Learning a State Transition Model of an Underactuated Adaptive Hand

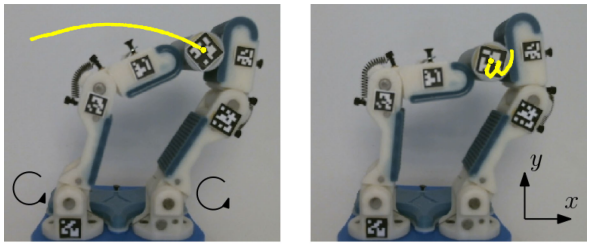
Avishai Sintov, Andrew S. Morgan, Andrew Kimmel, Aaron M. Dollar, Kostas E. Bekris, and Abdeslam Boularias

***Abstract*—Fully actuated multifingered robotic hands are often expensive and fragile. Low-cost underactuated hands are appealing but present challenges due to the lack of analytical models. This letter aims to learn a stochastic version of such models automatically from data with minimum user effort. The focus is on identifying the dominant, sensible features required to express hand state transitions given quasi-static motions, thereby enabling the learning of a probabilistic transition model from recorded trajectories. Experiments both with Gaussian processes (GP) and neural etwork models are included for analysis and evaluation. The metric for local GP regression is obtained with a manifold learning approach, known as *Diffusion Maps*, to uncover the lower-dimensional subspace in which the data lies and provide a geodesic metric. Results show that using Diffusion Maps with a feature space composed of the object position, actuator angles, and actuator loads, sufficiently expresses the hand-object system configuration and can provide accurate enough predictions for a relatively long horizon. To the best of the authors' knowledge, this is the first learned transition model for such underactuated hands that achieves this level of predictability. Notably, the same feature space implicitly embeds the size of the manipulated object and can generalize to new objects of varying sizes. Furthermore, the learned model can identify states that are on the verge of failure and which should be avoided during manipulation. The usefulness of the model is also demonstrated by integrating it with closed-loop control to successfully and safely complete manipulation tasks.**

***Index Terms*—Tendon/Wire Mechanism, Underactuated Robots, Dexterous Manipulation.**

I. Introduction

TRADITIONAL robotic hands, such as the Shadow and the Allegro hands [1], have achieved significant accuracy and performance. Nevertheless, they have a complex structure with many degrees-of-freedom, while they can be large, fragile and costly. They also rely on exact hand-object models, high-fidelity sensors, and sophisticated control and planning to achieve robust manipulation [2], [3]. Previous efforts have depended on tactile sensing to learn, estimate and achieve grasp stability [4]–[6].



(a) (b)

Fig. 1. Traversed paths (in yellow) using manipulation primitives: (a) Primitive of rotating the actuators in the same direction with equal velocities. Circular arrows show the counter-clockwise direction of the actuators' rotation. (b) Rotating the actuators in the opposite direction. The actuators' rotation is switched twice here from clockwise to counter-clockwise. With both primitives, resulting trajectories are clearly non-linear.

On the other hand, underactuated hands with compliant fingers are appealing due to their ability to passively adapt to objects of uncertain size and shape. Therefore, they can provide a stable and robust grasp without tactile sensing or prior planning, and with open-loop control [7]–[10]. In addition, due to the low number of actuators, they enable a low cost and compact design. It has already been demonstrated that such hands can perform precise in-hand manipulations along with stable grasps [11].

An advantageous approach to robotic manipulation is through the use of open source hardware, which is easily altered, fabricated, and distributed for scientific contribution [12]. Due to uncertainties in the manufacturing process, however, fabricated models of the hands differ in size, weight and inertia according to the manufacturing technique. The hand used in this letter, as in Figure 1, is a compliant and underactuated hand that exhibits this property. Due to this uncertainty, hand-crafting precise models for these hands is a significant challenge on top of the difficulty in modeling passively elastic joints in underactuated hands [13]. Some approaches [14], [15] rely on specific contact locations, contact forces, and assumptions in the Coulomb (uniform) friction model to account for reconfiguration of the hand. These parameters are typically not easy to accurately identify, however, in a physical system. Consequently, precise models for such hands are usually unavailable as they are hard to derive analytically or to fine-tune. Along with this, analytical control

Manuscript received September 10, 2018; accepted January 12, 2019. Date of publication January 23, 2019; date of current version February 15, 2019. This letter was recommended for publication by Associate Editor G. Endo and Editor P. Rocco upon evaluation of the reviewers' comments. The work of A. Sintov, K. E. Bekris, and A. Boularias was supported by the National Science Foundation Awards 1734492 and 1723869. *(Corresponding author: Avishai Sintov.)*

A. Sintov, A. Kimmel, K. E. Bekris, and A. Boularias are with the Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ 08854 USA (e-mail: avishai.sintov@gmail.com; akimmel42@gmail.com; kostas.bekris@cs.rutgers.edu; boularias@gmail.com).

A. S. Morgan and A. M. Dollar are with the Mechanical Engineering, Yale University, New Haven, CT 06520 USA (e-mail: andrew.morgan@yale.edu; aaron.dollar@yale.edu).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This video presents the analysis and evaluation conducted on learning the transition model of an underactuated adaptive hand. We show results for predicting the motion of one object, generalization to unknown objects and implementation in closed-loop control.

Digital Object Identifier 10.1109/LRA.2019.2894875

schemes [16], [17] also require joint position sensing, which is often not available to keep costs low. The absence of joint positions is also an obstacle for model identification, which is required to implement analytical control. These challenges, in turn, complicate the control and manipulation planning problems. Given this desire for precision and to cope with the lack of a good model, this work aims to learn this transition model from data under assumptions of quasi-static motion. Thus, this letter provides a data-driven modeling approach, which best fits low-cost 3D printed hands. While it can be time consuming to generate offline, it is easy to implement and provides good results.

To cope with the above challenges and towards applying a classical control method, previous work simplified the problem by providing an approximation of the hand kinematics [18]. Manipulation primitives were introduced such that rotating the two actuators of a planar hand in the same or opposite direction is assumed to move the grasped object along the x- or y- directions, respectively. Motion in other directions was neglected. Nevertheless, when applying these primitives, the object tends to move in non-linear, arc-like trajectories, as shown in Figure 1. Thus, the kinematics derived from this assumption provide only a rough approximation of the motion. In addition, uncertainties such as friction and elasticity were not modeled. The resulting prior approach can only be used in visual servoing closed-loop control [19] with substantial tuning, and not for planning.

This letter focuses on learning a probabilistic transition model of such hands from a captured sequence of trajectories. It is possible to formulate the hand's equations of motion with some oscillating terms due to the springs. Dissipative forces in the joints, however, overwhelm the inertial effects such that they are negligible. In addition, manipulation actions are performed slow enough that the inertias are negligible with lightweight objects, while the natural compliance of the hand provides stable grasps. Thus, this work models quasi-static transitions. The primary objective is to identify what are the dominant features (e.g., object and links positions, actuator angles and torques) of an adaptive hand that are sufficient to express its motion and to provide accurate predictions. Another objective tackled here is to train a model that can generalize to different objects and provide accurate predictions for the manipulation of a new, unknown object. To achieve these objectives, a probabilistic model of motion generated by the manipulation primitives is learned.

Given a state and action, local Gaussian Process (GP) regression on the collected data was used to acquire a Gaussian distribution of the next state. Such probability distributions provide a mechanism to quantify model and state uncertainty. Knowledge of this uncertainty can also be exploited to make more robust predictions. In local GP regression, the function's value at a given test point is predicted from the values of a subset of the training points that are in the vicinity of the test point. The Euclidean distance is frequently used as a metric for choosing the nearest neighbors. A sufficient amount of data is required for accurate regression, which can be computationally expensive. Nevertheless, data corresponding to trajectories of hand-object states lie on a lower-dimensional manifold embedded in the full state space. Therefore, this work proposes the use of a Manifold Learning based Gaussian Process (MLGP) to select nearest neighbors according to the actual distance along the manifold, i.e., the geodesic distance. This letter argues that this serves as a more efficient metric in GP regression for learning transitions of underactuated hands. Furthermore, through simple sparsity analysis, the acquired transition data is used for identification of failure states, i.e., object drop or actuator overload.

II. Related Work

Compliance in underactuated hands makes deriving models non-trivial due to the response of their passively elastic joints and assumptions often made in modeling friction between the hand and the object. Reconfiguration of the gripper is directly dependent on the forces being applied to each of the fingers. When tactile sensing is not available, a precise frictional model is required in order to estimate such forces. It is not feasible to estimate friction at every point on the contact surface in the non-uniform physical world, so typical frictional models assume uniformity [14]. For this reason, along with the inability to control individual joint positions in an underactuated system, feasible models that remain precise in the physical world are difficult to derive. Modeling tools for underactuated manipulation have been introduced in several works [20]–[22], which examine joint configurations, joint torques, and energy with the simplified frictional model. A popular modeling technique applies a hybrid parallel/serial approach using *screw theory*, which further simplifies the derivation for an accurate model and can easily be transferred to the spatial domain [23]. Nevertheless, these proposed modeling techniques have been shown to be sensitive to assumptions in external constraints and are typically only suitable for simulations.

Other derived models have focused on different features of the hand. The free swing trajectory of a 3-link underactuated finger was modeled in accordance to the passively elastic flexure joint [24]. This model can then be used to design fingers for specific tasks by varying the stiffness of the joints. Kinematic models considering synergies of the hand [8] have been considered, as well as "soft synergies", which account for controlling the grasping force [25]. An approach was proposed [17] to control an object with a pair of fully-actuated fingers without object sensing, which required, however, the kinematics of the fingers along with the inertia matrix. For low-cost 3D printed fingers, this approach is not feasible as the physical properties are not accurately available and there is no access to joint positions for loop closure. Model-based stability control is also required [17], while it is naturally obtained with compliant hands.

Different machine learning approaches have been followed to control robotic hands. For example, a model-free approach was proposed which applies tactile sensing with Reinforcement Learning (RL) to learn manipulation motions [13]. The Enhanced Kinematic Model (E-KM) [26] uses a Sparse Online Gaussian Process (SPOG) to iteratively adapt the estimated kinematic model to describe hand motions. This approach was evaluated on a rigid hand, which has high repeatability and precision compared to the soft-material hand studied here. Other work [27], [28] uses RL to learn time varying local linear models to control dexterous hand manipulations. The method is considered partly model-based as it applies a linear regression with a

Gaussian Mixture model prior based on local trajectory samples. It does not provide, however, global learning of the system.

A transition model is a mapping from a given state and action to the next state. Such models are a key component in model-based RL and are often obtained through non-linear regression in a high-dimensional space. Transition modeling can be divided into two classes, deterministic and stochastic. Deterministic models provide the same prediction for a given state and action [29]. As such, a deep neural network was applied to model the dynamics of helicopters [30]. Stochastic models provide a probability distribution over predictions. Tools which generate such models are Dynamic Bayesian Networks (DBN) [31] and the Locally Weighted Bayesian Regression (LWBR) [32], which was applied to helicopter control. The most common approach for stochastic modeling of dynamical systems is the Gaussian Process (GP) [33]. GP has been employed on a range of real-world tasks [34], [35] and has shown efficient learning, but relies on good coverage of the underlying space in the training data. This work uses a GP to learn the transition model of a low-cost, adaptive, 3D-printed hand made with elastic materials. The letter shows that Diffusion Maps [36] can provide an efficient similarity function for the proposed GP model. This work demonstrates long-horizon predictions of the GP that can be helpful for belief space planning of in-hand manipulation, as well as the model's capacity to generalize across objects.

III. Problem Definition

This work considers a two-finger adaptive hand, shown in Figure 1. The fingers are opposed to each other such that the hand achieves planar manipulation. Each finger has two compliant joints with springs. In addition, two actuators provide flexion to the fingers through tendons running along the length of each finger. The gripper also has high friction pads to avoid slipping. This work considers cylindrical objects and leaves other shapes for future work. Based on this design, a model learning problem is defined accordingly.

Let $\mathbf{x} \in \mathbb{R}^n$ be an observable state vector of the hand-object system and $\mathbf{a} \in \mathscr{U}$ be an action taken from a set $\mathscr{U}$ of possible actions. An action is, in practice, unit changes to the angles of the actuators at each time step. That is, an action moves the two actuators with an angle vector of $\delta(\gamma_1, \gamma_2)$ where δ is a predefined unit angle and γ_i is equal to either 1, -1 or 0. The observable state of the system can correspond to different measurable features, such as object position, gripper link positions, actuator positions and torque. At each time-step t, the system is at an observable state $\mathbf{x}_t$ and executes action $\mathbf{a}_t$, resulting in transition to the next state $\mathbf{x}_{t+1}$ according to $f : \mathbb{R}^n \times \mathscr{U} \rightarrow \mathbb{R}^n$. The objective is to learn probability distributions over f from example trajectories. Uncertainty over f is due to hidden state variables that play a role in the transition but cannot be easily measured, as well as a limited amount of data. The task is therefore to find the set of features from the data, which best represents the system and allows to train a low variance transition model that returns $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)$. Such a model can then be used for predictions defined by the probability distribution over the future states given current states and actions. Since quasi-static motion is considered, the state of the hand is defined directly by the proposed set of features. Terms related to object velocity are negligible, and thus, not included.

IV. Learning the Model

This section presents the manipulation primitives used to generate motions. This is followed by a brief presentation of the proposed MLGP approach for learning a model for the hand. Then, the learning procedure and sparsity analysis for avoiding failure states is discussed.

A. Manipulation Primitives

The Precision Manipulation Primitives presented in earlier work [18] define the following operations. The first primitive moves the actuators in the same direction with the same velocities so as to move the object along the x-axis, while rotating it. Similarly, the second primitive moves the actuators in the opposite direction with the same velocity so as to move the object along the y-axis. In both primitives, motion in the second axis is neglected. Under this assumption, a linear approximation of the kinematic model, which maps the object velocity $\mathbf{v}_o$ to the required actuator velocity $\dot{\mathbf{q}}$, was proposed:

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{K_x} & \frac{1}{K_y} \\ -\frac{1}{K_x} & \frac{1}{K_y} \end{bmatrix} \mathbf{v}_o, \tag{1}$$

where K_x and K_y are constant scalars related to the hand.

As can be seen in Figure 1, the primitives cannot be distinctively associated to any of the axes, and in practice, different non-linear motion patterns can be observed in different states of the system. In addition, motion is not repetitive. Applying a primitive in one direction for some time and switching to the other direction does not ensure backtracking along the same path (as shown in Figure 1b). The hand kinematics, as well as uncertainty due to elasticity and friction, define a non-linear pattern of motion, which should be taken into account. Thus, a model of this non-linear behavior is required to capture the motion that results from applying these manipulation primitives.

B. Training Data

Training and test data sets comprise of observable state-action trajectories, with no additional labeling required. The training set is acquired automatically by executing random actions, while recording the observable states. Thus, the resulting data is a set of state-action trajectories $\mathscr{P} = \{(\mathbf{x}_0, \mathbf{a}_0), \ldots, (\mathbf{x}_k, \mathbf{a}_k)\}$. The trajectories in $\mathscr{P}$ are pre-processed to a set of training inputs $(\mathbf{x}_i, \mathbf{a}_i)$ and corresponding output labels of the next state $\mathbf{x}_{i+1}$ to define $\mathscr{T} = \{(\mathbf{x}_i, \mathbf{a}_i), (\mathbf{x}_{i+1})\}_{i=1}^N$. The data points are then normalized by the minimum and maximum values in the set, such that each feature is in the interval [0, 1].

C. Gaussian Processes Model

The transition model is learned by using a GP [33], which is a non-parametric regression method. The model is briefly presented here. Let k be a kernel function, $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, for measuring the similarity between two states in $\mathbb{R}^n$. Let $X_N = (x_1, x_2 \ldots, x_N)$ be a set of observable hand-object states, and

let $Y_N = (y_1, y_2 \ldots, y_N)$ be the set of observable hand-object states resulting from applying a given action in the states in X_N, where $y_i = f(x_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is independent white noise. The kernel *Gram matrix* K_{X_N, X_N} is an $N \times N$ positive-definite matrix, defined as $K_{X_N, X_N}(i, j) = k(x_i, x_j)$, for $i, j \in \{1, \ldots, N\}$. Given X_N and Y_N, the posterior distribution on the values of f in any set of states $\tilde{X} = (\tilde{x}_1, \tilde{x}_2 \ldots, \tilde{x}_m)$ is a Gaussian with mean vector $\mu_{\tilde{X}}$ and covariance matrix $\Sigma_{\tilde{X}, \tilde{X}}$. A vector of values of f at states $\tilde{X}$ can be sampled by drawing a vector $\psi \sim \prod \mathcal{N}(0, 1)$ of independent, one-dimensional, standard Gaussian random values. Then, the sampled vector of observable state transitions is:

$$\tilde{Y} = \mu_{\tilde{X}} + C\big(\Sigma_{\tilde{X}, \tilde{X}}\big)\psi^T, \tag{2}$$

where $C(\Sigma_{\tilde{X}, \tilde{X}})$ is the Cholesky upper matrix of $\Sigma_{\tilde{X}, \tilde{X}}$:

Kernel k is given by the Squared Exponential Kernel,

$$k(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j)\|_w}{2\tau^2}\right), \tag{3}$$

where τ is the bandwidth of the kernel, and $\|.\|_w$ is a weighted Euclidean distance with weight vector $w \in \mathbb{R}^n$. Bandwidth τ and noise variance σ^2 are optimized by maximizing the likelihood of the training data [33].

D. Gaussian Process with Manifold Learning

The computational bottleneck in evaluating the GP is computing the Cholesky decomposition of the covariance function. Thus, the computational complexity scales cubically $O(N^3)$ with the size of the dataset N, making often global regression infeasible [37]. Nearest-neighbor GP provides a scalable alternative by using local information for regression [38]. Only data points in some proximity of the query point are used to make a prediction. Nevertheless, the question of what is the appropriate metric to choose nearest neighbors for regression arises. One option is to use a simple Euclidean metric for the GP (EGP). The reachable hand-object states, however, lie on a lower-dimensional manifold in the state space, and therefore, samples that seem close in the state space may be far from each other across the manifold. One method that has evolved to help solve such problems is based on nonlinear dimensionality reduction of data in high-dimensional spaces. This is achieved by sampling from the observable state space to a lower dimensional subspace and then perform nearest neighbor search in the later.

This work proposes the Manifold Learning-based GP (MLGP), which utilizes *Diffusion Maps* to learn the distance along the manifold [36]. Diffusion maps are a graph-based dimensionality reduction method. Specifically, a diffusion map is an embedding in the Euclidean space $\mathbb{R}^n$ and thus, the diffusion distance inherits all the metric properties of $\mathbb{R}^n$. Therefore, the diffusion map is an efficient method for acquiring a metric subspace corresponding to the non-linear manifold in the Euclidean space. The general idea is to find the underlying manifold that the data has been sampled from.

Suppose there are N samples. For approximating the probability of transition from one sample to the next, a Gaussian kernel function similar to (3) can be defined, where τ is now a decaying rate. This kernel establishes prior local geometry information of the samples. In order to have a probability function of taking a step from the i-th sample to the j-th sample, the reversible Markov chain is calculated on the dataset as:

$$\text{connectivity}(\mathbf{x}_i, \mathbf{x}_j) = p(\mathbf{x}_i, \mathbf{x}_j) = \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sum_i k(\mathbf{x}_i, \mathbf{x}_j)}. \tag{4}$$

The normalized diffusion matrix P can now be defined such that $P_{i,j} = p(\mathbf{x}_i, \mathbf{x}_j)$. Each component $P_{i,j}$ encapsulates the local knowledge of the connectivity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Dimensionality reduction is done by neglecting certain dimensions in the diffusion space of the normalized diffusion matrix P. In practice, the eigenvectors of P are found, choosing only the $m < N$ dimensions associated with the dominant eigenvectors of P, and map $\mathbf{x}_i$ to $\mathbb{R}^m$.

Since a diffusion map cannot be generated for the large training data, local maps are built instead. Given a query state-action pair $(\mathbf{x}, \mathbf{a})$, the K_1-nearest neighbors in training data $\mathscr{T}$ are found. The pair $(\mathbf{x}_h, \mathbf{a}_h)$ is also found in $\mathscr{T}$ that is the closest to $(\mathbf{x}, \mathbf{a})$ and included in the K_1 samples. Next, a diffusion map for the K_1 samples is built, acquiring a reduced representation $\mathscr{E} \subset \mathbb{R}^n \times \mathscr{U}$ of the data in the lower-dimensional subspace. Finally, the K_2 ($K_2 << K_1$) closest points in $\mathscr{E}$ to the reduced representation of $(\mathbf{x}_h, \mathbf{a}_h)$ are chosen and GP regression is performed on these K_2 points.

E. Failure Avoidance

With the same collected dataset, classification is performed to identify failure modes in the observable state space. Voids or low-density regions in the space are assumed to be not well explored due to failures on their boundaries. Failure is defined as the occurrence of either losing contact with the object, i.e., dropping it, or reaching actuator torque overload (OL). It is important to note that an observable state itself is not sufficient to determine failure since at a certain state, not all actions lead to failure. Thus, observable state-action pairs are examined for failure.

Failure state-actions can be considered as obstacles for planning and control purposes, and observable states on the verge of failure can be identified by simple sparsity analysis. That is, given a query state-action pair, nearest neighbors are found using the weighted Euclidean metric, forming an ellipsoid in the observable state space. A lower bound threshold on the number of nearest neighbors is then applied to reject observable states that are close to failure. The appropriate values for the weight matrix and the threshold can be determined using a labeled test set as will be shown in the evaluation section.

V. Evaluation

Training data was collected using the three-finger Model O underactuated hand [12], modified to use only two opposing fingers (Fig. 2). To generate the training data, trajectories were recorded for five cylinders of different diameters. Each object was manipulated by randomly applying the defined set of possible actions in $\mathscr{U}$ manually via keyboard. Such process takes 5-6 hours and future work will focus on automating it. The positions of the objects as well as the configuration of the hand links were acquired using a camera mounted above the hand tracking

TABLE I
RMSE (MM) FOR THE TEST TRAJ. PREDICTIONS WITH REFERENCE TO THE GROUND TRUTH TRAJ

Feat. conf.	Traj. 1				Traj. 2				Traj. 3				$t_{EGP_{1000}}/t_{MLGP}$
	EGP_{100}	EGP_{1000}	MLGP	NN	EGP_{100}	$EGP_{10.0}$	MLGP	NN	EGP_{100}	EGP_{1000}	MLGP	NN	
1	6.49	3.95	3.87	5.72	6.36	5.98	4.94	6.24	16.04	22.28	19.57	18.76	25.41
2	17.73	9.73	16.8	4.55	6.67	14.37	8.04	5.25	11.02	110.34	17.11	23.44	32.32
3	10.52	67.31	17.0	8.62	18.28	4.18	96.3	5.56	20.6	10.82	333.56	19.71	32.72
4	16.87	4.88	7.7	4.00	10.34	5.86	5.11	3.34	4.49	6.96	8.53	36.48	28.19
5	8.67	2.36	2.07	2.61	8.57	3.97	3.81	4.32	18.93	2.53	1.71	8.74	27.0
6	7.72	3.12	2.98	2.84	9.32	3.47	2.97	4.27	9.81	4.53	2.71	4.03	25.14
7	-	2.89	2.35	2.75	-	2.28	2.39	2.35	-	3.0	4.72	5.3	32.04
Path length (mm)		95.46				82.91				184.45			
Motion time (s)		94.9				87.8				140.9			

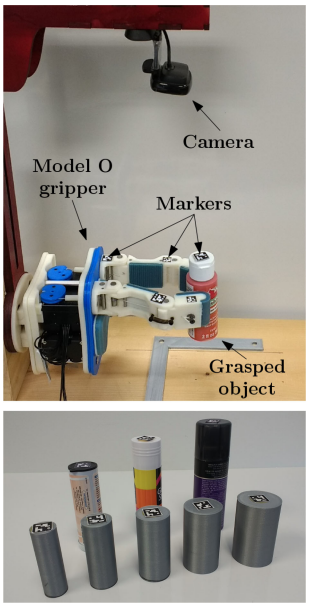


Fig. 2. (top) Setup. (bottom) Training objects (lower row) and test objects (upper row).

fiducial markers on the moving parts. For simplification, the experiments first consider four actions on the planar workspace derived from the two manipulation primitives defined in prior work [18] to move the object up, down, right and left (i.e., vector (γ_1, γ_2) is $(-1,-1), (1,1), (-1,1)$ and $(1,-1)$, respectively). There are also experiments presented with eight actions, which require more data for training. Actuator step size at each time step is $\delta = 0.8°$. During motion, the following aspects were recorded at 15 Hz data streams:

- the poses of the object and four links of the arms,
- the base pose to compensate for camera movements,
- and, the angles and loads of the actuators.

The experiments analyze 7 possible feature configurations:

1) Object position (2 dim.).
2) Object position and distal links position (6 dim.).
3) Object position and all links positions (10 dim.).
4) Object position and actuators angles (4 dim.).
5) Object position and actuators loads (4 dim.).
6) Object position and, actuators angles and loads (6 dim.).
7) All measured features (14 dim.).

A. *Predicting Trajectories of One Object*

The first evaluation studies the prediction model for one cylinder of 20 mm diameter. The transition data for the specific object comprised of approximately 230, 000 transition points. An additional three test trajectories were manually collected (via keyboard), and were not included in the training set. Given the start state and the sequence of actions from each test trajectory, the model was used in an open loop fashion. This process first considers only the mean of the GP predictions. The values $K_1 = 1000$ and $K_2 = 100$ were chosen for the diffusion maps and manual analysis found that reduction of the data to three ($m = 3$) dimensions provides the best results. Regression in EGP is performed with 100 (EGP_{100}) and 1, 000 (EGP_{1000}) nearest neighbors. In addition and for comparison, results are also presented for a feed-forward Rectified Linear Unit (ReLU) Neural-Network (NN) trained with the same data. The architecture of the NN was optimized. Two hidden layers and 72 neurons each gave the best results.

Table I summarizes the root mean squared errors (RMSE) for predictions, while considering the different feature configurations. The RMSE measures the accumulated error between corresponding points along the reference and predicted trajectories. As can be seen, results are poor when considering only the object position (feature conf. 1) as it cannot completely define the state of the hand. In feature configurations 2-3 the position of the hand along with positions of the links do not define the state well. The hand and object can be at the same geometrical configurations but with different tensions of the tendons, which could produce different patterns of motion. Similarly, actuator angles in 4 by themselves do not always produce better results. On the other hand, actuator loads in 5 sufficiently embed the geometric configuration of the hand through the tension on the tendons. Moreover, actuator loads along with the angles in feature configuration 6 provide more consistent accurate predictions. Feature configuration 7 generally produces good results but not consistently. Due to the high-dimensional nature of the space (14-D), it requires a lot of data. Feature configuration 6 is preferred due to its consistent accuracy. Furthermore, it is clear that using MLGP provides much better results than EGP_{100} and NN, and slightly better than using EGP_{1000}. However, Table I also shows the average ratio of one prediction between the runtimes of EGP_{1000} and MLGP ($t_{EGP_{1000}}$ and t_{MLGP}, respectively), showing that MLGP is substantially more efficient in runtime.

Figure 3 shows three predicted trajectories with feature configuration 6. As can be seen, error is accumulated along the path. Figure 4 shows the RMSE of the predicted mean trajectory using the sequence of actions of trajectory 1, as a function of the path's length. The figure shows that for MLGP and feature configurations 5-6, prediction errors are low, especially for a short horizon. That is, the model can be used in

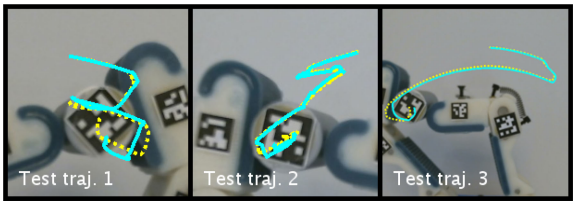


Fig. 3. Three test trajectories when manipulating a 20 mm cylinder and for MLGP. Actual recorded path (dashed yellow) and Predicted path (cyan).

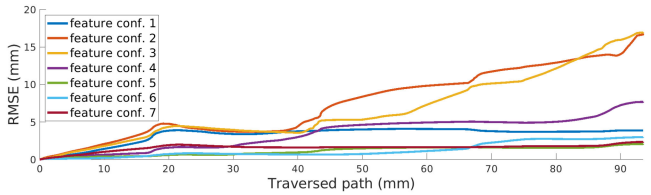


Fig. 4. RMSE of predictions along test trajectory 1 in Figure 3 using the proposed MLGP technique and seven different feature configurations.

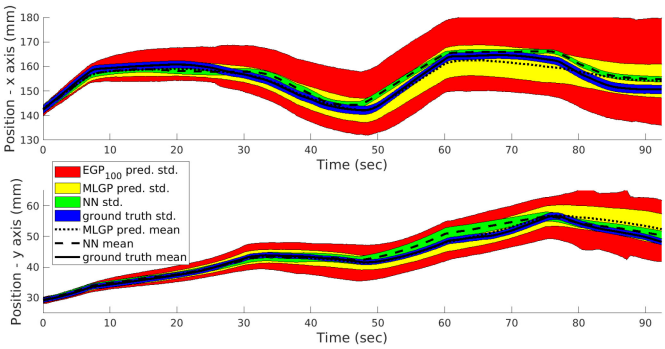


Fig. 5. Distribution propagation (mean and standard deviation) for the prediction of test trajectory 1 while sampling from the GP distribution.

closed-loop control or in a model predictive control scheme. When comparing between MLGP and NN, it can be seen that NN can generally apply better regression with different feature configurations. The NN observes global data, while MLGP performs local regression. Nevertheless, MLGP provides lower errors for an extended horizon in all trajectories and with feature configuration 6.

So far only the mean of the predicted next state provided by the GP has been used. The GP, however, provides a Gaussian distribution of the next state. Thus, the state distribution is propagated when predicting for some horizon. The following compares the distribution of a propagated prediction compared to the distribution of the actual motion. For that matter, the same set of actions of test trajectory 1 are applied from approximately the same starting state for the actual hand. This was repeated ten times and the paths were recorded. Next, a starting position is sampled from the distribution of the true trajectories and EGP and MLGP is executed with feature configuration 6 for the set of actions. The next state is sampled from the Gaussian distribution of the predicted next step, thereby propagating the distribution of the predicted states rather than just the mean. The same was performed for NN but with deterministic predictions. Figure 5 shows the mean and standard deviation of 100 predicted trajectories, for EGP_{100}, MLGP and NN. It is clear that MLGP provides significantly more accurate propagation of the distribution compared to EGP_{100}. In addition, the actual distribution is contained in the MLGP distribution. Uncertainty increases along the path as expected. Nevertheless, the uncertainty is very low for a long horizon and indicates that planning using MLGP is feasible for a fairly long horizon with low uncertainty. For NN, there is some deviation from the actual distribution, though it remains relatively close.

TABLE II
RMSE (MM) OF USING MLGP, NN AND (1), AND FOR ONE STEP PREDICTIONS

	Traj. 1	Traj. 2	Traj. 3
Eq. (1) (from [18])	2.38	2.43	2.47
MLGP	0.07	0.077	0.1
NN	0.036	0.034	0.04

B. Comparison to Manipulation Primitives

This section compares the predicted model with the linearized kinematic model proposed in prior work [18] and mentioned in (1). The training set was first used to find the optimal K_x and K_y values that best match state-actions to their corresponding next states. This was done by minimization (with Matlab's `fminsearch` function) of the accumulated squared error. The optimal values are $K_x = 0.9757$ and $K_y = 1.5315$. Then, for each state along the test trajectories, a prediction is made corresponding to the recorded action and sampling rate. Table II shows the RMSE between predictions with MLGP, NN and (1), and the real next state. Feature configuration 6 is used in the GP. MLGP and NN predictions are shown to be significantly more accurate than using (1). NN is observed to be a bit more accurate than MLGP for single steps as it preserves the step size length better. On the other hand, MLGP preserves the direction better and thus accumulates less error as indicated in the previous section.

C. Generalization to Objects of Different Size

This section examines the ability of the proposed combination of features to generalize to unknown objects. A generic set of objects of certain sizes is considered during training, where 160, 000 transition points were collected for each of the five cylinders and have diameters 20, 25, 30, 35 and 40 mm. Then, three test objects were picked: butter can, glue stick and hairspray, of diameters 26, 30 and 36 mm, respectively. All objects are seen in Figure 2.

For evaluation purposes, each test object was tested using training data, while omitting the cylinder with the closest diameter to it. In addition, each training point was associated with the distance between the markers of the distal links at its respected initial grasp. The initial grasp distance is a real-time measurement of the diameter plus some constant. The diameter of the object can be directly taken but is, in practice, not available on the fly given an unknown object. The initial grasp for each object is relatively constant with a low variance. Therefore, for any new object, the hand closes on it, measures the initial grasp distance and chooses the the data points for training based on nearest-neighbor query of this one dimensional set, i.e.,

TABLE III
ROOT MEAN SQUARED ERROR FOR TRAJECTORY PREDICTIONS OF UNTRAINED TEST OBJECTS

Test obj. (dia.)	Butter can (26)		Glue-stick (30)		Hair-spray (36)	
	MSE (mm)					
Action seq.	1	2	1	2	1	2
F. conf. 1	9.63	6.76	11.61	7.05	7.73	14.16
F. conf. 2	28.97	27.5	24.4	17.7	8.64	41.6
F. conf. 3	297.08	-	237.53	62.94	20.56	137.42
F. conf. 4	9.59	5.05	13.08	12.96	8.25	17.03
F. conf. 5	5.06	7.04	3.78	53.8	6.0	8.33
F. conf. 6	4.77	4.08	4.27	4.43	3.03	4.76
F. conf. 7	3.80	27.48	29.02	5.68	2.70	145.33
Path len. (mm)	137.9	79.19	134.15	69.09	134.87	70.55
Motion time (s)	109.2	93.0	96.53	89.6	107.0	93.8

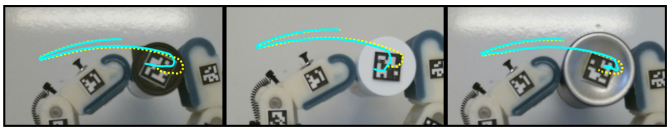


Fig. 6. MLGP predictions for three novel objects that have not been used in training (left) butter can, (middle) glue stick and (c) hair-spray.

TABLE IV
CLASSIFICATION RESULTS FOR SEVERAL FEATURE CONFIGURATIONS

Feat. conf.	Drop score	OL score	Normal score	Tot. score
1	100%	58.3%	100%	91.67%
5	94.4%	100%	83.33%	90%
6	100%	100%	100%	100%

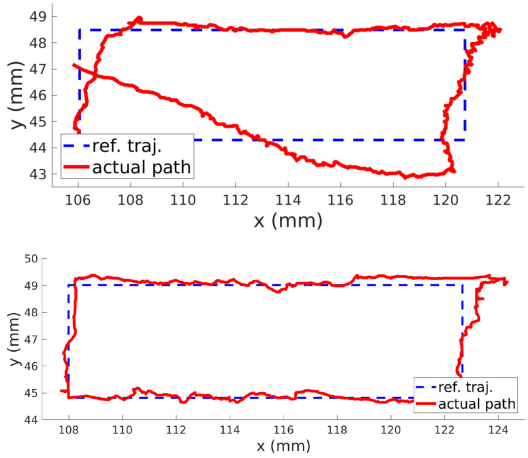


Fig. 7. Closed loop tracking with (top) visual servoing and (1) [18] and (bottom) learned model predictions. Motion is from the bottom left corner and going clockwise.

TABLE V
CLOSED-LOOP TRACKING RESULTS

		visual servoing	learned model
Rectangle	Ref. path length (mm)	9.89	
	RMSE (mm)	0.78	0.53
	Max error (mm)	2.9	1.6
Ellipse	Ref. path length (mm)	26	
	RMSE (mm)	0.34	0.26
	Max error (mm)	0.96	0.66

appropriate training data is automatically chosen based on the initial grasp distance.

For each test object, a recorded test trajectory was retraced by MLGP predictions as performed in Section V-A. Two sequences of actions are considered for all three tested objects resulting in a similar trajectory pattern but, due to different object sizes, with different feature values. Table III presents the RMSE for predicting the test sequences of the untrained objects. Comparing to the results in Table I, the prediction errors are slightly larger than for predicting with one object but still relatively low, mostly in the short horizon (Figure 6), when considering object position and gripper state. By observing the results for feature configurations 5 and 6, the load on the gripper plays an important role in the generalization and can embed the size of the object along with its future reaction to an action.

D. Failure Classification

The data for the 20 mm cylinder were used to evaluate the failure identification accuracy by sampling and labeling 30 normal state-actions and 30 state-actions on the verge of failure. Within the failure state-actions, half failed due to lose of contact, and half due to reaching torque limit. For the classification, different feature configurations were considered as previously. The data suggest that there is a clear density distinction between normal and failing state-action pairs. Based on this, a lower bound threshold can provide the best classification results. Table IV shows the classification results for three feature configurations. Feature configuration 6 is again the most representative of the hand's state, and can provide highly reliable classification. This simple sparsity classifier is used next for closed-loop control.

VI. APPLICATION

This section utilizes the transition model in closed-loop control for tracking a reference path given a 20 mm cylinder and trained with eight possible actions. The actions correspond to the eight cardinal directions of the object applied through the manipulation primitives of Eq. (1). For this purpose, 360, 000 transition points were acquired. As indicated in Figure V-B, NN is better for one step predictions and thus, is more appropriate for closed-loop control. So, the feed-forward ReLU NN presented previously is used in this process.

The model-based tracking is compared against the visual servoing approach proposed in [18]. In both approaches tracking was performed with a moving intermediate goal point, i.e., a simple "follow-the-carrot" scheme, where the intermediate point is moved along the path as the object advances. With visual servoing, Eq. (1) is directly used based on the desired direction of the object toward the intermediate goal point. With the learned model and in each step, feedback of the system's configuration (with feature configuration 6) was taken, and predictions are made for each feasible action. The action with the closest prediction to the intermediate point is applied. Actions that fail the sparsity classification are declared infeasible and not considered. The sparsity check for each action is performed prior to predictions and therefore, computational efficiency is achieved. Figure 7 shows tracking of both approaches along a rectangle starting from approximately the same object position. Table V presents the average RMSE for tracking a rectangle and an ellipse. Tracking with the learned model along with the sparsity classifier is shown to be more accurate. The visual servoing tracking can be improved by manually adjusting some

parameters but using the learned model predictions requires no adjustments while providing high accuracy.

VII. Conclusions

This letter evaluates the features required to sufficiently embed the state of a planar adaptive gripper. It is shown that the position of the grasped object, along with actuator angles and loads, provide the most accurate predictions. This observation is valuable since it permits closed-loop control with only external sensing of the state of the object. These features also embed information of the size of the manipulated object, and allow generalization to new objects of unknown size. Furthermore, the same training data allow to classify states on the verge of failure. Additionally, this work proposed to use manifold learning to acquire a geodesic metric for nearest-neighbors search, which showed significant performance improvement compared to a standard Euclidean metric. A closed-loop control application demonstrated the importance of a learned model for an underactuated hand.

Future work will involve applying such transition model in policy search algorithms. Also, since GP regression provides an uncertainty distribution over next states, belief space planning with asymptotic optimality would be applied to plan the most robust trajectory, i.e., the trajectory with the lowest uncertainty. It is also interesting to consider learning a model for spatial manipulation with an adaptive hand of more than two fingers. In such case and where manipulation primitives are not easy to derive, learning would be coupled with the search of an optimal policy for feasible manipulations. Such learning would be automated in a pick, learn until drop and repeat scheme.

References

[1] J. Bae, S. Park, J. Park, M. Baeg, D. Kim, and S. Oh, "Development of a low cost anthropomorphic robot hand with high capability," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4776–4782.

[2] K. Hertkorn, M. A. Roa, and C. Borst, "Planning in-hand object manipulation with multifingered hands considering task constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 617–624.

[3] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 1560–1565, 2014.

[4] H. Dang and P. K. Allen, "Learning grasp stability," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 2392–2397.

[5] Y. Bekiroglu, J. Laaksonen, J. A. Jorgensen, V. Kyrki, and D. Kragic, "Assessing grasp stability based on learning and haptic data," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 616–629, Jun. 2011.

[6] T. Yamada, T. Taki, M. Yamada, Y. Funahashi, and H. Yamamoto, "Static stability analysis of spatial grasps including contact surface geometry," *Adv. Robot.*, vol. 25, no. 3/4, pp. 447–472, Jan. 2011.

[7] A. M. Dollar and R. D. Howe, "The highly adaptive SDM hand: Design and performance evaluation," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 585–597, 2010.

[8] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi, "On the manipulability ellipsoids of underactuated robotic hands with compliance," *Robot. Autom. Syst.*, vol. 60, no. 3, pp. 337–346, 2012.

[9] D. M. Aukes *et al.*, "Design and testing of a selectively compliant underactuated hand," *Int. J. Robot. Res.*, vol. 33, no. 5, pp. 721–735, 2014.

[10] L. U. Odhner and A. M. Dollar, "Stable, open-loop precision manipulation with underactuated hands," *Int. J. Robot. Res.*, vol. 34, no. 11, pp. 1347–1360, Sep. 2015.

[11] B. Calli, K. Srinivasan, A. Morgan, and A. M. Dollar, "Learning modes of within-hand manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 3145–3151.

[12] R. R. Ma and A. M. Dollar, "Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption," *IEEE Robot. Autom. Mag.*, vol. 24, no. 1, pp. 32–40, Mar. 2017.

[13] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2015, pp. 121–127.

[14] K.-T. Yu, M. Bauzá, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 30–37.

[15] R. R. Ma, W. G. Bircher, and A. M. Dollar, "Toward robust, whole-hand caging manipulation with underactuated hands," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 1336–1342.

[16] S. Arimoto, K. Tahara, J.-H. Bae, and M. Yoshida, "A stability theory of a manifold: Concurrent realization of grasp and orientation control of an object by a pair of robot fingers," *Robotica*, vol. 21, no. 2, pp. 163–178, 2003.

[17] R. Ozawa, S. Arimoto, S. Nakamura, and J.-H. Bae, "Control of an object with parallel surfaces by a pair of finger robots without object sensing," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 965–976, Oct. 2005.

[18] B. Calli and A. M. Dollar, "Vision-based precision manipulation with underactuated hands: Simple and effective solutions for dexterity," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 1012–1018.

[19] B. Calli and A. M. Dollar, "Vision-based model predictive control for within-hand precision manipulation with underactuated grippers," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 2839–2845.

[20] T. Laliberté and C. M. Gosselin, "Simulation and design of underactuated mechanical hands," *Mech. Theory*, vol. 33, no. 1/2, pp. 39–57, Jan. 1998.

[21] L. U. Odhner and A. M. Dollar, "Dexterous manipulation with underactuated elastic hands," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 5254–5260.

[22] A. Rocchi, B. Ames, Zhi Li, and K. Hauser, "Stable simulation of underactuated compliant hands," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 4938–4944.

[23] J. Borras and A. M. Dollar, "A parallel robots framework to study precision grasping and dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 1595–1601.

[24] I. Hussain *et al.*, "Modeling and prototyping of an underactuated gripper exploiting joint compliance and modularity," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2854–2861, Oct. 2018.

[25] G. Grioli, M. Catalano, E. Silvestro, S. Tono, and A. Bicchi, "Adaptive synergies: An approach to the design of under-actuated robotic hands," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 1251–1256.

[26] Y. Su, Y. Wu, H. Soh, Z. Du, and Y. Demiris, "Enhanced kinematic model for dexterous manipulation with an underactuated hand," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2493–2499.

[27] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 156–163.

[28] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 378–383.

[29] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *J. Intell. Robot. Syst.*, vol. 86, no. 2, pp. 153–173, May 2017.

[30] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 3223–3230.

[31] T. T. Nguyen, Z. Li, T. Silander, and T.-Y. Leong, "Online feature selection for model-based reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, vol. 1, pp. 498–506.

[32] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2001, vol. 2, pp. 1615–1620.

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA, MIT Press, 2005.

[34] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 742–747.

[35] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 3876–3881.

[36] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 5–30, 2006.

[37] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 380–385.

[38] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, "Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets," *J. Amer. Statist. Assoc.*, vol. 111, no. 514, pp. 800–812, 2016.