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

Jeffrey Mahler[1], Sanjay Krishnan[1], Michael Laskey[1], Siddarth Sen[1], Adithyavairavan Murali[1], Ben Kehoe[2], Sachin Patil[1], Jiannan Wang[1], Mike Franklin[1], Pieter Abbeel[1], Ken Goldberg[3]

***Abstract*— Precise control of industrial automation systems with non-linearities such as joint elasticity, variation in cable tensioning, or backlash is challenging; especially in systems for which a detailed kinematics model is not available. Cable-driven Robotic Surgical Assistants (RSAs) are one example of such an automation system, as they are designed for master-slave teleoperation. We consider a problem in which we have sensors external to the system that measure the state, however we can only control the system in its native coordinate frame with its imprecise native controller. Gaussian Process Regression (GPR) is a data-driven technique that can estimate a non-linear mapping between control inputs and sensed kinematic responses. However, GPR is sensitive to outliers, and if our sensor values are corrupted, eg. occlusions in a vision system, this can lead to an inaccurate model. In this paper, we extend the use of GPR for precise control of cable-driven surgical robots by using i) velocity as a feature in the regression and ii) data cleaning based on rotation limits and the magnitude of velocity. We evaluate this approach on the Raven II Surgical Robot, using the PhaseSpace LED-based motion capture system to track the Raven end-effector. We record 303 trajectories as the robot grasps foam "damaged tissue" fragments, which are dirty due to occlusions of the LEDs. On these recorded trajectories, including velocity information as a feature in GPR reduces the norm position error by 50% and the norm Euler angle error by 21%, with an additional 17% reduction in norm position error and a 16% reduction in norm Euler angle error using data cleaning. We use the learned kinematic control 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/raven`.**

I. Introduction

Imprecision in actuation, where actual motion varies from desired motion, is an issue in almost all industrial automation systems due to joint elasticities, variations in cable tension, backlash, or wear on geartrains. Compensating for imprecision is particularly challenging when a detailed dynamic model of the system is unavailable, as is common for proprietary systems and when accurate internal system parameters are not known. However, even in the absence of a detailed model of the system, control inputs and observations from external sensors are often available.

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

[2]Department of Mechanical Engineering; benk@berkeley.edu

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

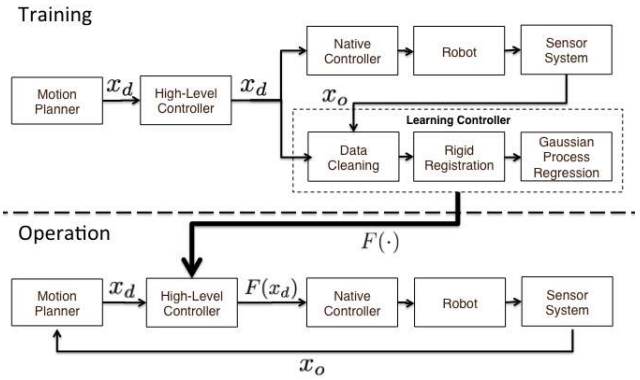

Fig. 1. System Architecture. In the training phase, we collect pairs of desired states x_d and observed states after execution x_o using the robot's imprecise kinematics model. In the learning controller, we remove corrupted data with data cleaning, and then learn F, a mapping that minimizes the difference 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 $F(x_d)$.

Semi-autonomous surgery with cable-driven Robotic Surgical Assistants (RSAs), which are designed for master slave teleoperation and have non-linearities due to cable elasticity and tensioning, is one example of a scenario with these challenges. In this paper, we present an approach to learning accurate kinematic control of cable-driven RSAs using a combination of rigid transformation and Gaussian Process Regression (GPR), which was shown by Pastor et al. to control the cable-driven DARPA ARM-S to an accuracy of 2 mm - 4 mm without parameterization of kinematic non-linearities [15]. However, past work has not considered using velocity as a feature in GPR, which can influence non-linearities at higher speeds. Addtionally, past work does not explicitly address the effect of dirty data on GPR, which often occur in automation settings due to occlusions and lighting changes in the workspace. We extend previous work on GPR for kinematic control by i) adding velocity as a feature in GPR and ii) using data cleaning to remove invalid pairs of states. 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 non-linear automation systems that do not have a detailed kinematic model available, that have an incomplete kinematics model, that can compund small kinematic errors due to lack of shaft

encoders, or that have complicated and/or state-dependent non-linear dynamics (e.g., coupling between many joints or state-dependent cable tension). Examples of such systems include the ABB IRB 6600, an low-cost industrial robot with dynamic coupling effects and flexible joints [25], the Baxter robot, the Barrett arm, the Ekso Bionix ProStep©, and cable-driven 7-DOF humanoid arms [2].

We evaluate of our approach on the Raven II Surgical Robot, an open source hardware platform for research on RSAs [5]. 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. The PhaseSpace cameras can capture at 480 Hz and localize LEDs to within 0.1 mm, resulting in accurate pose and velocity estimation when the LEDs are not occluded from rotation of the tool.

We record 303 observed and desired trajectories for our two-arm Raven surgical robot system as it autonomously executes surgical debridement, where the goal is to find, grasp, and transport "damaged tissue" fragments, following the setup of Kehoe et al. [10]. We use these recordings to train the learning controller in pose space, and we measure the improvement in the mean absolute accuracy of our approach on a held-out set of test trajectories. Our results show that including velocity as a feature in GPR reduces the norm position error by 50% and the norm angular error by 21% on these recorded trajectories. Furthermore, the addition of data cleaning reduces the norm position error by an additional 17% and the norm angular error by an additional 16% over the best result without any data cleaning. The results also indicate that our contributions reduce the standard deviation of the test error. We also measure the accuracy and repeatability of reaching desired poses in open-loop using our correction method to characterize the effect of drift when using our correction method. Using the new kinematic control model, we achieve a 3.8× speedup over previous work on the autonomous surgical debridement task [10].

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 [7] for an overview. In [14], Nicosa and Tornambé use output injection to model joint elasticity parameters, but they assume the exact state of the robot is known. 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. [26]. 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 [2]. Naerum et al. considered both offline and online parameter estimation using the Unscented Kalman Filter [13] 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 [20].

Cable-driven robots can also be controlled without without explicit parameterization of non-linearities. Abdholli 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 [1]. In [27], 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. Feedforward-feeback control has also been proposed to compensate for unmodeled non-linearities in 2-DOF oscillating piezocantilevers [19]. 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® [23]. While this allows for an incomplete kinematics model, the linearization may cause divergence when the error dynamics are inaccurate or approximated poorly.

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 [8], [9]. 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 [15]. The authors demonstrate mean position error for the DARPA ARM-S on the order of 2 mm - 4 mm, but did not explicity consider including velocity as a feature to GPR and the effect of data cleaning on model accuracy. 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. For general statistical models, Random Sample Consensus (RANSAC) has been extensively studied to remove high-magnitude outliers [3]. Various extensions to the RANSAC model have been proposed to cope with tuning: Least of Medians [12] and Residual Consensus [17], 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 [18]. 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 [11]. 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 [16], 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 futher 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, repectively, 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, repectively
- $R(\mathbf{p}) \in \mathbb{R}^{3\times3}$ is the rotation matrix corresponding to the yaw, pitch, and roll rotations of $\mathbf{p}$
- $\mathbf{t}(\mathbf{p}) \in \mathbb{R}^{3\times1}$ is the translation of $\mathbf{p}$
- $T(\mathbf{p}) = \left(\begin{smallmatrix} R(\mathbf{p}) & \mathbf{t}(\mathbf{p}) \\ \mathbf{0} & 1 \end{smallmatrix}\right) \in \mathbb{R}^{4\times4}$ is the rigid transformation matrix corresponding to $\mathbf{p}$
- $\mathcal{T} = \{\mathbf{x}_1, ..., \mathbf{x}_t\}$ denotes a trajectory of length t
- $\mathcal{X} = \{(\mathbf{x}_{c,1}, \mathbf{x}_{d,1}), ..., (\mathbf{x}_{c,M}, \mathbf{x}_{d,M})\}$ denotes the set of M pairs of corresponding observed and desired training states
- $\mathcal{Y} = \{(\mathbf{x}_{c,1}, \mathbf{x}_{d,1}), ..., (\mathbf{x}_{c,N}, \mathbf{x}_{d,N})\}$ denotes the set of N pairs of corresponding observed and desired testing states

Our primary goal is to learn how to send commands to 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 estimates a function F to tranform desired states before sending them to the native controller. We consider a variant of the approach of [15] 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 $\|F(\mathbf{x}_d) - \mathbf{x}_c\|$ is minimized over our test data set $\mathcal{Y}$ given the constraints on F.

We estimate the constant rigid offset using orthornormally constrained least squares over the set of rigid transformation matrices. Following the approach of [15], we estimate the non-linear component using Gaussian Proccess Regression (GPR). We review these regression methods in Section IV to highlight details specific to the kinematic control setting. For data cleaning, we model corruption of the training and test data as sparse noise that is not physically realizable by the robot. Our data cleaning can remove these examples to avoid biasing our model.

We assume a one-to-one 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 assume that the sensor does not introduce additional systematic biases into the state estimates.

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 consectutive stages performed offline:

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

We collect our training and test sets by recording time-synchronized pairs of commanded and observed robot states with PhaseSpace motion capture, and treat the observed states as the desired states in training. Details of the data collection procedure can be found in Section V.

Our data cleaning consists of removing states that are outside the physical limitations of the sensor and are outliers with respect to our model. We estimate the rigid transformation by solving an orthonormally-constrained least squares problem on the set of training pose matrices. In GPR, we use the entire current state as features to regress to the desired poses. We review the methods of estimating the rigid transformation and GPR to highlight details specific to the kinematic control setting and to clarify some of the motivation for data cleaning; then we describe our data cleaning process.

A. 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 ($\mathcal{X}$). A rigid tranformation is composed of an orthornormal rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^{3\times1}$. We minimize the error with respect to the Frobenius norm:

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

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 [6].

B. 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. We apply the rigid transformation $G(\cdot)$, and we fit the transformed observations to a Gaussian process. Solving this problem amounts to a form of kernel linear regression.

A key parameter to the problem is the kernel, a measure of similarity between two training examples. We apply GPR with a kernel (called the Radial Basis Function RBF) of the following form:

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

where σ denotes the signal variance (can be interpreted as a smoothing parameter) 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 estimate these parameters using the GPML Toolbox [22]. See [21] 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. As opposed to past work, we directly include the velocity of $\mathbf{x}_o$ in the features of GPR, which makes training time longer but provides more information about the sources of non-linearities. Thus, we perform kernelized regression for output dimension i:

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

where $\mathbf{y}_i = (\mathbf{x}_{d,1}^{(i)}, ..., \mathbf{x}_{d,M}^{(i)})$ 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.

C. 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 occlusions of the LED markers, and an occlusion for one or more of the cameras can lead low quality data. We further observed discontinuities in the output at the point where tool moved into the occluded region. 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 hyper-parameters: 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 Medians (LMEDS) [12] and Residual Consensus [17] 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 conjuction 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 our how our cleaning results in reduced mean absolute test error and also reduces the standard deviation of these errors.

V. Experiments

A. Experimental Setup

We tracked the motion of the Raven II with a four camera PhaseSpace Impluse X2 motion capture system. The four cameras were placed in an arc of approximately 120° at a distance of approximately 1.5 m to the Raven instrument, oriented towards the instrument. We mounted three LEDs to the Raven instrument to measure the pose and velocity (Fig. 3). The Raven performs a surgical debridement task, in which the goal is to find, grasp, and transport "damaged tissue" fragments [10], and we represented the tissue fragments with pieces of red foam. The locations of the red foam were determined with a stereo camera setup. This setup is illustrated in Fig. 2

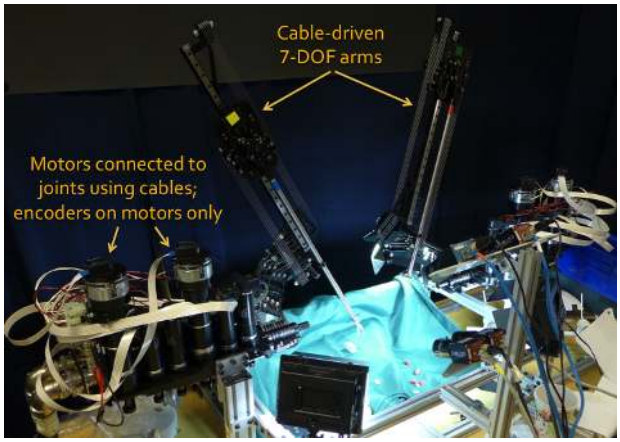

Fig. 2. Workspace for autonomous surgical debridment with the Raven II Surgical robot. The robot jointly grasps and transports the fragments with two cable-driven arms. Encoders are located only on the motors.

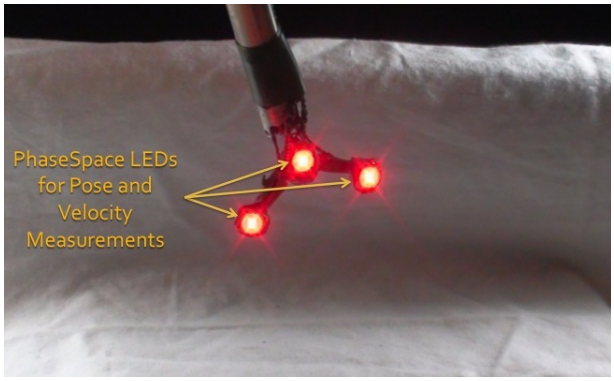

Fig. 3. PhaseSpace setup for tracking the pose and velocity of the Raven instrument. We mount three LEDs: one in the center of the gripper for position measurements, and two more on the fingers for orientation measurements.

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 cm $\times$ 8 cm $\times$ 2 cm grid, and for saftey 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 trajectores to the training set and held out 20% for testing. Training and testing were performed using the GPML Toolbox in matlab [22], and velocities were computed using the five-point stencil method for numerical differentiation. 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 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. We compare the following correction methods: i) no correction, ii) only a fixed rigid offset, iii) a fixed rigid offset with data cleaning iv) a fixed rigid offset and GPR without velocity information, v) a rigid transformation and GPR with velocity information, and vi) a rigid transformation and GPR with velocity information and data cleaning (our proposed method). 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 21%. 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 17% and the norm angular error by an additional 16%. 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

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.

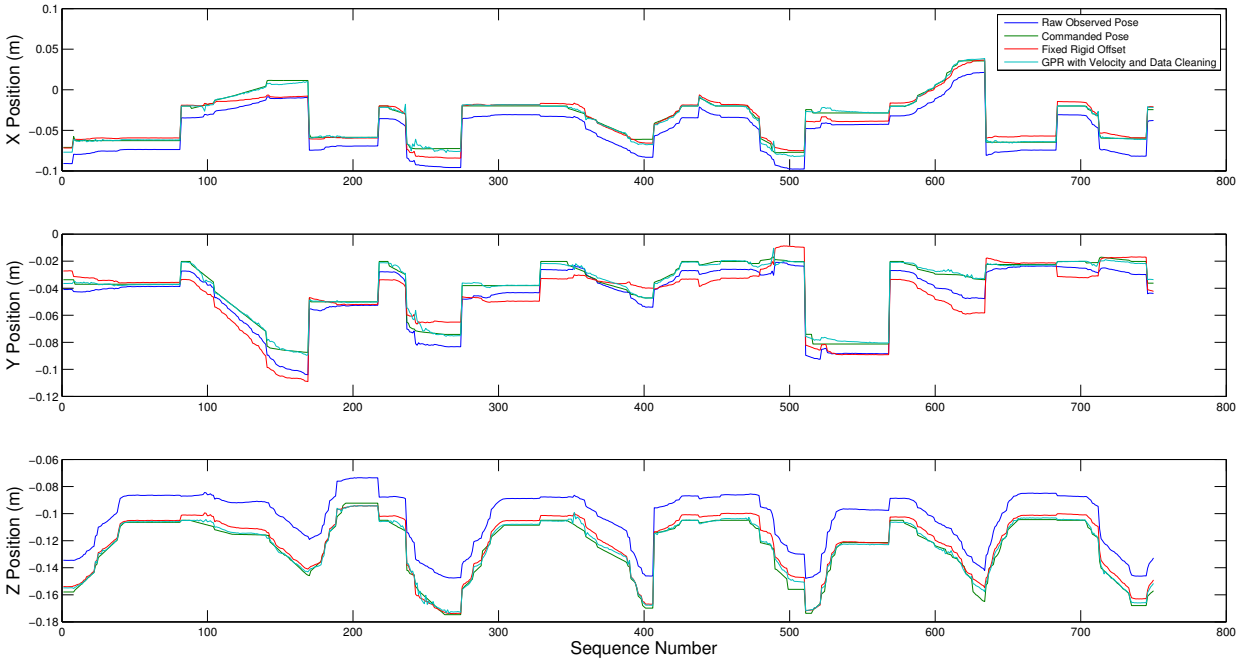

Fig. 4. Robot commands (green) and pose observations from PhaseSpace motion capture (blue), observations with fixed rigid offset (red), and observations with fixed rigid offset and GPR, using velocity information and data cleaning (teal). The robot commands are the target of the non-linear mapping. The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

dataset, evaluated on a machine with OS X with a 2.7 GHz Intel core i7 processor, and 16 GB 1600 MHz memory.

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, the test error rates can be achieved in practice using feedback control at the same frequency at which commands are sent to the controller during training, 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 Raven II gold model 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 optimiztion [24].

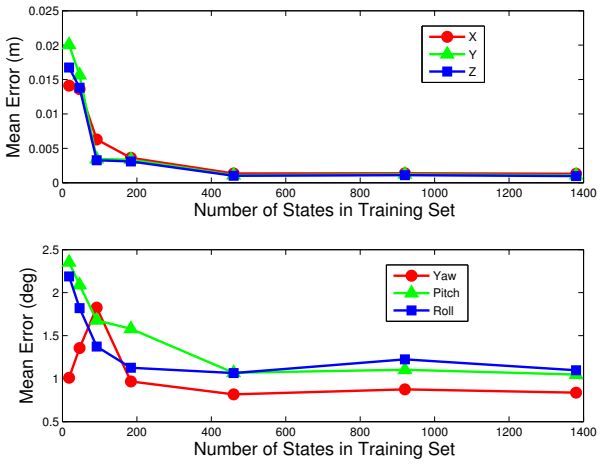

Fig. 5. The top graph shows absolute mean position (X, Y, and Z) versus the number of states used in training and the bottom graph shows rotation (Yaw, Pitch, and Roll) error versus the number of states used in training. States were sampled uniformly at random from the full training set of trajectories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

We compare the accuracy and repeatabilty of the end-effector position between using no error correction, applying 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 ADN 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 evaulated our controller on the task of autonomous surgical debridement with foam tissue fragments. In prior work [10], 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× faster than the fastest previously reported result, while maintaining the task success rate [10]. Further information on this research, including data, code, photos, and video, is available at: `http://rll.berkeley.edu/raven`.

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 futher 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 [4] and concentrate training on parts of the state space known to have significant non-linearities.

Another particularly promising result is that we were able to avoid replanning altogether in the surgical debridement task. However, for more complex tasks this may not be possible. We will explore not only returning a corrected pose from our learning controller but also a confidence interval. 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 Before Failure analysis. We can explore the tradeoff between online model learning and batch re-learing with respect to long-term task reliability.

VII. Conclusion

The Raven II surgical robot's nonlinear kinematics and inaccessible native controller make precise kinematic control challenging. We proposed a technique using Gaussian Process Regression combined with data cleaning and poses augmented with velocity features to learn a mapping between commanded states and the observed states with a external motion capture system. We found that our technique led to more precise executions of surgical debridement tasks on the Raven. We showed that including velocity as a feature in GPR reduced the norm position error by 50% and the norm angular error by 21% on a set of trajectories recorde with PhaseSpace motion capture. Furthermore, the addition of data cleaning reduced the norm position error by an additional 17% and the norm angular error by an additional 16% over the best result without cleaning on this dataset. In future work we will apply this approach to a retrofitted Intuitive da Vinci® surgical robot to achieve precise control for executing autonomous tasks such as surgical debridement and suture tying, and we will make the code available online at `http://rll.berkeley.edu/raven` 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. 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 and Hawkeye King. We also thank Evan Chang-Siu and Kan Anant from PhaseSpace for their guidance in setting up our motion capture system.

References

[1] 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.

[2] Q. Chen, W. Chen, G. Yang, and R. Liu, "An integrated two-level self-calibration method for a cable-driven humanoid arm," 2013.

[3] 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.

[4] A. Gotovos, N. Casati, G. Hitz, and A. Krause, "Active learning for level set estimation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[5] 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.

[6] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[7] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in *Springer Handbook of Robotics*. Springer, 2008, ch. 14, pp. 321–344.

[8] 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.

[9] 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.

[10] 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 debridement with the raven surgical robot," in *International Conference on Robotics and Automation, 2014, (Under Review)*. IEEE, 2014.

[11] M. Kuss, T. Pfingsten, L. Csató, and C. E. Rasmussen, "Approximate inference for robust gaussian process regression," *Max Planck Inst. Biological Cybern., Tubingen, GermanyTech. Rep*, vol. 136, 2005.

[12] D. Mintz, P. Meer, and A. Rosenfeld, "Analysis of the least median of squares estimator for computer vision applications," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*. IEEE, 1992, pp. 621–623.

[13] 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.

[14] S. Nicosia and A. Tornambé, "A new method for the parameter estimation of elastic robots," in *Systems, Man, and Cybernetics, 1988. Proceedings of the 1988 IEEE International Conference on*, vol. 1. IEEE, 1988, pp. 357–360.

[15] 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.

[16] R. K. Pearson, "Outliers in process modeling and identification," *Control Systems Technology, IEEE Transactions on*, vol. 10, no. 1, pp. 55–63, 2002.

[17] 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.

[18] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 500–513.

[19] M. Rakotondrabe, K. Rabenorosoa, J. Agnus, and N. Chaillet, "Robust feedforward-feedback control of a nonlinear and oscillating 2-dof piezocantilever," *Automation Science and Engineering, IEEE Transactions on*, vol. 8, no. 3, pp. 506–519, 2011.

[20] 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 *2012 IEEE International Conference on Robotics and Automation, St. Paul-Minneapolis,*, May 2012.

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

[22] 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.

[23] 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.

[24] 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.

[25] E. Wernholt, *On multivariable and nonlinear identification of industrial robots*. Division of Automatic Control and Communication Systems, Department of Electrical Engineering, Linköping University, 2004.

[26] 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.

[27] 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.