

Learning Accurate Kinematic Control of Cable-Driven Surgical Robots Using Data Cleaning and Gaussian Process Regression

Jeffrey Mahler¹, Sanjay Krishnan¹, Michael Laskey¹, Siddarth Sen¹, Adithyavairavan Murali¹, Ben Kehoe², Sachin Patil¹, Jiannan Wang¹, Mike Franklin¹, Pieter Abbeel¹, Ken Goldberg³

Abstract—Precise control of industrial automation systems with non-linear kinematics due to joint elasticity, variation in cable tensioning, or backlash is challenging; especially in systems that can only be controlled through an interface with an imprecise internal kinematic model. Cable-driven Robotic Surgical Assistants (RSAs) are one example of such an automation system, as they are designed for master-slave teleoperation. We consider the problem of learning a function to modify commands to the inaccurate control interface such that executing the modified command on the system results in a desired state. To achieve this, we must learn a mapping that accounts for the non-linearities in the kinematic chain that are not accounted for by the system’s internal model. Gaussian Process Regression (GPR) is a data-driven technique that can estimate this non-linear correction in a task-specific region of state space, but it is sensitive to corruption of training examples due to partial occlusion or lighting changes. In this paper, we extend the use of GPR to learn a non-linear correction for cable-driven surgical robots by using i) velocity as a feature in the regression and ii) removing corrupted training observations based on rotation limits and the magnitude of velocity. We evaluate this approach on the Raven II Surgical Robot on the task of grasping foam “damaged tissue” fragments, using the PhaseSpace LED-based motion capture system to track the Raven end-effector. Our main result is a reduction in the norm of the mean position error from 2.6 cm to 0.2 cm and the norm of the mean angular error from 20.6 degrees to 2.8 degrees when correcting commands for a set of held-out trajectories. We also use the learned mapping to achieve a 3.8× speedup over past results on the task of autonomous surgical debridement. Further information on this research, including data, code, photos, and video, is available at <http://rll.berkeley.edu/surgical>.

I. INTRODUCTION

Almost all industrial automation systems are affected by actuator imprecision, where the actual motion differs from desired motion. A variety of factors, such as joint elasticities, variations in cable tension, backlash, or wear on geartrains, can lead to imprecision. Compensating for imprecision is particularly challenging when the system can only be commanded through a control interface with an inaccurate internal model, as is common for proprietary systems. Semi-autonomous surgery with cable-driven Robotic Surgical Assistants (RSAs), which are designed for master-

¹Department of Electrical Engineering and Computer Sciences; {jmahler, sanjaykrishnan, laskeymd, siddarthsen, adithya_murali, sachinpatil, jnwang, franklin, pabbeel}@berkeley.edu

²Department of Mechanical Engineering; benk@berkeley.edu

³Department of Industrial Engineering and Operations Research and Department of Electrical Engineering and Computer Sciences; goldberg@berkeley.edu

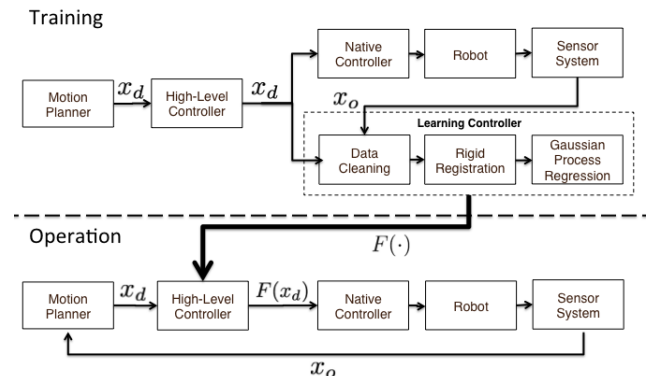


Fig. 1. System Architecture. In the training phase, we collect pairs of desired states x_d , which are left unmodified by the high-level controller, and observed states after execution x_o using the robot’s imprecise kinematics model and motion capture. In the learning controller, we remove corrupted data with data cleaning, and then learn F , a mapping that minimizes the difference between x_d and the observed x_o after sending command $F(x_d)$ to the robot. We model this function as a composition of a rigid and a non-linear transformation, which are learned with a constrained least squares and Gaussian Process Regression respectively. During operation, the user-operated high-level controller applies this learned function to the desired state and commands the system’s native controller to go to $F(x_d)$. The precise motion capture system is no longer necessary during this phase of operation.

slave teleoperation and have non-linearities due to cable elasticity and tensioning, is one example of a scenario with these challenges [12].

To achieve precision in this setting, we must learn how to command the robot such that the resulting state of the tool after execution matches a desired state. While we may not have access to the internal system parameters, we often have access to correlated control inputs and observations of the state from an external sensor, which we can use to learn the residual error between commands and observations. Although the space of possible states is very large, for a specific automation task it may suffice to learn a correction only over the states relevant to that task. Pastor et al. proposed a combination of a rigid transformation and Gaussian Process Regression (GPR) to learn an error correction function in a task-specific region of state space, demonstrating that this task-specific correction could be used to control the cable-driven DARPA ARM with an position accuracy of 2-4 mm for grasping and manipulation tasks [20]. We extend previous work on GPR for learning a residual error correction function for cable-driven surgical robots by i) including velocity as a feature in GPR, which can influence non-linearities at high speeds of operation, and ii) studying the effect of removing

corrupted observations from our training data, which can occur due to occlusions or lighting changes in the workspace. We call our three step procedure of data cleaning, rigid transformation, and GPR the “learning controller,” illustrated in Fig. 1.

In addition to cable-driven surgical robots, this method has the potential to improve control of other non-linear automation systems that have complicated non-linear kinematics, an incomplete kinematics model, or a control interface that uses an inaccurate kinematics model. Examples of such systems include the ABB IRB 6600, a low-cost industrial robot with dynamic coupling effects and flexible joints [31], the Baxter robot, the Barrett arm, the Ekso Bionix ProStep[®], and cable-driven 7-DOF humanoid arms [3].

We evaluate our approach on the Raven II Surgical Robot, an open source hardware platform for research on RSAs [6]. To collect training data for our model, we use PhaseSpace LED-based motion capture to record the trajectory of the end-effector using four cameras [1]. The PhaseSpace cameras can capture at 480 Hz and localize LEDs to within 0.1 mm, but still result in corrupted observations when the LEDs become occluded in one or more of the cameras.

We train the learning controller on recorded trajectories of our two-arm Raven surgical robot system as it autonomously executes surgical debridement following the setup of Kehoe et al., where the goal is to find, grasp, and transport “damaged tissue” fragments [12]. Our results show that including velocity as a feature in GPR reduces the norm of the position error by 50% and the norm of the angular error by 19% on these recorded trajectories. Furthermore, the addition of data cleaning reduces the norm position error by an additional 4% and the norm angular error by an additional 21% over the best result without any data cleaning. We also evaluate the accuracy and repeatability of reaching desired states using our correction method to modify commands sent to the robots and executing the commands in open-loop. Using this correction function, we achieve a $3.8\times$ speedup over previous work on autonomous surgical debridement [12].

II. RELATED WORK

One method to control cable-driven manipulators is to directly model and estimate the non-linear parameters. Estimating the kinematic parameters of serial-link robot manipulators has been well studied in the calibration literature. See [9] for an overview. Wernholt et al. learned non-linear parameters through regression, dividing up the states of the non-linear model into locally linearly regions and compute a transfer function for each area. [32]. Chen et al. propose a two-level self calibration method for a 7-DOF cable-driven humanoid arm based on iterative linearization and updates of error in state with respect to kinematic parameters [3]. In [15], Krause et al. design feed-forward controllers based on stiffness models of the cables. Naerum et al. considered both offline and online parameter estimation using the Unscented Kalman Filter [19] with an explicit model of a cable-driven 1-DOF system with motor angle measurements. This work was extended to the 7-DOF Raven Surgical robot in

simulation, but the accuracy was sensitive to hand-tuned process noise estimates [24].

Cable-driven robots can also be controlled without explicit parameterization of non-linearities. Abdhollu et al. used a neural network to learn the dynamics of non-linear systems and demonstrated the approach on elastic joint systems with motor angle and velocity measurements [2]. In [33], Williams et al. develop a slack-free controller by ensuring that cable tension is positive for all motion, but the authors assume known constants for cable elasticity. Reiter et al. track the error between the pose from the forward kinematics and the pose from keypoint-based visual tracking in an Extended Kalman Filter to precisely estimate the pose of the cable-driven Intuitive Surgical da Vinci[®] [27]. While this allows for an incomplete kinematics model, the linearization may cause divergence when the error dynamics are inaccurate or approximated poorly. Surgical robot control has also been improved by using inertial measurement units for gravity compensation [17].

Past work in system identification has studied regression with a Gaussian Radial Basis Function (RBF) kernel in the context of controlling nonlinear systems without explicit kinematic parameterizations [10], [11]. Gaussian Process Regression has also been used to learn dynamics models without a precise kinematics model [29], [14], [13]. However, these works assume that the robot can be commanded in an accurate reference frame, whereas our work can handle systems that have an inaccessible native control interface with an inaccurate kinematics model. Pastor et al. use a two-stage system consisting of a rigid transformation and Gaussian Process Regression to model the state-dependent relationship between robot desired poses and camera-observed poses on cable-driven robots [20]. In our work, we extend this method by including velocity as a feature in GPR and using data cleaning to improve mean accuracy on the Raven II Surgical Robot to 1 mm.

Our work is also related to prior work in data cleaning and outlier rejection, which has been shown to improve accuracy of statistical models; for example, it can reduce bias in aggregate query processing for large datasets [30]. For general statistical models, Random Sample Consensus (RANSAC) has been extensively studied to remove high-magnitude outliers [4]. Various extensions to the RANSAC model have been proposed to cope with tuning: Least of Medians [18] and Residual Consensus [22], which leverages the fact that the test error should have small variance over the true inlier set. In addition, there are adaptive techniques that allow for early stopping if a good enough model has been found [23]. Expectation Propagation (EP) has been proposed to make Gaussian Process Regression robust to outliers by using a mixture model posterior for the regressor consisting of separate components for inliers and outliers [16]. This approach requires a Gaussian Process prior on outliers whereas our approach is free of such a prior. Pearson studied the effects of outliers in system identification [21], stressing the importance of data cleaning for proper identification. Pearson considers impulse responses of single-input single-

output systems, and argues that using a specialized median filter (called the Hampel Filter) works well empirically to remove outliers in many datasets. Our model extends Pearson’s insights on data cleaning to the multi-dimensional case using a similar median filter (Least of Medians variant of RANSAC), and additionally basing further data cleaning on the physical properties of the system and workspace.

III. PROBLEM DEFINITION

We consider a cable driven robot equipped with a **native controller** that uses an imperfect internal kinematics model and encoder values to convert state commands to physical voltages on the robot, and a **sensor system** that observes the state of the robot, as illustrated in Fig. 1. Our poses are defined with respect to the frame of the native controller. We adopt the following variable naming conventions throughout this paper:

- t_x, t_y , and t_z are the translations in the x, y , and z directions with respect to the global coordinate frame
- ϕ_y, ϕ_p , and ϕ_r are the rotations about the yaw, pitch, and roll axes, respectively, with respect to the global coordinate frame
- $\mathbf{p} = (t_x, t_y, t_z, \phi_y, \phi_p, \phi_r)$ is the pose of the robot
- $\dot{\mathbf{p}}$ is the derivative of the pose
- $\mathbf{x} = (\mathbf{p}, \dot{\mathbf{p}})$ is the state of the robot, with subscripts $\mathbf{x}_c, \mathbf{x}_o$, and \mathbf{x}_d to denote the commanded, observed, and desired states, respectively
- $R(\mathbf{p}) \in \mathbb{R}^{3 \times 3}$ is the rotation matrix corresponding to the yaw, pitch, and roll rotations of \mathbf{p}
- $\mathbf{t}(\mathbf{p}) \in \mathbb{R}^{3 \times 1}$ is the translation of \mathbf{p}
- $T(\mathbf{p}) = \begin{pmatrix} R(\mathbf{p}) & \mathbf{t}(\mathbf{p}) \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{4 \times 4}$ is the rigid transformation matrix corresponding to \mathbf{p}
- $\mathcal{T} = \{\mathbf{x}_1, \dots, \mathbf{x}_t\}$ denotes a trajectory of length t
- $\mathcal{X} = \{(\mathbf{x}_{c,1}, \mathbf{x}_{d,1}), \dots, (\mathbf{x}_{c,M}, \mathbf{x}_{d,M})\}$ denotes the set of M pairs of corresponding observed and desired training states
- $\mathcal{Y} = \{(\mathbf{x}_{c,1}, \mathbf{x}_{d,1}), \dots, (\mathbf{x}_{c,N}, \mathbf{x}_{d,N})\}$ denotes the set of N pairs of corresponding observed and desired testing states

Our primary goal is to learn how to command the native controller such that observed states \mathbf{x}_o , closely match the desired states \mathbf{x}_d . As illustrated in Fig. 1, we augment the system with a **learning controller** that learns a function F to transform desired states before sending them to the native controller. We consider a variant of the approach of [20] to learn the function F . Specifically, we constrain F to be the composition of two functions: a rigid transformation G and a non-linear function H , so that $F(\mathbf{x}) = H(G(\mathbf{x}))$. We learn F by minimizing the difference between the transformed desired state $F(\mathbf{x}_d)$ and the state command \mathbf{x}_c that generated the corresponding observed state \mathbf{x}_o on the robot. In summary, our formal goal is to find some function $F : \mathbb{R}^{12} \rightarrow \mathbb{R}^{12}$ such that $\|F(\mathbf{x}_d) - \mathbf{x}_c\|_2$ is minimized over our test data set \mathcal{Y} given the constraints on F .

For data cleaning, we model corruption of the training and test data as sparse noise that is not physically realizable by the robot. We estimate the rigid transformation

using orthonormally constrained least squares. Following the approach of [20], we estimate the non-linear component using Gaussian Process Regression (GPR). We review these regression methods in Section IV to highlight details specific to the control setting.

We assume a surjective mapping between the state commanded to the robot and the observed states after executing this command in our analysis. We note that this is not true for overactuated robots and automation systems, but for these systems our method can be applied on joint angles instead of the end-effector pose. We also acknowledge that this technique will be only as good as the quality of readings from the external sensors.

IV. METHOD

Given a training set \mathcal{X} and a test set \mathcal{Y} , our method for estimating the function F relating observed states \mathbf{x}_o to desired states \mathbf{x}_d consists of three consecutive stages performed offline:

- 1) Data Cleaning
- 2) Estimation of Rigid Transformation
- 3) Gaussian Process Regression

Our method requires training and test sets of time-synchronized pairs of commanded and observed robot states. Details of our specific data collection procedure can be found in Section V.

A. Data Cleaning and Outlier Rejection

The output from the PhaseSpace motion capture system can be contaminated by outliers, ie. examples that significantly disagree with our model. We found that these outliers are largely caused by full or partial occlusions of the LED markers. In some parts of the workspace, the LEDs are only visible in two or three of the cameras, resulting in poor conditioning of the localization. “Dirty” data occurs when the LED markers move into an occluded or unoccluded region for one or more cameras because the conditioning of the localization changes and the markers “jump” to a new location. This results in artificially high velocity estimates. PhaseSpace does not grant access to the appropriate low-level information, such as the unfiltered data from each camera, which would allow us to detect these problems in real-time.

Corruption is not always in the form of outliers, and sometimes examples that lie close to the mean of the model, or “inliers”, may actually come from sequences of states that are physically impossible, such as states outside of the joint limits of the robot or states in which the velocity is higher than the maximum possible on the system. Data corruption can bias our learned model if the corruption is correlated with one of our features, e.g., some parts of the state-space are more likely to have outliers. We handle dirty data by first removing outliers using Least of Medians (LMEDS), a variant of RANSAC, and then removing potentially corrupted inliers using thresholds on the rotation, position, and velocity based on the physical limitations of the robot.

RANSAC has been extensively used for fitting statistical models in the presence of high magnitude outliers. Classical

RANSAC is often challenging to tune, as it has two hyperparameters: a distance threshold for classifying inliers and minimum number of consensus points. The first hyperparameter is particularly difficult to select in our setting because the distance is in an abstract metric space which includes both translation and rotation. Least of Median Squares (LMEDS) [18] and Residual Consensus [22] have been proposed as parameter-free variants of RANSAC, and we found that the LMEDS method gave us the most accurate final model on testing data without tuning.

While LMEDS gives us a way to reject outlier training examples, it does not address inliers that are potentially corrupted. It also ignores the time-series structure of the data and processes each training example independently. To address this problem, we can incorporate additional knowledge about physical process and the workspace. We designed an additional *cleaning* method, which runs in conjunction with LMEDS, that incorporates trajectory and workspace information to reject sequences of examples that were not physically realizable by the robot. We analyzed a dataset of observed robot states from the motion capture system and set angle based thresholds where the markers would not be visible. We also set thresholds to reject data points which corresponded to velocity magnitudes larger than what the robot could physically execute given the speed of operation.

Finally, we counted the number of rejections within a trajectory and if more than a threshold $P\%$ of its points were rejected, we rejected the entire trajectory. We found $P = 90\%$ worked well empirically. In our experiments, we show how our cleaning results in reduced mean absolute test error and also reduces the standard deviation of these errors.

B. Estimation of Rigid Transformation

To reduce the linear component of the error, we find the rigid transformation that minimizes the sum of squared errors for the training set (\mathcal{X}). This transformation serves as a prior on the source of the kinematic error; we expect that a portion of this error is due to camera registration error, poor encoder calibration, etc. A rigid transformation is composed of an orthonormal rotation matrix $R \in \mathbb{R}^{3 \times 3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^{3 \times 1}$. We minimize the error with respect to the Frobenius norm:

$$R^*, \mathbf{t}^* = \operatorname{argmin}_{R^T R = I, \mathbf{t}} \sum_{i=1}^N \left\| \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} T(\mathbf{p}_{i,d}) - T(\mathbf{p}_{i,c}) \right\|_F^2.$$

This objective can be solved in closed form using the Singular Value Decomposition, the accepted method of solving the linear transformation between two rigid bodies in the Computer Vision community [8].

C. Gaussian Process Regression

Gaussian Process Regression (GPR) is a Bayesian non-linear function learning technique that models a sequence of observations as generated by a Gaussian process. GPR predicts a parametric mean and variance for test points conditioned on the training points, which helps to characterize the quality of calibration for points not encountered

in the training set. The predictive distribution for other non-parametric learning techniques, such as neural networks, is not as well-characterized.

We first apply the rigid transformation $G(\cdot)$, and then fit the transformed observations to a Gaussian process. A key parameter to the problem is the kernel function, a measure of similarity between two training examples. We apply GPR with a kernel (called the Radial Basis Function, or RBF) of the following form:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2l^2}}$$

where σ denotes the signal variance and l denotes the characteristic length scale for the training data. GPR also typically involves a regularization constant β to model noise in the output measurements. We note that the RBF kernel is not necessarily appropriate for rotation angles because the space of angles is circular rather than Gaussian, but we find that it works well empirically. We estimate the parameters of this model using the GPML Toolbox [26]. See [25] for a comprehensive description of GPR, including how to estimate these parameters.

GPR learns a map from the input observed states \mathbf{x}_o to the output poses of the desired states \mathbf{p}_d , since we cannot directly command the velocity of the robot. We directly include the velocity of \mathbf{x}_o in the features of GPR. Thus, given the test set (X) and training set \mathcal{Y} we perform kernelized regression to predict output dimension i :

$$\begin{aligned} K_{\mathcal{X}, \mathcal{Y}} &= \begin{pmatrix} k(\mathbf{x}_1, \mathbf{y}_1) & \dots & k(\mathbf{x}_1, \mathbf{y}_M) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{y}_1) & \dots & k(\mathbf{x}_N, \mathbf{y}_M) \end{pmatrix} \quad x \in \mathcal{X}, y \in \mathcal{Y} \\ \mu_i &= K_{\mathcal{X}, \mathcal{Y}}^T (K_{\mathcal{X}, \mathcal{X}} + \beta^{-1}I)^{-1} \mathbf{z}_i \\ \Sigma_i &= K_{\mathcal{Y}, \mathcal{Y}} - K_{\mathcal{X}, \mathcal{Y}}^T (K_{\mathcal{X}, \mathcal{X}} + \beta^{-1}I)^{-1} K_{\mathcal{X}, \mathcal{Y}}. \end{aligned}$$

where $\mathbf{z}_i = (\mathbf{x}_{d,1}^{(i)}, \dots, \mathbf{x}_{d,M}^{(i)})^T$ is the vector of the i -th component of all training outputs. Also, K is the kernel matrix, and μ_i and Σ_i are the mean and variance of the prediction for the i -th component, respectively.

From these equations we see that GPR is able to generalize to parts of the workspace that were not observed in the training set by a non-linear interpolation between the function values at the training points. This is why corrupted training points can lead to errors when making predictions on new data, especially when a test point is “close to” the erroneous training point with respect to the kernel function.

V. EXPERIMENTS

A. Experimental Setup

We tracked the motion of the Raven II with a four camera PhaseSpace Impluse X2 motion capture system [1]. The four cameras were placed in an arc of approximately 120° at a distance of approximately 1.5 m to the Raven tool, oriented towards the instrument. We mounted three LEDs to the Raven tool to measure the pose and velocity (Fig. 3). The Raven performed a surgical debridement task, in which the goal is to find, grasp, and transport “damaged tissue” fragments [12], and we represented the tissue fragments with pieces of red foam. The locations of the red foam

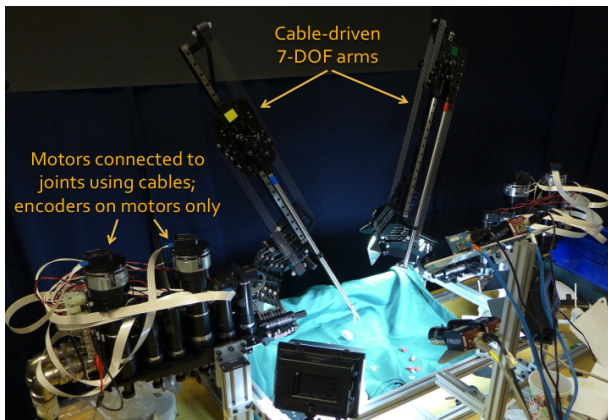


Fig. 2. Workspace for autonomous surgical debridement with the Raven II Surgical robot. The robot jointly grasps and transports the fragments with two cable-driven arms. Encoders are located only on the motors.

were determined with a stereo camera setup. This setup is illustrated in Fig. 2.

B. Test Accuracy of Learning Controller

In our first experiment, we evaluated the accuracy of the learning controller on a set of unseen testing trajectories. We first ran the Raven with its native controller to collect data to train the learning controller. We executed different instances of the debridement task; fragment locations were chosen uniformly at random over the $8\text{ cm} \times 8\text{ cm} \times 2\text{ cm}$ grid, and for safety we offset the target states by 1 cm above the platform on which target tissue fragments are placed. We collected 303 debridement task trajectories consisting of over 24,000 pairs of observed states and commanded states captured at 100 Hz. During the data collection, we operated the Raven at a speed of 5 cm per second, and each task trajectory was approximately 1 second long. We treat the states observed with PhaseSpace motion capture as the desired goal states in our training objective.

We randomly assigned a subset of 80% of these observed trajectories to the training set and held out 20% for testing. Training and testing were performed using the GPML Toolbox in matlab [26], and velocities were computed using the five-point stencil method for numerical differentiation. Since we can accurately observe the grippers at each timestep this is similar to using Model Predictive Control (MPC) for plan correction at each timestep. We evaluate accuracy on both a clean testing set and a dirty testing set. The evaluation on the clean testing set illustrates the value of data cleaning at execution time. While it is easy to clean a pre-recorded dataset, it can be quite complex to discard erroneous states during a real execution. For example, during execution of a surgical procedure, if the sensor observes a dirty state, the robot will need to perform an error recovery procedure due to the missed observation such as halting. We defer this question to future work, but our results suggest that cleaning during execution can lead to more precise control.

1) *Testing Accuracy:* Results on the test dataset using our non-linear correction are detailed in Table I below. We

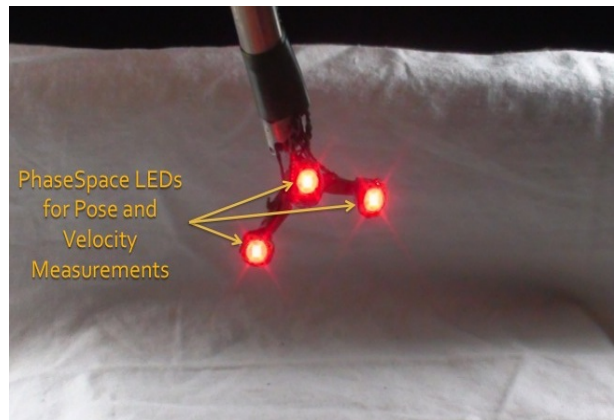


Fig. 3. PhaseSpace setup for tracking the pose and velocity of the Raven tool. We mount three LEDs: one in the center of the gripper for position measurements, and two on the fingers for orientation measurements.

evaluate the different components of the learning controller, and we compare the mean and 1-standard deviation of the error between commands and corrected desired poses along each of the pose dimensions $t_x, t_y, t_z, \phi_y, \phi_p,$ and ϕ_r for the test set. The results show that the addition of velocity significantly improves the accuracy of the learned non-linear mapping; reducing the norm position error by 50% and the norm angular error by 19%. This suggests that some of unparametrized non-linearities are correlated with the velocity, and including the velocity as a feature makes their effects easier to learn.

Furthermore, the addition of data cleaning to GPR with velocity information reduces the norm position error further by an additional 4% and the norm angular error by an additional 21%. Data cleaning also reduces the error and standard deviation of the fixed rigid offset, but the high error compared to GPR suggests the presence of non-linearities. We also found that the standard deviation is significantly reduced when using GPR with the addition of velocity information and data cleaning. We visualized the results in the translation dimensions ($t_x, t_y,$ and t_z) for the correction methods i), ii), and vi) for a sequence of 1,000 poses from the testing set in Fig. 4.

2) *Training Time and Training Set Size:* Gaussian Process Regression involves an $O(M^3)$ matrix inversion, where M is the size of the training set, potentially leading to long training times. We explored the tradeoff between the size of the training set and testing error. We randomly subsampled a fixed percentage of the examples in the training set \mathcal{X} to form a reduced training set, and trained the learning controller on this smaller set. We then evaluated the accuracy of this model on the held out testing set (Fig. 5).

We find that after 460 states, or 2.5% of the original training set, further reductions in absolute mean error from larger training sets are less than 0.1 mm for position and 0.1° for rotation. Training with 2.5% of the set takes only 18.6 seconds, as opposed to 4228.8 seconds for 100% of the dataset, evaluated on a machine with OS X with a 2.7 GHz Intel core i7 processor, and 16 GB 1600 MHz memory.

Test Set	State Variable	No Correction	Fixed Offset	Fixed Offset and Data Cleaning	Fixed Offset and GPR	Fixed Offset, GPR, and Velocity	Fixed Offset, GPR, Velocity, and Data Cleaning
Dirty	t_x (mm)	17.6 ± 8.3	4.2 ± 10.2	2.6 ± 4.6	4.2 ± 9.9	1.6 ± 3.9	1.6 ± 1.6
	t_y (mm)	17.5 ± 5.0	6.0 ± 4.7	5.9 ± 3.9	2.6 ± 3.8	1.6 ± 1.9	1.5 ± 1.5
	t_z (mm)	9.0 ± 7.4	8.3 ± 7.2	8.1 ± 6.4	2.3 ± 5.1	1.5 ± 2.2	1.4 ± 1.6
	ϕ_{yaw} (deg)	5.5 ± 10.4	4.9 ± 9.8	4.6 ± 7.8	2.0 ± 7.1	1.8 ± 3.4	1.4 ± 2.3
	ϕ_{pitch} (deg)	11.6 ± 9.3	7.7 ± 9.4	6.9 ± 7.0	3.2 ± 7.9	1.9 ± 3.6	1.5 ± 1.8
	ϕ_{roll} (deg)	22.3 ± 22.5	7.3 ± 23.6	12.0 ± 21.9	1.7 ± 9.4	2.1 ± 3.4	1.7 ± 3.3
Clean	t_x (mm)	17.4 ± 2.6	2.9 ± 2.3	2.6 ± 2.1	3.4 ± 3.2	1.3 ± 1.3	1.4 ± 1.7
	t_y (mm)	17.1 ± 4.1	5.8 ± 3.5	4.0 ± 4.2	2.2 ± 2.1	1.5 ± 1.6	1.4 ± 1.5
	t_z (mm)	7.5 ± 4.2	7.8 ± 6.2	7.1 ± 5.3	1.6 ± 1.5	1.3 ± 1.3	1.1 ± 1.2
	ϕ_{yaw} (deg)	3.2 ± 2.7	2.8 ± 1.8	2.9 ± 6.2	1.3 ± 1.8	1.5 ± 2.1	1.3 ± 2.0
	ϕ_{pitch} (deg)	10.8 ± 6.4	6.0 ± 2.3	5.9 ± 4.2	2.3 ± 2.3	1.7 ± 2.0	1.6 ± 1.9
	ϕ_{roll} (deg)	16.6 ± 9.8	6.3 ± 3.2	6.6 ± 9.8	1.7 ± 3.2	2.0 ± 3.4	1.9 ± 3.1

TABLE I

MEAN AND 1-STANDARD DEVIATION OF THE ERROR BETWEEN THE CORRECTED DESIRED STATE AND ROBOT COMMAND ON A TEST DATASET FOR EACH OF THE 6 DEGREES OF FREEDOM. THE FIRST SET OF ROWS EVALUATE THE ACCURACY OF THE FOLLOWING TECHNIQUES ON A DIRTY TESTING SET: NO CORRECTION, A FIXED RIGID OFFSET ONLY, A FIXED RIGID OFFSET WITH DATA CLEANING, A FIXED RIGID OFFSET AND GPR WITHOUT VELOCITY INFORMATION, A FIXED RIGID OFFSET AND GPR WITH VELOCITY INFORMATION, AND OUR PROPOSED SEQUENCE OF A FIXED RIGID OFFSET, GPR WITH VELOCITY INFORMATION, AND DATA CLEANING. THE SECOND SET OF ROWS EVALUATE THE SAME TECHNIQUES ON A CLEAN TESTING SET.

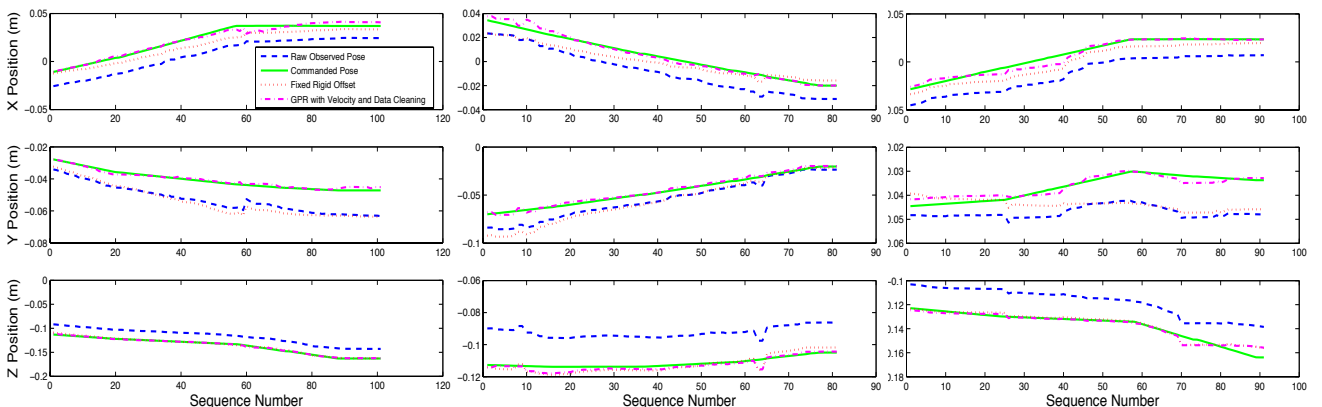


Fig. 4. Robot commands (green) and cleaned pose observations from PhaseSpace motion capture (blue), observations with fixed rigid offset (red), and observations with fixed rigid offset and GPR, using velocity information and data cleaning (magenta) for three clean test trajectories. The robot commands are the target of the non-linear mapping. The shape of the error between the fixed rigid offset and commands indicate that non-linearities are present. Our proposed method clearly reduces these non-linear errors between observation and command.

C. Repeatability of End Effector State When Applying non-linear Correction

After training the learning controller, we evaluated the accuracy and repeatability of reaching desired states by transforming and executing planned debridement task trajectories using our model. While pre-recorded trajectories provide precise velocity estimates around a given pose at each timestep, at task execution time the future velocities must be estimated from planned states. These estimates can introduce additional error in the final end-effector pose in addition to drift accumulated over the course of a trajectory. Therefore, in practice the test error rates could be achieved using feedback control at the same rate as commands are sent to the robot, but in practice this is not always possible.

We measured the accuracy and repeatability of reaching a desired position in the debridement workspace by executing

an open loop trajectory on the left arm of the Raven II platform (see [7] for details of about this arm). We visit each unique end-effector pose 10 times, and measure the actual state of the end-effector using the PhaseSpace motion capture setup described in Section V-B. We measure accuracy by computing the average absolute difference over all attempts between the desired end-effector state \mathbf{x}_d and the observed end-effector state \mathbf{x}_o after executing the a target trajectory \mathcal{T} . We measure the repeatability by computing the variance of the observed state \mathbf{x}_o over all attempts. We chose 10 random foam fragment locations uniformly across the workspace as in Section V-B, and repeatedly planned and executed trajectories to reach these locations using trajopt, a motion planning algorithm based on sequential convex optimization [28].

We compare the accuracy and repeatability of the end-effector position between using no error correction, applying

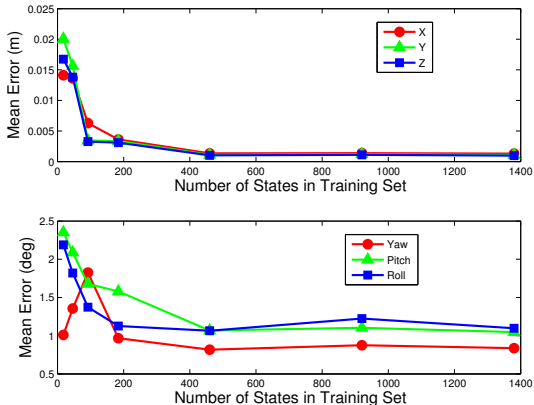


Fig. 5. The top graph shows absolute mean position (X, Y, and Z) versus the number of states used in training and the bottom graph shows rotation (Yaw, Pitch, and Roll) error versus the number of states used in training. States were sampled uniformly at random from the full training set of trajectories without replacement.

the non-linear correction to each step along the trajectory, and applying the non-linear correction to only the start and end pose along a trajectory in Table II below. The results indicate that adding the non-linear correction reduces the error in end-effector pose by approximately a factor of 10 in each state dimension. The correction improved repeatability to less than 1.0 mm using our correction, suggesting that the drift was systematic. We also get a slight improvement with endpoint correction.

State Variable	No Correction	Full Trajectory Correction	Endpoint Correction
t_x (mm)	20.3 ± 0.8	1.5 ± 1.1	2.0 ± 0.9
t_y (mm)	13.0 ± 2.5	4.9 ± 1.5	3.0 ± 1.0
t_z (mm)	22.8 ± 2.9	2.4 ± 1.4	1.0 ± 0.7

TABLE II

MEAN ACCURACY AND 1-STANDARD DEVIATION OF REACHING DEBRIDEMENT TASK POSES WITH VARIOUS POLICIES FOR APPLYING THE NON-LINEAR CORRECTION. WE COMPARE USING NO CORRECTION, CORRECTING 10 WAYPOINTS ALONG THE DESIRED TRAJECTORY, AND CORRECTING ONLY THE START AND STOP POSE OF THE TRAJECTORY.

D. Debridement Task Speedup

In our final experiment, we evaluated our controller on the task of autonomous surgical debridement with foam tissue fragments. In prior work [12], the robot had to replan its trajectory at a fixed interval to account for inaccuracies in the kinematics. We found that the replanning interval was no longer necessary to complete the task due to our accuracy of only a few millimeters with open-loop control as reported in Section V-C. Furthermore, we were able to increase the speed of the Raven from 1.0 cm per second to 6.0 cm per second. This resulted in an average task execution time of 15.8 seconds per tissue fragment, $3.8\times$ faster than the fastest previously reported result, while maintaining the task success rate [12]. Further information on this research, including

data, code, photos, and video, is available at: <http://rll.berkeley.edu/surgical>.

VI. DISCUSSION AND FUTURE WORK

We believe that this technique can generalize well to a broader class of automation problems involving imprecise state estimates and non-linear models. In future work, we will explore running this procedure in an online or bootstrapped setting, where the learning controller incrementally learns a better model during task execution from the output of a previously learned controller. We can further formulate this problem as a reinforcement learning problem with a tradeoff between exploration (executing a variety of states to learn a better model) and exploitation (completing the desired task). Recent work by Gotovos et al. using Gaussian Processes to probe the maxima and level sets of functions could be extended to probe for residual error maxima in our 6 dimensional pose space [5] and concentrate training on parts of the state space known to have significant non-linearities.

Also, although we were able to avoid replanning altogether in the surgical debridement task, for more complex tasks this may not be possible. We will explore returning a corrected pose from our learning controller as well as a confidence interval returned from GPR. This can help us automate replanning if we detect that our controller's corrected command is of low confidence.

Finally, repeated executions may change the kinematic parameters of the robot over time. Consequently, we will further consider modeling the learning controller's degradation over time. This is strongly related to models in reliability engineering and mean time between failure (MTBF) analysis. We can explore the tradeoff between online model learning and batch re-learning with respect to long-term task reliability.

VII. CONCLUSION

With the goal of precisely controlling cable-driven surgical robots, we proposed to use Gaussian Process Regression augmented with velocity features and to remove corrupted sensor observations to learn an accurate mapping between commanded poses and poses observed with an external motion capture system. We found that our technique led to more precise executions of surgical debridement tasks on the Raven II surgical robot. We showed that including velocity as a feature in GPR reduced the norm of the mean position error and the norm of the mean angular error on a set of trajectories recorded with PhaseSpace. The addition of data cleaning further reduced both the norm of the mean position error and the norm of the mean angular error over the best result without cleaning on this dataset. This technique may also be relevant for other automation systems with unmodeled non-linearities in their kinematic chains, or systems that can only be controlled through an interface with an inaccurate internal model. In future work we will apply this approach to a retrofitted Intuitive da Vinci[®] surgical robot to achieve precise control in autonomous tasks such as surgical debridement and suture tying, and we will make the code available

online at <http://rll.berkeley.edu/surgical> so that others can experiment with it on other systems such as the Baxter, Barrett Arm, snake robots, or the ABB IRB 6600.

VIII. ACKNOWLEDGMENTS

This work has been supported in part by a seed grant from the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS) and by the U.S. National Science Foundation under Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems, and by unrestricted grants from Google and Cisco. We thank our colleagues who gave feedback and suggestions, in particular PI Allison Okamura and co-PIs Greg Hager, Blake Hannaford, and Jacob Rosen, as well as Ji Ma, Hawkeye King, and Zoe McCarthy. We also thank Evan Chang-Siu and Kan Anant from PhaseSpace for their guidance in setting up our motion capture system.

REFERENCES

- [1] “Phasespace motion capture,” <http://www.phasespace.com>.
- [2] F. Abdollahi, H. A. Talebi, and R. V. Patel, “A stable neural network-based observer with application to flexible-joint manipulators,” *Neural Networks, IEEE Transactions on*, vol. 17, no. 1, pp. 118–129, 2006.
- [3] Q. Chen, W. Chen, G. Yang, and R. Liu, “An integrated two-level self-calibration method for a cable-driven humanoid arm,” in *IEEE Int. Conf. on Automation Science and Engineering (CASE)*. IEEE, 2013.
- [4] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [5] A. Gotovos, N. Casati, G. Hitz, and A. Krause, “Active learning for level set estimation,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.
- [6] B. Hannaford, J. Rosen, D. C. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. Kosari, and L. White, “Raven-II: AN open platform for surgical robotics research,” *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 954–959, Apr. 2013.
- [7] B. Hannaford, J. Rosen, D. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. Nia Kosari, and L. White, “Raven-ii: an open platform for surgical roboticsresearch,” 2013.
- [8] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [9] J. Hollerbach, W. Khalil, and M. Gautier, “Model identification,” in *Springer Handbook of Robotics*. Springer, 2008, ch. 14, pp. 321–344.
- [10] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin, “Model selection approaches for non-linear system identification: a review,” *International journal of systems science*, vol. 39, no. 10, pp. 925–946, 2008.
- [11] A. Juditsky, H. Hjalmarsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg, and Q. Zhang, “Nonlinear black-box models in system identification: Mathematical foundations,” *Automatica*, vol. 31, no. 12, pp. 1725–1750, 1995.
- [12] B. Kehoe, G. Kahn, J. Mahler, J. Kim, A. Lee, A. Lee, K. Nakagawa, S. Patil, W. D. Boyd, P. Abbeel, and K. Goldberg, “Autonomous multilateral debriement with the raven surgical robot,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2014.
- [13] J. Ko and D. Fox, “Learning gp-bayesfilters via gaussian process latent variable models,” *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, 2011.
- [14] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, “Gp-ukf: Unscented kalman filters with gaussian process prediction and observation models,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2007, pp. 1901–1907.
- [15] W. Krause, V. Schmidt, P. Rajendra, and A. Pott, “System identification and cable force control for a cable-driven parallel robot with industrial servo drives,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2014.
- [16] M. Kuss, T. Pflingsten, L. Csató, and C. E. Rasmussen, “Approximate inference for robust gaussian process regression,” *Tech. Rep, Max Planck Inst. Biological Cybern., Tubingen, Germany*, vol. 136, 2005.
- [17] A. Lewis and B. Hannaford, “Dynamically evaluated gravity compensation for the raven surgical robot,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2014.
- [18] D. Mintz, P. Meer, and A. Rosenfeld, “Analysis of the least median of squares estimator for computer vision applications,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1992, pp. 621–623.
- [19] E. Naerum, H. H. King, and B. Hannaford, “Robustness of the unscented kalman filter for state and parameter estimation in an elastic transmission,” in *Robotics: Science and Systems*, 2009.
- [20] P. Pastor, M. Kalakrishnan, J. Binney, J. Kelly, L. Righetti, G. Sukhatme, and S. Schaal, “Learning task error models for manipulation,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013.
- [21] R. K. Pearson, “Outliers in process modeling and identification,” *Control Systems Technology, IEEE Transactions on*, vol. 10, no. 1, pp. 55–63, 2002.
- [22] R. Raguram and J.-M. Frahm, “Recon: Scale-adaptive robust estimation via residual consensus,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1299–1306.
- [23] R. Raguram, J.-M. Frahm, and M. Pollefeys, “A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus,” in *European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 500–513.
- [24] S. Ramadurai, S. Kosari, H. H. King, H. Chizeck, and B. Hannaford, “Application of unscented kalman filter to a cable driven surgical robot: A simulation study,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, May 2012.
- [25] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [26] C. E. Rasmussen and H. Nickisch, “Gaussian processes for machine learning (gpml) toolbox,” *The Journal of Machine Learning Research*, vol. 9999, pp. 3011–3015, 2010.
- [27] A. Reiter, P. K. Allen, and T. Zhao, “Appearance learning for 3d tracking of robotic surgical tools,” *The International Journal of Robotics Research*, p. 0278364913507796, 2013.
- [28] J. Schulman, J. Ho, A. Lee, H. Bradlow, I. Awwal, and P. Abbeel, “Finding locally optimal, collision-free trajectories with sequential convex optimization,” in *Robotics: Science and Systems (RSS)*, 2013.
- [29] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Gaussian process dynamical models,” in *Neural Information Processing Systems (NIPS)*, vol. 18, 2005, p. 3.
- [30] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Milo, and T. Kraska, “A sample-and-clean framework for fast and accurate query processing on dirty data,” in *To Appear: ACM Special Interest Group on Management of Data (SIGMOD)*, June 2014.
- [31] E. Wernholt, *On multivariable and nonlinear identification of industrial robots*. Division of Automatic Control and Communication Systems, Department of Electrical Engineering, Linköping University, 2004.
- [32] E. Wernholt and S. Moberg, “Nonlinear gray-box identification using local models applied to industrial robots,” *Automatica*, vol. 47, no. 4, pp. 650–660, 2011.
- [33] R. L. Williams, P. Gallina, and J. Vadia, “Planar translational cable-direct-driven robots,” *Journal of Robotic Systems*, vol. 20, no. 3, pp. 107–120, 2003.