articulated manipulator which is used in the simulations. The x-y positions of the end-effector $\mathbf{y}$ are controlled by the three joint angles $\mathbf{x}$.

null space) is used for $-\beta\nabla f_{\Xi}$. Value of β determines how *strongly* the algorithm pushes the target state $\tilde{\mathbf{x}}_{t+1}$ toward the low uncertainty region. A large value of β increases $||\tilde{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t||$ because the null space is orthogonal to the space of $\mathbf{J}_{\mu}^{+}$.

IV. Results

A. Simulation of 3-DOF Articulated Manipulator

For validation of the algorithm, we conducted a simulation with a 3-DOF articulated manipulator. Fig. 2 demonstrates the configuration of the simulated 3-DOF articulated manipulator. The input $\mathbf{x}$ consists of three joint angles and the output $\mathbf{y}$ consists of the x-y positions of the end-effector. Based on the system configuration, the kinematics are governed by a nonlinear function g as shown in (24):

$$\mathbf{y} = g(\mathbf{x}) \tag{24}$$

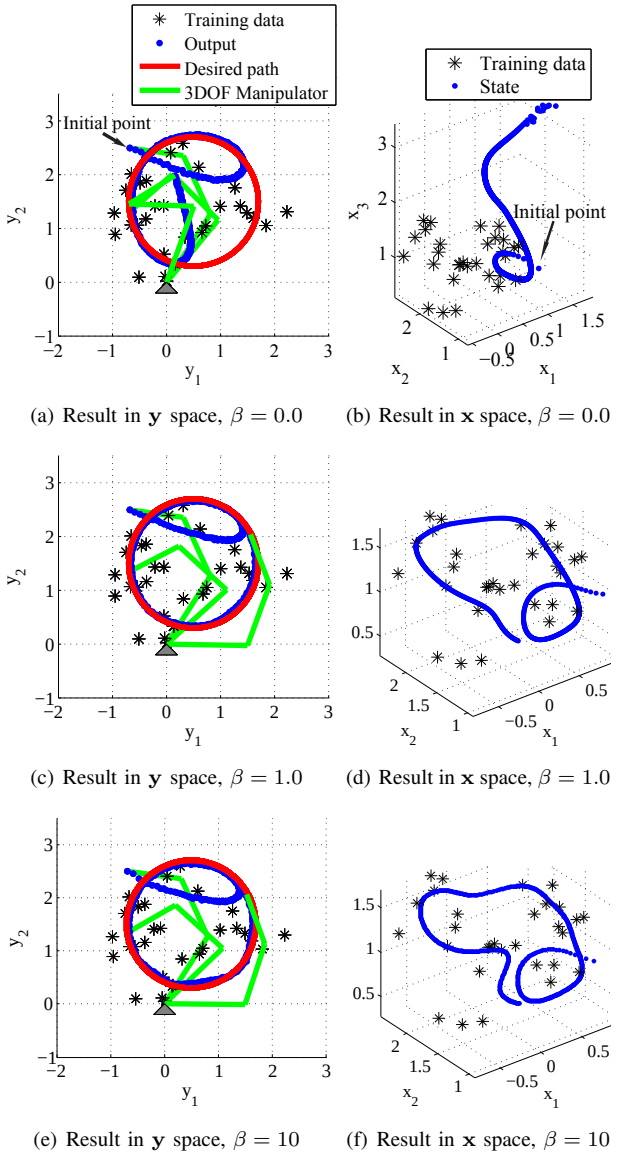
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} l_1\cos(x_1) + l_2\cos(x_1+x_2) + l_3\cos(x_1+x_2+x_3) \\ l_1\sin(x_1) + l_2\sin(x_1+x_2) + l_3\sin(x_1+x_2+x_3) \end{bmatrix}$$

where l_1,l_2, and l_3 are length of each link. ($\mathbf{y}$ in Fig. 2 has an offset for generality.)

The model of (24) is built with the GPR. For generating the training data collection, 30 input angles, $\mathbf{x}$, were randomly sampled within $\pi/2$ rad ranges. This range constraint has two purposes. The first is to simulate a realistic situation; most articulated arm actuators have a limited operation range of motion due to hardware contact and safety. The second reason is to create a sparse region in the training data, in order to clearly show the effectiveness of CRROS. Corresponding training data outputs $\mathbf{y}$ were collected using (24).

Based on the GPR model constructed with the collected training data, CRROS controlled the manipulator to track a desired circular trajectory. The simulations were conducted with three different conditions, $\beta = 0.0, 1.0$ and 10, to investigate the effect of β. We assumed observation error and control error, $\hat{\mathbf{x}} = \mathbf{x} + \epsilon_x$, $\hat{\mathbf{y}} = \mathbf{y} + \epsilon_y$ and $\mathbf{x}_{t+1} = \tilde{\mathbf{x}}_{t+1} + \epsilon_c$ where ϵ is unbiased Guassian noise. This simulates a noisy environment for training data collection and trajectory tracking control. Table I specifies the parameters which were used in the simulation, and Fig. 3 and Fig. 4 illustrates the results.

In the first case $\beta = 0$, as shown in Fig. 3 (a), (b) and Fig. 4, the algorithm failed to track a desired trajectory. With $\beta = 0$, CRROS does not consider the distribution of training



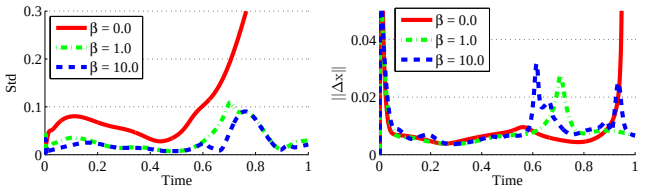
(a) Result in $\mathbf{y}$ space, $\beta = 0.0$ (b) Result in $\mathbf{x}$ space, $\beta = 0.0$

(c) Result in $\mathbf{y}$ space, $\beta = 1.0$ (d) Result in $\mathbf{x}$ space, $\beta = 1.0$

(e) Result in $\mathbf{y}$ space, $\beta = 10$ (f) Result in $\mathbf{x}$ space, $\beta = 10$

Fig. 3. Simulation results of 3-DOF articulated manipulator control. Experiments were conducted with three different conditions, $\beta = 0.0, 1.0$ and 10. Each row shows the results for different β values. The analysis were performed in $\mathbf{y}$ space (left column) and $\mathbf{x}$ space (right column). If $\beta = 0$, CRROS provides the minimum norm solution, and does not consider the distribution of training data. As a result, it fails to track a desired trajectory. In the second and third cases ($\beta = 1.0, 10$), CRROS successfully tracked a desired trajectory. In $\mathbf{y}$ space, their results (c)(e) look similar, but the state path in $\mathbf{x}$ space are significantly different. The path in (f) tends to be closer to training data points than (d), thus the trajectory has several sharp curvatures. This is caused by a high gain β pushing strongly the state toward a reliable region.

data, and simply outputs the minimum norm solution. As a result, the state entered the region with sparse training data, and this significantly deteriorated the control performance. In the second and third cases, CRROS successfully tracked the desired path, and simultaneously stayed in low uncertainty region. This verifies that the algorithm successfully *pushed* the state toward a reliable region by projecting ∇f_{Ξ} on a null space. Because the low uncertainty region exists around a training data point, the trajectory of state space tended to



(a) Prediction uncertainty (b) $||\tilde{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t||$

Fig. 4. Simulation results of 3-DOF articulated manipulator control. (a) shows the prediction uncertainty by standard deviation. It is expressed as $\sqrt{f_\Xi}$. Because CRROS in the case $\beta = 0$ does not consider prediction uncertainty, the prediction uncertainty dramatically increased. After all, this unreliable model made the control failed. The case $\beta = 10$ had a slightly smaller prediction uncertainties during the control time than the case $\beta = 1.0$. (b) shows the norm of $\tilde{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t$. Initially the first case had the smallest value because it provides the minimum norm solution. However, at the end, large error in $\mathbf{y}$ space sharply increased the value. For the third case, CRROS strongly pushed the state to reduce the prediction uncertainty, and as a result, several peaks appear in the plot.

TABLE I

THE PARAMETERS USED IN THE SIMULATION

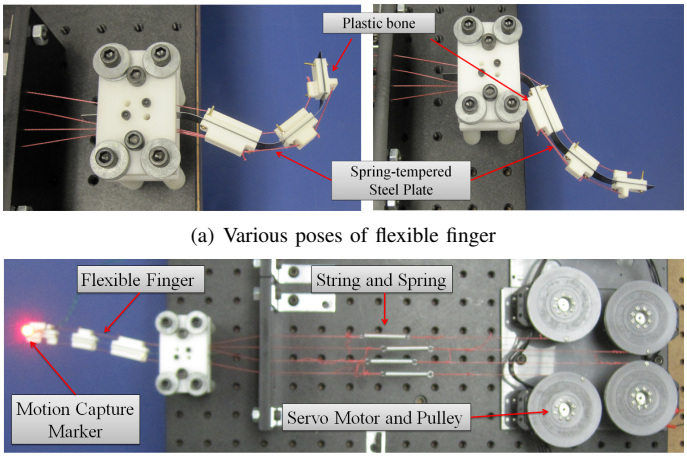
l_1	l_2	l_3	Std (ϵ_x)	Std (ϵ_y)	Std (ϵ_c)	ξ	β
1.5	1.2	1.0	0.02 rad	0.01	0.02 rad	0.05	0,1,10

be close to those points. This effect was more significant for the third case because in this case CRROS pushed the state strongly toward the low uncertainty region. This resulted in several sharp turns in the state space (Fig. 3 (f)) and longer travel distances (Fig. 4 (b)). The uncertainty in this case is the lowest during the control time as shown in Fig. 4 (a).

B. Experiment with Flex-finger

In the previous subsection, we presented simulation results of a 3-DOF articulated manipulator system to validate our methodology. Of course, for such a system the control engineers could also use an analytical model. However, there are a number of nonlinear systems whose models cannot be represented analytically. Additionally, many a times robot designers stay away from optimal designs only to ensure that analytical models can be developed for their systems. A significant motivation of CRROS is to provide control schemes for a wide range of systems. To demonstrate the effectiveness of our method, we selected a robotic system whose analytical model is difficult to derive.

The robot is called the Flex-finger which is designed to study various features of the human finger (Fig. 5). The robot is a highly nonlinear system. Its main body is made of a spring-tempered steel plate and plastic bones, which act as phalanges and joints, but do not have a deterministic rotational center. Four compliant cables (corresponding to the human tendons), consisting of strings and springs, connect the plastic bones to the motors in order to generate flexion/extension motion. Four servo motors (corresponding to human muscles) change the cable lengths, which leads to a change in the pose of the Flex-finger. The pose of the fingertip is observed by a motion capture system (corresponding to the human eye).



(a) Various poses of flexible finger

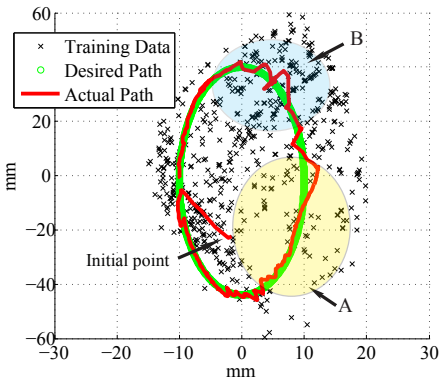
(b) Overview of Flex-finger

Fig. 5. A manipulator called the Flex-finger was used to validate the performance of CRROS. The Flex-finger consists of a spring-tempered steel plate, plastic bone, string, spring, servo motor and pulley. Compliant elements and nonlinear tendon configuration make it difficult to build an analytical model.

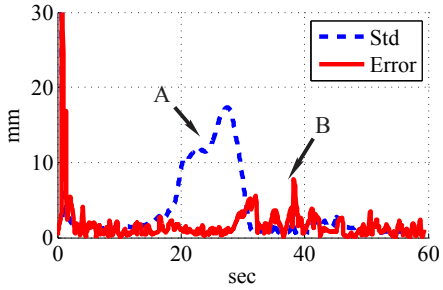
The objective in this experiment is to control the tip of the Flex-finger to track a desired path, which is similar to the control of the human finger with muscles. In this case, going back to (1), $\mathbf{x}$ consists of the rotational angles of servo motors and $\mathbf{y}$ consists of the x-y coordinates of fingertip position. This is a challenging control problem. First, the kinematics of the system are highly nonlinear. Particularly, there are several bifurcation points due to the characteristics of spring-tempered steel plate. This means that for a small change in $\mathbf{x}$, $\mathbf{y}$ can dramatically diverge. Second, the available operation range of $\mathbf{x}$ is quite complex. The servo motor always needs to generate a minimum load on the cable to prevent cables from slacking, which severely affects the available operation range. The problem is that the operation range of each servo motor is not fixed but is dependent on the configuration of other cables. Because of these reasons, it is impossible to generate an analytical model to represent this system.

For collecting the training data, 500 motor angle values, or $\mathbf{x}$, were supplied and corresponding finger positions, or $\mathbf{y}$, were recorded. The input values were randomly selected from within the available operation range. A GPR model was built based on the training data. CRROS algorithm was implemented in C++ (with RTAI Linux) and executed in a desktop with Intel Xeon 3.1 GHz CPU and 8 GB RAM. In the program, the control loop ran at 30 Hz. The loop frequency was mainly determined by the communication speed with servo motors, not by the speed of CRROS. The execution time of CRROS was less than 10 ms. The tracking experiment results are shown in Fig. 6 and in the video [13].

As Fig. 6 (a) and (b) illustrate, overall CRROS successfully tracked the desired path with reasonably low errors. As soon as the CRROS algorithm starts, the output immediately converged into the desired trajectory and which led to reduction in prediction uncertainty. The tracking was good



(a) Results from control experiments with the Flex-finger (in output space)



(b) Prediction uncertainty and error ($||y_{des} - y||$)

Fig. 6. For the validation of CRROS, we controlled the Flex-finger to track a desired trajectory. (a) and (b) illustrate that successful control was achieved with reasonably low error. As soon as the control starts, CRROS reduced both the tracking error and the prediction uncertainty. Because area A is a sparse training data region, the prediction variance becomes higher even though CRROS worked to minimize it. In area B, friction led to jerky motions, resulting in peaks in error plot. The video [13] shows the detail of the Flex-finger experiments.

until the finger tip reached area A. CRROS tried to minimize the prediction uncertainty, but because the area A is a region with sparse training data, the rise of prediction variance was inevitable, leading to a peak in error plot. Another interesting region is area B. As shown in Fig. 6, its prediction uncertainty is low because it is a region with dense training data. Nevertheless, the fingertip made several error peaks and exhibited jerky motions. Our analysis showed that in order to create the output in area B, high cable tensions are required, which caused friction and jerky motions.

V. Conclusion and Future Works

We have presented a statistical model-based control algorithm using GPR, called CRROS. CRROS drives the state toward a low uncertainty region while tracking a desired output trajectory. The algorithm is validated by simulation with a 3-DOF articulated manipulator and experiments with the Flex-finger, a robotic system which is difficult to model analytically. The simulation and experiments with the Flex-finger demonstrates the potential of the novel method. We are able to control this highly nonlinear system using the statistical model developed with GPR, while keeping the states in the reliable region. Moreover, since CRROS takes advantages of the Jacobian operation which provides a continuous local optimum solution, it does not need any iteration, and thus allows for high speed control loop, which is essential in many robotics applications.

There are a number of ways for improving the presented method. The first one is the development of a more systematic data collection method. Because the performance of the statistical model-based control is highly dependent on the distribution of a training data set, a better strategy of data collection would lead to better control performance. Another important issue is heteroscedastic noise, or input-dependent noise [14]. Uncertainty of prediction is caused by two sources: sparsity of data and measurement noise. CRROS is using a standard GP algorithm which assumes a consistent noise level. The improved CRROS algorithm with a modified GP which can cover the heteroscedastic noise is a part of our ongoing works. Lastly, connecting with reinforcement learning and policy searching algorithms [15][16] might lead a better statistical model-based control algorithm.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.

[2] A. Smola and B. Schlkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[4] A. D. Deshpande, J. Ko, D. Fox, and Y. Matsuoka, "Control strategies for the index finger of a tendon-driven hand," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 115–128, 2013.

[5] C. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-13, no. 2, pp. 245–250, 1983.

[6] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright, "Nonlinear predictive control and moving horizon estimationan introductory overview," in *Advances in control*, pp. 391–449, Springer, 1999.

[7] B. Likar and J. Kocijan, "Predictive control of a gasliquid separation plant based on a gaussian process model," *Computers and Chemical Engineering*, vol. 31, no. 3, pp. 142 – 152, 2007.

[8] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the American Control Conference*, vol. 3, pp. 2214–2219, 2004.

[9] G. Gregori and G. Lightbody, "Gaussian process approach for modelling of nonlinear systems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 45, pp. 522 – 533, 2009.

[10] D. F. Specht, "A general regression neural network," *Neural Networks, IEEE Transactions on*, vol. 2, no. 6, pp. 568–576, 1991.

[11] S. Mondal, Y. Yun, and W. K. Chung, "Terminal iterative learning control for calibrating systematic odometry errors in mobile robots," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 311–316, 2010.

[12] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical analysis*, pp. 144–157, Springer, 1978.

[13] Y. Yun and A. D. Deshpande, "Experimental result video," *http://youtu.be/mptKKNfLcLE*, 2014.

[14] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic gaussian process regression," in *Proceedings of the 24th international conference on Machine learning*, pp. 393–400, 2007.

[15] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472, 2011.

[16] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.