2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Adaptive Nonparametric Kinematic Modeling of Concentric Tube Robots

Georgios Fagogenis[1], Christos Bergeles[2] and Pierre E. Dupont[1]

***Abstract*— Concentric tube robots comprise telescopic pre-curved elastic tubes. The robot's tip and shape are controlled via relative tube motions, *i.e.* tube rotations and translations. Non-linear interactions between the tubes, *e.g.* friction and torsion, as well as uncertainty in the physical properties of the tubes themselves, *e.g.* the Young's modulus, curvature, or stiffness, hinder accurate kinematic modelling. In this paper, we present a machine-learning-based methodology for kinematic modelling of concentric tube robots and *in situ* model adaptation. Our approach is based on Locally Weighted Projection Regression (LWPR). The model comprises an ensemble of linear models, each of which locally approximates the original complex kinematic relation. LWPR can accommodate for model deviations by adjusting the respective local models at run-time, resulting in an adaptive kinematics framework. We evaluated our approach on data gathered from a three-tube robot, and report high accuracy across the robot's configuration space.**

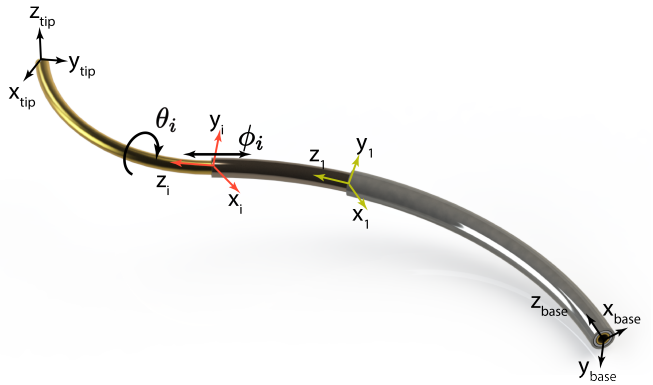


Fig. 1. Exemplary concentric tube robot comprising three pre-curved tubes.

I. Introduction

While surgical robot design for laparoscopy has converged around a standard architecture, the design of robots for minimally invasive treatment of deep-seated legions is still an area of research. Catheter-like robots can navigate anatomical pathways such as the vessels or urinary tract to perform interventions; however, the compliance and passive shape control of traditional catheters preclude their use for procedures requiring tissue manipulation.

Concentric tube robots (CTR; Fig.1) alleviate these limitations by actively controlling the pose (*i.e.* position and orientation) of the robot's distal end (tip) [1], [2]. Moreover, certain concentric tube robot designs enable partial control of the robot's shape. These devices offer a means to perform elaborate tip motion, tissue interaction, and navigation of anatomical pathways. Concentric tube robots consist of pre-curved, superelastic tubes, which rotate and telescopically extend relative to each other. These relative motions enable shape and tip pose control. The pre-curvature, stiffness, and curved length of each tube can be computationally selected to accomplish the desired surgical tasks [3], [4], [5], which range from cardiac surgery [6], to neurosurgery [7], and prostate surgery [8].

This work was supported by the National Institutes of Health under Grant R01HL124020.

[1]G. Fagogenis and P. E. Dupont are with the Department of Cardiovascular Surgery, Boston Children's Hospital, Harvard Medical School, Boston, MA, 02115, USA. firstname.lastname@childrens.harvard.edu

[2]Christos Bergeles is with the Translational Imaging Group, Centre for Medical Image Computing, Department of Medical Physics and Biomedical Engineering, UCL, London, NW1 2HE, United Kingdom. c.bergeles@ucl.ac.uk

CTR kinematic models predict the tip's pose given the relative displacement of the tubes. Mechanics-based models exist that account for torsion [1], [9], friction [10], and known external forces [11], [12]. Although exteroceptive sensing via EM trackers or proprioceptive sensing via Fiber-Bragg-Gratings may also be available [13], the importance of accurate kinematics is paramount for planning and control.

Inverse kinematics provide an estimation of the relative tube motion that would yield a desired tip pose (feedforward control). Sensor measurements are then used within this scheme to reduce the estimation error further (closed loop control). Closed loop control, however, also relies on kinematics: closed-loop schemes use the robot's inverse Jacobian to map tip-error minimizing directions onto respective tube motion. Besides control, motion planners also require a kinematic model [14], [15] . Moreover, kinematics may drive the robot "blindly" between low-frequency measurements. Finally, accurate kinematics, combined with an appropriate recursive estimation algorithm, may provide graceful degradation in case of sensor failure.

What current approaches do not fully account for is complex nonlinear interactions between tubes. For example, existing friction models have seen limited experimental evaluation on multi-tube robots. Further, even though NiTi, the common material of concentric tubes, is well characterised, measurements of Young's modulus and Poisson ratio for a particular tube will contain inaccuracies. Similarly, the pre-curvature, stiffness, and length estimations for each tube will contain errors. All these lead to misestimation of tip pose and robot shape, even when the most advanced kinematics models are used. Adaptive kinematics-model estimation is,

thus, desirable. Ongoing research on online adaptation of the truncated Fourier series defining the kinematics model of [1] in [16], and neural-network-based kinematics trained through observation in [17] are preliminary approaches that address the above issues. However, both [16], [17] will overfit locally, tuning the parameters of a global model based on local information. Local regression methods can remedy the problem of overfitting by adapting only where necessary, preserving global model accuracy.

Here, we present a machine learning approach for adaptively modeling concentric tube robots kinematics based on Locally Weighted Projection Regression (LWPR) [18], yielding an accurate, non-parametric representation of the robot's kinematics. Incoming pose measurements refine the local models at runtime, *in situ*, to accommodate complex non-linear phenomena. Local adaptation improves model accuracy while preserving global goodness-of-fit. Our approach is evaluated experimentally using a three-tube robot and a tip-mounted EM tracker. High accuracy across the robot's configuration space is achieved, outperforming existing models.

II. Kinematics

A kinematic model is a map between two spaces: the configuration space and the task space. The configuration space for an n-tube CTR is the set of relative tube rotations $\alpha_{i1} = \theta_i - \theta_1$ and translations $d_{i1} = \phi_i - \phi_1, i \in \{2,...,n\}$; both relative rotations and translations are computed with respect to the first tube. Hence, for an n-tube robot, a point in the configuration space is defined as: $\mathbf{x} = [\alpha_{i1}, d_{i1}, ..., \alpha_{n1}, d_{n1}]^T \in \mathbb{R}^{2n-2}$. The task space comprises all feasible end-effector (tip) poses. A pose is represented as a homogeneous transformation $\mathbf{y} \in SE(3)$ between a coordinate frame attached at the robot's tip and a static reference frame attached at the robot's base. Finally, the robot's base can move as a rigid body in space. Hence, the full kinematic representation is defined as follows:

$$\mathbf{y} = \mathbf{H}_{\text{base}} f(\mathbf{x}) \tag{1}$$

where $\mathbf{H}_{\text{base}} \in SE(3)$ is the homogeneous transformation that accounts for the motion of the robot's base, and $f(\mathbf{x}) : \mathbb{R}^{2n-2} \mapsto SE(3)$ is the robot's non-linear kinematics.

In this paper, we use non-parametric regression, specifically Locally Weighted Projection Regression [18], to approximate the kinematic map $f(\mathbf{x})$. This section describes the application of this technique to kinematic modeling.

A. Locally Weighted Projection Regression

Regression is used to approximate relations between variables from an input space X to an output space Y; *e.g.*, the kinematic model $f : \mathbb{R}^{2n-2} \mapsto SE(3)$. The spaces X and Y are known as the *domain* and *range* of the *target* function $f(\mathbf{x})$. Kernel-based regression, specifically, will predict the tip's pose $f(\mathbf{x})$, for any configuration $\mathbf{x} \in \mathbb{R}^{2n-2}$, using training data; *i.e.*, tuples of the form $(\mathbf{x}_{\text{train}}, \mathbf{y}_{\text{train}})$ such that $\mathbf{y}_{\text{train}} = f(\mathbf{x}_{\text{train}})$, $\mathbf{x}_{\text{train}} \in X$. Configurations close to $\mathbf{x}$ provide information for the robot's tip pose $f(\mathbf{x})$. Like a distance metric, a kernel quantifies proximity in the configuration space by assigning a scalar value k (weight) to any pair of robot configurations. For regression, training data for which $k(\mathbf{x}, \mathbf{x}_j) > \varepsilon > 0$ all contribute to the function estimation at $\mathbf{x}$; these data correspond to robot configurations sufficiently close to the query point.

Locally Weighted Projection Regression approximates nonlinear functions using a collection of linear models. To this end, the function's input space is partitioned into local neighborhoods. At each neighborhood, the target function is approximated by a hyperplane. To partition the space into such neighborhoods, we utilize *Gaussian* kernels. The mean (center) and covariance (width) define a Gaussian kernel uniquely. The Gaussian kernel assigns a weight to any point in the domain, defined as follows:

$$w(\mathbf{x}) = exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^T \mathbf{D}(\mathbf{x}-\mathbf{c})\right) \tag{2}$$

where $\mathbf{c} \in \mathbb{R}^{2n-2}$ is the center of the Gaussian kernel and $\mathbf{D}^{-1} \in \mathbb{R}^{(2n-2)\times(2n-2)}$ is the covariance; equivalent to the kernel's width. The points around the center with $w > \varepsilon > 0$ define the kernel's *receptive field*.

Ideally, the union of all receptive fields covers the domain of the target function. In practice, this is not guaranteed, as it depends on the distribution of the training data. For each receptive field, LWPR fits training data with a linear model through Partial Least Squares (PLS; [19]). Partial Least Squares combines linear regression with dimensionality reduction. For this reason, LWPR's accuracy is preserved for high-dimensional problems.

B. Learning

Learning the kinematic model entails partitioning the robot's configuration space with kernels and fitting local models to each kernel's receptive field. To this end, the center and width of each receptive field should be specified. For each neighborhood in the robot's configuration space, the coefficients of the hyperplane that fits the training data locally are also required. All these parameters are updated by minimizing the following cost function:

$$C = \frac{\sum_{i=1}^{M} w_i (\mathbf{y}_i - \hat{\mathbf{y}}_{i,-i})^2}{\sum_{i=1}^{M} w_i} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{3}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ are M training examples, and $\hat{\mathbf{y}}_{i,-i}$ is the estimation of the target function with the i-th data point excluded from the training set [20]. In the above equation, N is the number of activated Receptive Fields. The first term in (3) is the *leave-one-out-cross-validation* error of the model [21]. The second term penalizes infinitely small receptive fields to avoid model overfitting. The parameters of each RF are updated by applying Stochastic Gradient Descent on cost C:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \tag{4}$$

where $\boldsymbol{\theta}$ summarizes all the training parameters of LWPR (center and width of each receptive field, local hyperplane parameters), and η is the *step* of the stochastic

gradient descent (also known as the *learning rate*). The difference between stochastic and standard gradient descent lies in the computation of the cost gradient: in gradient descent, all data samples are used to compute the expected value of the gradient; whereas, stochastic gradient descent calculates the cost gradient locally at the neighborhood of the incoming training data sample. Therefore, the standard gradient descent yields a batch optimization method, whereas stochastic gradient descent affords incremental parameter updates. The update rules for the model parameters have been omitted due to space limitations; readers are referred to [18].

C. Model Prediction

Given the set of all local models, LWPR combines individual, local predictions into a final estimation of the tip's pose. For a configuration $\mathbf{x}_q$, LWPR detects all models, relevant to that estimation; namely, all models $k \in \{0,...,K\}$ for which $\mathbf{x}_q \in RF_k$, where RF_k denotes the receptive field of the k-th model. The kernel associated with each of the K models assigns a weight to each local prediction of the tip's pose. The final estimation $\hat{\mathbf{y}}$, *i.e.* the predicted end-effector pose, is the weighted sum of all activated model-predictions:

$$\hat{\mathbf{y}} = \left(\sum_{k=1}^{K} w_k \right)^{-1} \sum_{k=1}^{K} w_k \hat{\mathbf{y}}_k \tag{5}$$

where w_k denotes the weight, as indicated by each kernel (see Equation (2)), and $\hat{\mathbf{y}}_k$ the local prediction of the respective model.

D. Adaptation

Using Stochastic Gradient Descent, LWPR continuously incorporates new information. LWPR's parameters are updated as follows: when a new set of joint variables and tip pose $(\mathbf{x}_i, \mathbf{y}_i)$ is provided, the algorithm first detects the activated models by checking which of the respective kernels yield weights w_k above a predefined threshold ε. Next, the activated models update the local parameters using (4). When none of the models produces a sufficiently large weight, LWPR creates a new Receptive Field, centered on this particular training sample. Given a second sample within the newly created receptive field, PLS computes the local hyperplane. This procedure is repeated as more data become available. In this way, LWPR adapts at runtime, effectively incorporating new information into the initial model. This feature is exploited to refine the kinematic model in real-time, given an initial approximation of the kinematics.

Adapting the kinematics at one point in the configuration space improves model estimation at nearby configurations, as well. While at configuration $\mathbf{x}_1$, LWPR adapts all active models in the neighborhood of $\mathbf{x}_1$ (see Fig. 2). The union of the receptive fields of all activated models defines a region in configuration space around $\mathbf{x}_1$. Note that the same models are responsible for model prediction at $\mathbf{x}_1$. Given a small displacement $\Delta\mathbf{y}$ in the task space, the robot's configuration becomes $\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{J}^{\dagger}\Delta\mathbf{y}$, where $\mathbf{J}^{\dagger}$ stands for the pseudoinverse of the robot's Jacobian matrix $\mathbf{J}$. Similarly, a new region is defined at $\mathbf{x}_2$, based on the models activated by $\mathbf{x}_2$. For a small $\Delta\mathbf{x} = \mathbf{x}_2 - \mathbf{x}_1$ the two successive regions of activation overlap. The activated models within the overlapping region, have been already updated once when the robot was at $\mathbf{x}_1$. Hence, the prediction at $\mathbf{x}_2$ is improved, even before adapting again at $\mathbf{x}_2$.

The above concept is depicted in Fig. 2, where the gray ellipses represent activated regions as the robot moves along a trajectory in configuration space (marked as a red line). Zooming in, the activation region at $\mathbf{x}_1$ (left circle) partially overlaps with activation region at $\mathbf{x}_2$. The overlapping region (shaded area) is responsible for the rapid improvement of model accuracy during online adaptation. The magnitude of $\Delta\mathbf{x}$ directly influences the size of the overlapping regions, and depends on the desired displacement $\Delta\mathbf{y}$ in the workspace, as well as the magnitude of $1/det(\mathbf{J}^T\mathbf{J})$. From that follows that close to singular configurations, where $det(\mathbf{J}^T\mathbf{J}) \to 0$, the robot needs to move very slowly to maintain the overlap of successive regions of activation.

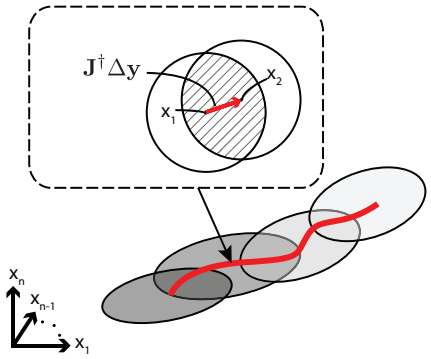


Fig. 2. Model-estimation improvement due to overlapping adaptation regions.

III. Generating a Nominal Kinematic Model

LWPR can be trained on a synthetic dataset to yield an initial kinematic model. The mechanics-based parametric model of [1], [12] is used here to generate a training data set. Initially, an n-dimensional grid is created, spanning the robot's configuration space. The grid consists of equispaced points, ensuring in this way uniform sampling of the configuration space. For each configuration $\mathbf{x}$, the parametric model is applied, yielding a pose $\mathbf{y}$ for the robot's end-effector. The resulting $\{\mathbf{x}, \mathbf{y}\}$ pairs are used to train the initial LWPR model.

The generated data are split into two datasets: the training and test datasets. The training dataset consists of 85% of the original data and is used to identify LWPR's internal parameters. The remaining data (test dataset) are used to estimate the model's generalization capability. Generating data is not essential to apply LWPR; *i.e.*, the model can be trained online while controlling the robot. However, an initial model, even a crude one, will exhibit high accuracy much faster.

Two parameters may influence LWPR's learning performance: the width of a newly created receptive field ($\mathbf{D}_{init}$)

and the learning rate η in (4). Initial width was chosen based on the density of the grid. Specifically, initial width was set to three times larger than the distance between successive training configurations. Model accuracy is not very sensitive to this parameter since it is refined online by the algorithm. The learning rate η is, in turn, adjusted using the Incremental Delta Bar Delta (IDBD) algorithm [22].

Relative tube rotations require some special handling. Rotational joint variables are constrained in the closed interval $[-\pi,\pi]$ with the limit points $-\pi,\pi$ *identified*. Identification declares two points equivalent. The endpoints of the line segment (corresponding to the limit points of the closed interval) are "glued" together, forming a closed loop. During distance computation, the loop can be traversed in both directions. In this way, points that are in the neighborhood of the two endpoints get closer. The distance metric should reflect this property of the target function's domain.

To alleviate this issue, we replicate data from the bounds and extend the training dataset to $[-\pi-\varepsilon,\pi+\varepsilon],\varepsilon>0$. Specifically, data that correspond to $[\pi-\varepsilon,\pi]$ will be replicated in the range of $[-\pi-\varepsilon,-\pi]$; whereas, data from $[-\pi,-\pi+\varepsilon]$ will be copied at $[\pi,\pi+\varepsilon]$. Modifying the dataset in this way has the same result as changing the metric, but it does not require modification of the algorithm's implementation.

IV. Experiments

In this paper, we used the CTR shown in Fig. 3 that consists of three NiTi tubes. Tubes 1 and 2 have one constant curvature section and are equally stiff. Moreover, curved sections of both tubes have the same radius of curvature. The second tube is merely allowed to rotate relatively to the first; conversely, relative tube translation is fixed at zero ($d_{21}=0$). Based on the above, tubes 1 and 2 form a balanced pair; *i.e.*, the combination of the two tubes at $\alpha_{21}=\pi$ yields a straight robot body. The third tube comprises two sections: one straight and one with nonzero constant curvature. Table I summarizes all robot parameters:

TABLE I
Robot Parameters

	Tube 1	**Tube 2**	**Tube 3**	
	Section 1	Section 1	Section 1	Section 2
Length (mm)	150	150	150	87
Curvature (m^{-1})	3.7736	3.7736	0	14.29
Relative Stiffness	1	1	0.2857	0.2857

The configuration space of this robot consists of all $\mathbf{x}=[\alpha_{21},\alpha_{31},d_{31}]^T$, where $\alpha_{21},\alpha_{31}\in[-\pi,\pi]$ and $d_{31}\in[0,87]$. The task space consists of end-effector position and orientation. In this experiment, we controlled just the direction of $\mathbf{z}_{\text{tip}}$ (tangent vector at robot's tip). For this reason, the representation for the orientation is merely the 3-dimensional coordinates of this unit vector. Due to the unit-vector constraint, only two of the three coordinates of $\mathbf{z}_{\text{tip}}$ are independent. Therefore, pose $\mathbf{y}$ belongs to a five-dimensional manifold ($\mathbb{R}^3\times\mathbb{S}^2$) embedded in $\mathbb{R}^6$. LWPR approximates the map $f:\mathbb{R}^3\mapsto\mathbb{R}^6$. Next, the result corresponding to $\mathbf{z}_{\text{tip}}$ is normalized to a unit vector, projecting the LWPR estimation from $\mathbb{R}^6$ onto the manifold. The robot can also translate and rotate as a whole along $\mathbf{z}_{\text{base}}$, which adds two more degrees of freedom for control. Since this is merely a rigid body transformation, LWPR does not consider these two degrees of freedom during training.

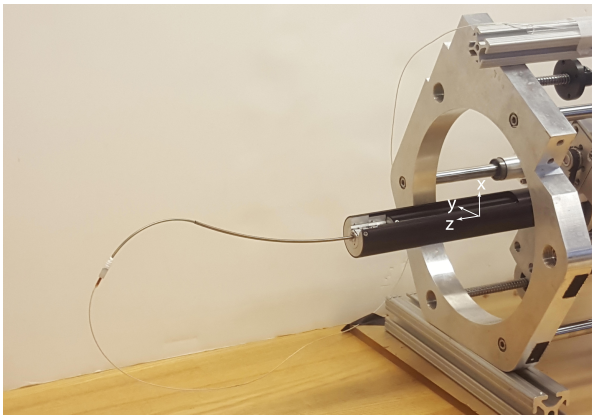


Fig. 3. Robot comprised of three tubes with tip-mounted EM sensor.

For training, we generated a $80\times80\times80$ grid covering the robot's configuration space uniformly. For each configuration, we computed the corresponding end-effector position, as described in Section III. Then an LWPR model was trained on the generated data. Table II provides the absolute and normalized Root Mean Square Error (RMSE) of the model predictions computed on the test dataset for all task space dimensions. For evaluating orientation accuracy, we used the relative angle between the predicted and measured tangent vectors at the robot's tip. Next, the trained kinematics model

TABLE II
Model Validation

	x	y	z	orientation
RMSE	0.97mm	0.87mm	0.41mm	1.89 deg
nRMSE [100%]	0.68	0.58	0.52	1.05

was tested on the robot. The experimental setup consists of the three-tube robot mounted on a drive system, a haptic device (Phantom Omni) for teleoperation, and a Trackstar EM-tracker by NDI. Control software is implemented in C++ and is executed on a Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz with 8Gb of RAM. Communication between the computer and the motors of the drive system is accomplished using a Controller Area Network (CAN) adapter with 1Mbps maximum bit-rate. The CAN adapter provides both position and velocity control of the individual motors. In all experiments, the control law is as follows:

$$\mathbf{u}=\mathbf{J}^{\dagger}\mathbf{K}_P\odot(\mathbf{y}_{\text{des}}-\mathbf{y}_{\text{current}}) \tag{6}$$

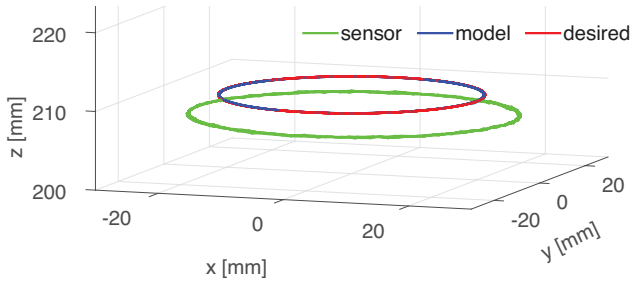
$$\mathbf{J}^{\dagger}=(\mathbf{J}^T\mathbf{J}+\varepsilon\mathbf{I})^{-1}\mathbf{J}^T,\varepsilon>0 \tag{7}$$

where $\mathbf{y}_{\text{des}},\mathbf{y}_{\text{current}}\in\mathbb{R}^6$ represent the goal and current end-

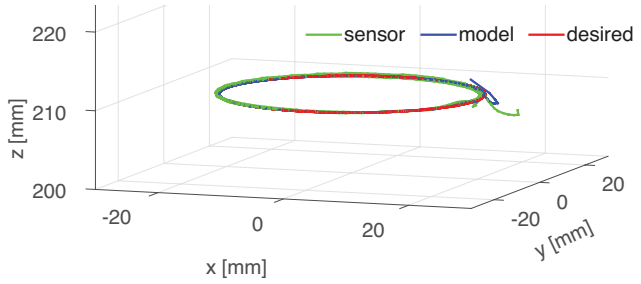
effector pose respectively, $\mathbf{K}_P \in \mathbb{R}^6$ is a vector of gains, $\mathbf{J}^\dagger$ is the damped least-square *pseudo-inverse* of Jacobian $\mathbf{J} \in \mathbb{R}^{6\times5}$, and $\odot$ is the Schur product of two vectors (elementwise multiplication). Damped least squares inverse provides robustness to kinematic singularities [23]. $\mathbf{J}$ is computed numerically by successive perturbation of each joint coordinate. For Jacobian computation, the full configuration is considered; including the rigid body transformation of the base. The configuration space velocity $\mathbf{u} \in \mathbb{R}^5$ is applied to the motors via the CAN card.

To test model accuracy, we performed trajectory-following experiments, in which we provided the robot with time-stamped end-effector target poses. The robot used the learned kinematic model to estimate the current pose of the end-effector. We used two types of trajectories: a circular one on the $x-y$ plane, and a random one that we recorded by teleoperating the robot.

Figure 4a compares the desired, model-predicted and actual robot tip positions when moving on the commanded circle. Since model adaptation is off, there is a consistent error throughout the experiment. When adaptation is turned on, the result is shown in Fig. 4b. It is observed that the model-predicted path, and thus the sensed path, rapidly converge to the desired path.



(a) Adaptation off

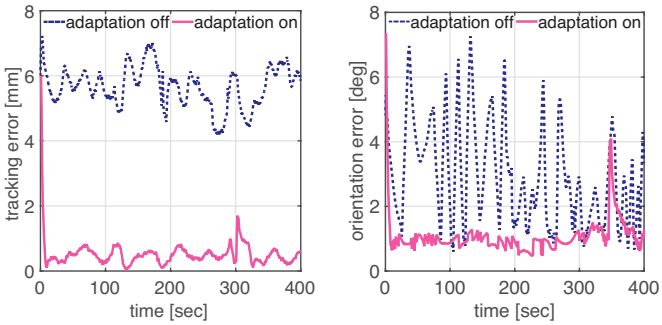


(b) Adaptation on

Fig. 4. Circular trajectory with and without online adaptation.

The speed of convergence is clearly shown in Fig. 5, which depicts model error as a function of time. The position error is less than 1mm for most of the experiment duration. Likewise, the orientation error drops to about one degree.

For a more general trajectory, the robot was moved under teleoperation, while recording tip pose. The resulting path is more convoluted, providing a stronger test for online adaptation. We used the recorded trajectory to perform a second set of experiments. Initially, adaptation is switched



(a) Position (b) Orientation

Fig. 5. Difference between sensor measurements and kinematic model prediction for the circular trajectory.

off. As previously, the initial model estimate deviates from sensor measurements. Figure 6a shows the desired trajectory juxtaposed with the model predictions and the sensor measurements. Next, we switched adaptation back on and repeated the experiment. As shown in Fig. 6b the model predictions, and consequently the actual robot trajectory, match the desired path accurately. Figure 7 shows the error of the kinematic model for the two experiments; online mode adaptation improves the prediction accuracy significantly. Table III summarizes the tracking error for both position and orientation for all experiments.

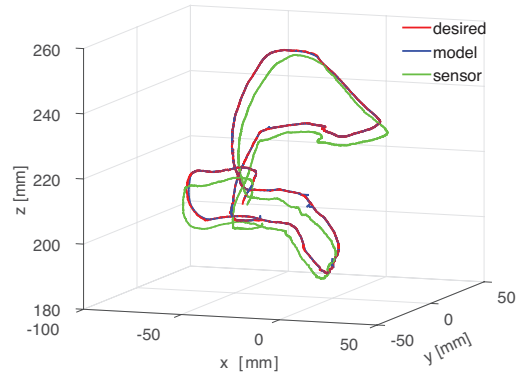
TABLE III
EXPERIMENTAL RESULTS

trajectory	adaptation	x[mm]	y[mm]	z[mm]	θ[deg]
circle	off	3.34	3.03	2.66	2.96
circle	on	0.28	0.33	0.13	1.09
teleop	off	2.01	1.67	2.785	3.39
teleop	on	0.63	0.65	0.38	1.11

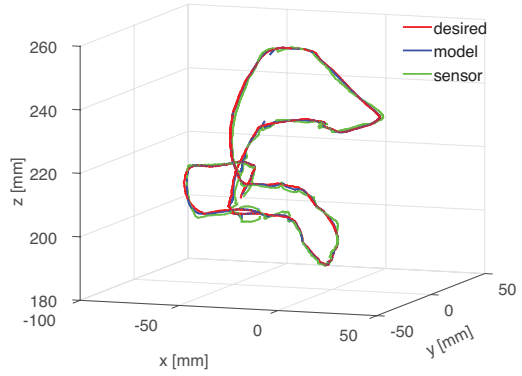
V. CONCLUSIONS

We presented a novel approximate kinematic model for Concentric Tube Robots, based on Locally Weighted Projection Regression, that approximates complex relations with multiple locally linear models. The main strength of our approach is efficient model adaptation due to incremental learning of local model parameters. The model is updated based on sensor information while it preserves goodness-of-fit across the rest of the configuration space. All the above yield an accurate kinematic representation that can account for unmodeled phenomena. Using non-parametric regression eliminates the need to define model order or parameters of any sort; all internal parameters are learned autonomously during training.

A kinematic model for a three-tube CTR was learned and evaluated on experimental data. Adaptation improved accuracy significantly on both a predefined regular shaped (circular) trajectory, as well as on a trajectory generated by teleoperation. This last experiment was performed to ensure that the

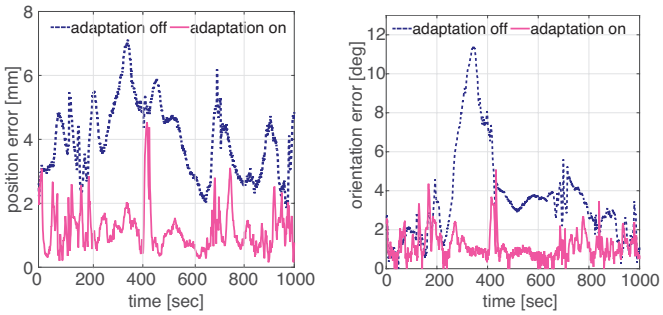


(a) Adaptation off



(b) Adaptation on

Fig. 6. Teleoperation generated trajectory with and without online model adaptation



(a) Position (b) Orientation

Fig. 7. Difference between sensor measurements and kinematic model prediction for a trajectory generated by teleoperation.

algorithm performs well for random non-smooth trajectories generated such as may be generated by a surgeon during a procedure. In both experiments, adaptation substantially reduced model error.

REFERENCES

[1] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric tube robots," *IEEE Trans. Robotics*, vol. 26, no. 2, pp. 209–225, 2010.

[2] R. J. Webster, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Trans. Robotics*, vol. 25, no. 1, pp. 67–78, 2009.

[3] C. Bergeles, A. Gosline, N. V. Vasilyev, P. Codd, P. J. del Nido, and P. E. Dupont, "Concentric tube robot design and optimization based on task and anatomical constraints," *IEEE Trans. Robotics*, vol. 31, no. 1, pp. 67–84, 2015.

[4] C. Baykal, L. G. Torres, and R. Alterovitz, "Optimizing design parameters for sets of concentric tube robots using sampling-based motion planning," *IEEE Int. Conf. Robotics and Automation*, pp. 4381–4387, 2015.

[5] J. Burgner, H. B. Gilber, and R. J. Webster III, "On the computational design of concentric tube robots: incorporating volume-based objectives," *IEEE Int. Conf. Robotics and Automation*, pp. 1185–1190, 2013.

[6] A. H. Gosline, N. V. Vasilyev, E. J. Butler, C. Folk, A. Cohen, R. Chen, N. Lang, P. J. Del Nido, and P. E. Dupont, "Percutaneous intracardiac beating-heart surgery using metal MEMS tissue approximation tools," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1081–1093, 2012.

[7] J. Burgner, D. C. Rucker, H. B. Gilbert, P. J. Swaney, P. T. Russell, K. D. Weaver, and R. J. Webster III, "A telerobotic system for transnasal surgery," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 3, pp. 996–1006, 2014.

[8] R. J. Hendrick, Richard J and Mitchell, Christopher R and Herrell, S Duke and Webster, "Hand-held transendoscopic robotic manipulators: A transurethral laser prostate surgery case study," *The International Journal of Robotics Research*, vol. 34, no. 13, pp. 1559–1572, 2015.

[9] D. C. Rucker, R. J. Webster, G. S. Chirikjian, and N. J. Cowan, "Equilibrium conformations of concentric-tube continuum robots," *The International Journal of Robotics Research*, vol. 29, no. 10, pp. 1263–1280, 2010.

[10] J. Lock and P. E. Dupont, "Friction modeling in concentric tube robots," *IEEE Int. Conf. Robotics and Automation*, pp. 1139–1146, 2011.

[11] J. Lock, G. Laing, M. Mahvash, and P. E. Dupont, "Quasistatic modeling of concentric tube robots with external loads," *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 2325–2332, 2010.

[12] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally-loaded concentric-tube continuum robots," *IEEE Trans. Robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[13] S. C. Ryu and P. E. Dupont, "FBG-based shape sensing tubes for continuum robots," *IEEE Int. Conf. Robotics and Automation*, pp. 3531–3537, 2014.

[14] C. Bergeles and P. E. Dupont, "Planning stable paths for concentric tube robots," *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3077–3082, 2013.

[15] L. G. Torres, A. Kuntz, H. B. Gilbert, P. J. Swaney, R. J. Hendrick, R. J. Webster, and R. Alterovitz, "A motion planning approach to automatic obstacle avoidance during concentric tube robot teleoperation," *IEEE Int. Conf. Robotics and Automation*, pp. 2361–2367, 2015.

[16] C. Kim, S. C. Ryu, and P. E. Dupont, "Real-time adaptive kinematic model estimation of concentric tube robots," *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3214–3219, 2015.

[17] C. Bergeles, F. Lin, and G. Yang, "Concentric tube robot kinematics using neural networks," *Hamlyn Symposium on Medical Robotics*, pp. 1–2, 2015.

[18] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: A scalable method for incremental online learning in high dimensions," 2005.

[19] H. Wold, "Partial least squares," *Encyclopedia of statistical sciences*, 1985.

[20] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[21] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[22] R. S. Sutton, "Adapting bias by gradient descent: An incremental version of delta-bar-delta," in *AAAI*, 1992, pp. 171–176.

[23] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.