Data Driven Inverse Kinematics of Soft Robots using Local Models

Fredrik Holsten[2], Morten Pol Engell-Nørregård[2], Sune Darkner[1], and Kenny Erleben[1]

***Abstract*— Soft robots are advantageous in terms of flexibility, safety and adaptability. It is challenging to find efficient computational approaches for planning and controlling their motion. This work takes a direct data-driven approach to learn the kinematics of the three-dimensional shape of a soft robot, by using visual markers. No prior information about the robot at hand is required. The model is oblivious to the design of the robot and type of actuation system. This allows adaptation to erroneous manufacturing. We present a highly versatile and inexpensive learning cube environment for collecting and analysing data. We prove that using multiple, lower order models of data opposed to one global, higher order model, will reduce the required data quantity, time complexity and memory complexity significantly without compromising accuracy. Further, our approach allows for embarrassingly parallelism. Yielding an overall much more simple and efficient approach.**

I. Introduction

Fabrication and manufacturing of soft robots is available to everyone and does not require much more than what can be found in most households [1]. The robots are inexpensive to make and the designs are simple. The challenge lies in controlling the motion of the robots. Often, the control methods have a specialized component for the actuation system, whether it is embedded networks of pneumatic chambers [2], shape memory alloy (SMA) springs [3] or cables [4]. Moreover, extensive knowledge about the design and the properties of the soft robots and actuator system is often incorporated into the models [5], [6], [7]. This makes it challenging for researchers to implement the methods. We propose a control method that is completely oblivious to the kind of robot used and its actuation system. Figure 1 depicts a sample of possible robots that can be used with our method.

In Figure 2 we demonstrate how our own kitchen table manufactured soft robot can be controlled using a data-driven approach to create multiple lower order models of the robot motion. Shape vectors are extracted from multiple RGB-D sensors at different viewpoints. Polynomial regression is used to learn the kinematics of the shape with respect to the configuration parameters. The inverse kinematics problem is solved by optimization methods to produce the optimal configuration for a desired pose. Validation of the models shows that the mean configuration error is typically smaller than 8 motor steps, corresponding to less than 0.5 millimeters of cable length.

[1]Department of Computer Science, University of Copenhagen, Denmark. darkner@di.ku.dk, kenny@di.ku.dk.
[2]The Alexandra Institute, Denmark. fredrik.holsten@alexandra.dk, morten.engell-norregard@alexandra.dk.

Fig. 1: Possible DIY soft robots that can be used with our approach.

We specifically target controlling motion of DIY soft robots that would suffer from reality gap challenges with conventional methods based on finite element analysis, simply due to the poor or faulty manufacturing. For instance our robots suffer from degassing and poor mold building.

The contributions of this work can be summarized as follows

- A novel data-driven model for inverse kinematics of soft robots.
- No prior knowledge about the robot or actuation system is required.
- A decomposition method into local models and strategies for local model selections are validated.
- Results prove that local low order models outperform global higher order models.
- An experimental platform, the Learning Cube, that is easy accessible and inexpensive, supported by open source software and open access guides.

II. Related Work on Controlling Soft Robots

The choice of the control method is often determined by the soft robotic design. For instance, control of continuum robots, which are inspired by elephant trunks and snakes, rely on the piecewise constant curvature (PCC) assumption. This is robot specific, since they integrate the morphology of the robots [8]. Another popular approach for learning the controls of soft robots is to create a digital twin that forms the basis for a forward simulation of the physical robot. Optimization methods are used to solve the inverse kinematics and sub-millimetre precision has been reported by numerous groups [4], [9]. The FEM approach has been proven to be feasible for realtime simulation [10], [11], [5]. However, the digital twins have to be fitted with viscoelastic properties and exact measurements of mass and dimension. This information may not be accessible and our method does not need it.

Decomposing the problem into multiple sub-problems to enhance speed and accuracy has been investigated, but while Bosman and colleagues [12] decompose domains of

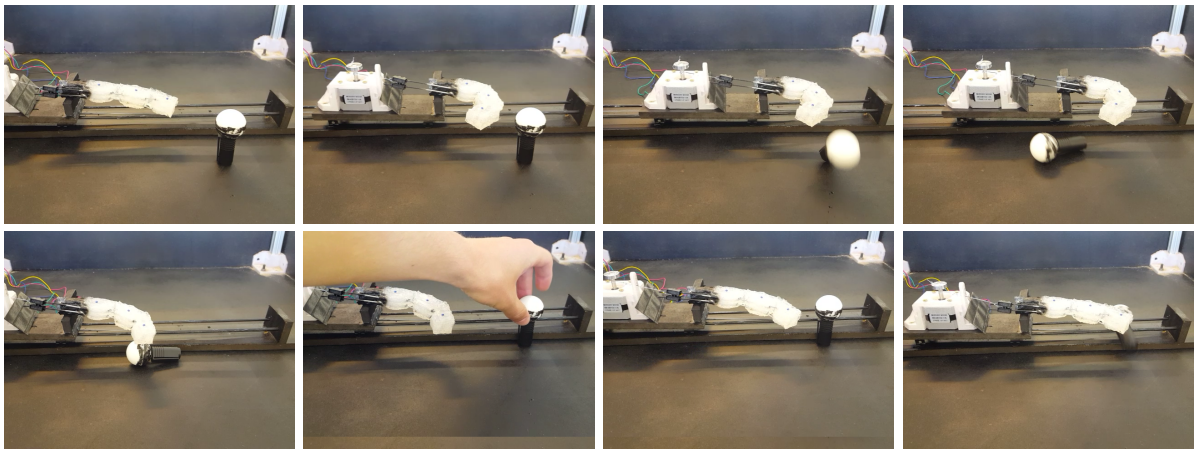


Fig. 2: A DIY soft robotics finger tips a ball over using multiple local lower order models created from direct data of shape deformations. The approach is more computational efficient than using one global model, and yields the same accuracy.

the physical robot, we decompose the configuration space. Moreover, their method is specialised for continuum robots.

Robot and actuator specific models are challenging, not only because of the high number of parameters, but because it is hard to make an ideal representation of a robot reflect the real world accurately. Machine learning techniques have been used to learn the non-linear kinematics of a soft robot directly from data. For instance Giorelli and colleagues [13] use feed-forward Neural Networks (FFN) to train an octopus-like soft robot. Chen and Lau used k-nearest neighbours regression (KNNR) to learn the inverse kinematics of a tendon driven manipulator. KNNR is interesting because of its high learning ability, but requires the training data to be stored to make predictions [14]. This is costly with respect to memory. Taylor et al use fourth order polynomial regression to do model displacements [15]. Our work take this approach further.

III. Data Collection with the Learning Cube

We chose a data driven approach to be able to better adapt to the real world. For this to be feasible, we need to collect large amounts of high quality data in a reasonable amount of time and have a way to interact with the soft robots after training. We created a platform for data acquisition and experiments, named the Learning Cube. Our Cube combines depth sensors, IoT motors and soft robots. The Cube has a MDF plate box with an aluminium frame. The box provides a rigid base to mount motors and soft robots and a smooth, hard surface. Uniform light conditions are provided by LED-strips. To make segmentation of RGB-D data easy, the base of the box is painted in a diffuse black color. To maximize data quality the cameras can easily be rigidly mounted almost everywhere on the aluminium frame without obstruction. For creating a data set for training and validation we actuate a cable driven robot by sampling a control parameter vector and then we capture the corresponding shape deformation of the robot. Thereby creating a large dataset of matching shapes and control parameters. We apply a grid search like method to sample the whole space of parameters to shape changes. This is affordable given we have a low number of cables, often less than four. For higher number of actuators, one may apply importance sampling techniques.

To control the soft robots accurately, we favour cables in this work, due to their simplicity, but other actuators could also be used with our approach. We use 2-phase mercury stepper motors with 1.8 degree steps. The radius of the motor shaft is 2 mm. One step will tighten/loosen the cable by only 0.06 mm. Each motor is connected to a stepper motor driver that is used to control speed, acceleration and absolute position of the shaft. The motor drivers are daisy chained to an Arduino, which makes creating a setup with multiple motors modular and easy. More detailed descriptions and build instructions are provided open access [16].

A configuration of a soft robot is given by how much the cables have been pulled. This is equivalent to measuring the amount of shaft rotation for each cable. The shaft position of the i^{th} cable is stored in the parameter $\alpha_i \in \mathbb{R}$, and if we have P cables, then we store the current configuration in the parameter vector

$$\alpha \equiv \begin{bmatrix} \alpha_0 & \cdots & \alpha_{P-1} \end{bmatrix}^T . \tag{1}$$

We use Intel's RealSense D415 sensors to capture depth and color data. During the training phase, the robots are equipped with visual markers. For each configuration, α, we extract a shape vector by segmenting visual markers from the colour images and extracting the corresponding x, y, z coordinates from the point cloud. We merge point clouds from multiple sensors into one common coordinate system to reconstruct the surface of the soft robot. Our initial testing of sensors showed that noise and degradation from sensors are negligible. Hence, two corresponding calibration points, as viewed from two sensors A and B, are related by a rigid transformation. The translation and rotation can be found by least squares fitting of calibration points from each coordinate system, given we know the correspondence between calibration points [17]. As calibration points, we use detected centroids of fuss-balls as shown in Figure 3. To

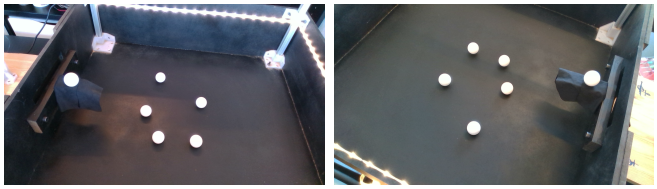


Fig. 3: Calibration balls viewed from two sensors. The ball shapes are easily identified and used to robustly detect calibration points given by the ball centroids. This gives an agile and robust calibration of multiple cameras.

determine the correspondence we search for the permutation, π, of detected calibration points in coordinate system B, as:

$$\pi^* \equiv \arg\min_{\pi} \frac{1}{C} \sum_{n=1}^{N} \left\| \mathbf{p}_n^A - \left(\mathbf{R}(\pi)\, \mathbf{p}_{\pi(n)}^B + \mathbf{t}(\pi) \right) \right\|^2 \quad (2)$$

where $(\mathbf{R}(\pi), \mathbf{t}(\pi))$ is the rigid transformation between calibration points $\mathbf{p}_n^A$ from sensor A and $\mathbf{p}_{\pi(n)}^B$ from sensor B and C is the number of calibration points.

After extracting a shape vector from each sensor, we merge them such that visual markers that are represented in both shape vectors are averaged out. Markers that are common to both sensors are detected by a lower threshold on pairwise marker-distance as shown in Figure 4. For each sampled

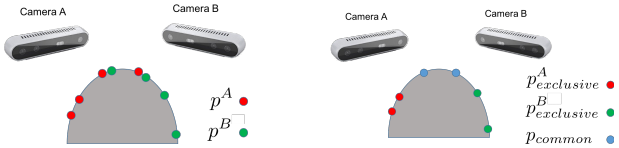


Fig. 4: The final shape vector of a robot configuration is given by the common set of visual markers from all sensors. We use averaging to combine common positions into the shape vector.

configuration, k, a set of marker points,

$$\mathcal{P}^k \equiv \left\{ \mathbf{p}_i^k = (x_i, y_i, z_i)^k \,|\, i \right\} , \quad (3)$$

is obtained by the above process. The points in $\mathcal{P}^k$ are not sorted, which means that there is no guarantee that points $\mathbf{p}_i^k$ and $\mathbf{p}_i^{k+1}$ refer to the same physical marker. Moreover, the size of the points sets may differ from configuration to configuration due to erroneous segmentation or noise in the depth data. This is solved by sorting all point sets from configurations $k = (1, 2, \ldots, K-1)$ with respect to the order of the first point set $\mathcal{P}^0$. This is done by greedily linking markers with the smallest displacement. Missing values are approximated as a Gaussian weighted mean of nearby marker displacements and false positives are detected as frequently approximated markers. Finally, if we have N markers, then the final shape vector is given by

$$\mathbf{s}^k \equiv \left[(\mathbf{p}_0^k)^T \quad \cdots \quad (\mathbf{p}_{N-1}^k)^T \right]^T , \quad (4)$$

$$= \begin{bmatrix} x_0^k & y_0^k & z_0^k & \cdots & x_{N-1}^k & y_{N-1}^k & z_{N-1}^k \end{bmatrix}^T . \quad (5)$$

Our software is provided as open source [18].

IV. Data Driven Inverse Kinematics

During the data acquisition phase, we sample the parameter space and extract the corresponding shape vectors, $\mathbf{s}$, from RGB-D data. The k^{th} configuration vector,

$$\alpha^k \equiv \begin{bmatrix} \alpha_0^k & \alpha_1^k & \ldots & \alpha_{P-1}^k \end{bmatrix}^T \quad (6)$$

contains the parameters for each actuator.

In this work we hypothesize that creating subdomains of the configuration space and using a local low-ordered model for each subdomain is more efficient than using a higher order global model for the whole configuration space. Below, we develop our IK model for a complete domain and in Section V we describe how subdomains are created.

Polynomial regression is used to learn a shape function $\mathbf{s}(\alpha) : \mathbb{R}^P \mapsto \mathbb{R}^N$. This is preferred over the Taylor approximation, due to the systematic bias resulting from fixing the intersection point. All the monomials of the sampled configurations, up to the degree of the approximation and the corresponding observed shape vectors are set up in a system of linear equations.

$$\mathbf{S} = \mathbf{W}\,\mathbf{B} \quad (7)$$

where

$$\mathbf{S} \equiv \begin{bmatrix} \mathbf{s}^0 & \mathbf{s}^1 & \cdots & \mathbf{s}^{K-1} \end{bmatrix} , \quad (8a)$$

$$\mathbf{B} \equiv \begin{bmatrix} \mathbf{b}(\alpha^0) & \mathbf{b}(\alpha^1) & \cdots & \mathbf{b}(\alpha^{K-1}) \end{bmatrix} . \quad (8b)$$

Here $\mathbf{b}$ is a mapping from a control vector to a vector containing all its monomials, up to degree equal to the order of the approximation. An efficient way of obtaining this mapping, such that $\mathbf{b}$ is without redundancies and with the correct coefficients is shown in Section VI. The shape matrix, $\mathbf{S}$, is linear in the weight coefficients, $\mathbf{W}$. Finding the optimal weights is thus a quadratic programming (QP) problem and is solved by taking the pseudo inverse of $\mathbf{B}$:

$$\mathbf{W} = \mathbf{S}\,\mathbf{B}^\dagger . \quad (9)$$

The shape function approximation we learn from the data is then given by

$$\mathbf{s}(\alpha) \approx \mathbf{W}\,\mathbf{b}(\alpha) . \quad (10)$$

We may now state the inverse kinematics problem for a soft robot. We want to find the configuration parameter, α^* , that minimizes the Euclidean distance between a desired goal shape, $\mathbf{s}^*$, and the predicted shape, $\mathbf{s}(\alpha^*)$. For this purpose, it is more convenient to deal with displacements, rather than shapes. Let β be the intersection of the manifold created by the polynomial regression, such that $\mathbf{W} = \begin{bmatrix} \beta & \hat{\mathbf{W}} \end{bmatrix}$. Thus, β contains the zeroth ordered weights and $\hat{\mathbf{W}}$ contains all the higher ordered weights. Note also that the first element of $\mathbf{b}(\alpha)$ is 1, $\mathbf{b}(\alpha) = \begin{bmatrix} 1 & \hat{\mathbf{b}}(\alpha)^T \end{bmatrix}^T$.

$$\mathbf{s}(\alpha) = \mathbf{W}\,\mathbf{b}(\alpha)\,, \tag{11a}$$

$$= \begin{bmatrix}\beta & \hat{\mathbf{W}}\end{bmatrix}\begin{bmatrix}\mathbf{1}\\ \hat{\mathbf{b}}(\alpha)\end{bmatrix}, \tag{11b}$$

$$= \beta + \hat{\mathbf{W}}\,\hat{\mathbf{b}}(\alpha)\,. \tag{11c}$$

If we denote shape displacements as $\mathbf{u} = \mathbf{s} - \beta$, then the desired $(*)$ control parameters can be found by:

$$\alpha^* \equiv \arg\min_{\alpha} \frac{1}{2}\left\|\mathbf{s}^* - \mathbf{s}(\alpha)\right\|^2, \tag{12a}$$

$$= \frac{1}{2}\left\|\mathbf{s}^* - \left(\beta + \hat{W}\hat{\mathbf{b}}(\alpha)\right)\right\|^2, \tag{12b}$$

$$= \frac{1}{2}\left\|\mathbf{u}^* - \hat{W}\hat{\mathbf{b}}(\alpha)\right\|^2. \tag{12c}$$

For first order approximations the shape function is linear in the configuration parameters, and (12) is a QP problem that can be solved efficiently using the Moore-Penrose inverse.

$$\alpha^*(\mathbf{s}^*) = \mathbf{W}^\dagger\,\mathbf{u}^*\,, \tag{13a}$$

$$= \mathbf{W}^\dagger\,\left(\mathbf{s}^* - \beta\right)\,. \tag{13b}$$

For higher order approximations, optimization methods are utilized to minimize (12). In this work, we used the built in fmincon solver in MATLAB with the Sequential Quadratic Programming (SQP) algorithm [19] for higher order approximations. We wrote our own MATLAB code for (13b).

V. Domain Decomposition of Configuration Space

As the amount of variability in the observed shape deformations increase, we need more complex shape approximations to capture it. The size of the weight matrix in polynomial regression is determined by the number of visual markers, N, configuration parameters, P, and the order of the approximation, d,

$$|\mathbf{W}| \equiv N\left(\sum_{i=1}^{d}\binom{P+i-1}{i} + 1\right). \tag{14}$$

The polynomial regression quickly becomes computationally infeasible as P and d grows, due to the size of the weight matrix. Moreover, $\mathbf{B}$ in (9) must have full column rank to be invertible. The minimum number of observations in the training data, K_{min}, must therefore exceed the number of unknown parameters in the system of linear equations:

$$K_{min} \geq \sum_{i=1}^{d}\left(\binom{P+i-1}{i} + 1\right). \tag{15}$$

This is an obstacle, since collecting data is time consuming. As an example, to make an 8th ordered approximation with a 12 DOF actuated robot, we need 125000 samples. Collecting one sample takes about three seconds, so acquiring all the data takes about four days. To overcome this problem, we partition the configuration space into disjoint regions using k-means clustering. k-means clustering provides fully connected regions of roughly the same extent. The partition

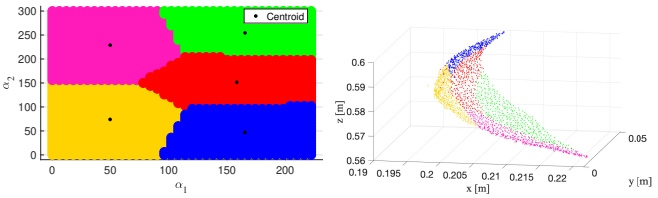


Fig. 5: Decomposed configuration space (left) and workspace (right) for a 2 DOF actuated soft robot with one marker. Observe the close to spatial linear shape of subdomains in workspace (right) promising good fit with lower order models.

of a 2D configuration space and the corresponding partition of visual marker observations in the work space is depicted in Figure 5. A local model is created for each region. Since the amount of variability is much smaller locally, a lower model order is sufficient. Moreover, the condition for full rank of $\mathbf{B}$ is then:

$$K_{min} \geq |\mathcal{L}|\sum_{i=1}^{d}\binom{P+i-1}{i}, \tag{16}$$

which grows linearly in the number of local models, $|\mathcal{L}|$. The assumption behind this bound is that the observations are evenly distributed between the regions, which is not guaranteed with k-means clustering. It can be obtained by letting smaller regions borrow observations from larger regions. The approximation order can then be reduced by increasing the number of local models.

We need a method to chose the correct local model label l to use for the shape function. We favour simplicity in our chosen methods and have decided to use two different methods, depending on the order of the local models. When the order is small, we can evaluate each local model quickly. The local model that can best describe the given shape is then chosen. That is,

$$l^* \equiv \arg\min_{l\in\mathcal{L}} \left\|\mathbf{s}_l(\alpha^*) - \mathbf{s}^*\right\|^2 \tag{17}$$

where α^* is solved using (13b) and $\mathbf{s}_l$ denotes the l^{th} local model. When the model order is high, evaluating every model is inconvenient since each model requires more computational resources. Instead, a pre-trained regression tree classifier is used to predict the optimal local model label. This is faster, since only one local model is evaluated, but less accurate due to possible misclassifications of the regression tree.

VI. Efficient Polynomial Regression

A monomial of degree d is a product where the sum of the exponents is equal to d. For a configuration vector α, all the possible monomials are the solution space of:

$$\prod_{p=0}^{P-1}(\alpha_p)^{\mathbf{L}_p} \quad \text{s.t.:} \quad \sum_{p=0}^{P-1}\mathbf{L}_p = d, \quad \mathbf{L}\in Z^P \tag{18}$$

known as the exponential Diophantine equation. $\mathbf{L}$ is any P-dimensional vector of positive integers whose sum is equal

to d. A set containing all the monomials of degree d of a vector $\alpha \in \mathbb{R}^P$, can be expressed using set builder notation

$$\left\{ \prod_{p=0}^{P-1} (\alpha_p)^{\mathbf{L}_p} \quad \forall \mathbf{L} \;\middle|\; \mathbf{L} \in Z^P, \quad \sum_{p=0}^{P-1} \mathbf{L}_p = d \right\} \tag{19}$$

This gives all the possible monomials of degree d, without redundancy, but not with the right coefficients. Just like the binomial distribution can be used to find the coefficients of $(x_0 + x_1)^d$ when expanded, the coefficients of $(\sum_{p=0}^{P-1} \alpha_p)^d$ can be found with the multinomial distribution. The coefficient of the term $\prod_{p=0}^{P-1} \alpha_p^{\mathbf{L}_p}$ is then $\frac{d!}{\prod_{p=1}^{P} \mathbf{L}_p!}$. For instance, upon expansion of $(\alpha_0+\alpha_1+\alpha_2)^4$, the term $(\alpha_1^2\alpha_2\alpha_3)$ will have the coefficient $\frac{4!}{2!1!1!}$. Moreover, d^{th} ordered terms in the polynomial regression are divided by $d!$. We get,

$$M_\alpha^d = \left\{ \prod_{p=0}^{P-1} \frac{(\alpha_p)^{\mathbf{L}_p}}{\mathbf{L}_p!} \quad \forall \mathbf{L} \;\middle|\; \mathbf{L} \in Z^P, \quad \sum_{p=0}^{P-1} \mathbf{L}_p = d \right\}$$

and

$$\mathbf{b}(\alpha) = \begin{bmatrix} 1 & M_\alpha^1 & M_\alpha^2 & \dots & M_\alpha^d \end{bmatrix} \tag{20}$$

The problem of computing $\mathbf{b}(\alpha)$ is then reduced to finding the set, $\mathbf{L}$, containing all vectors of length P that sums to d. This can be implemented efficiently using recursion.

When solving (12) numerically, $\mathbf{b}$ has to be recomputed in every step. This is an operation that becomes slower the higher the model order is. However, the order should be kept relatively small, since overfitting is more likely to occur for complex models. This is illustrated in Figure 6.

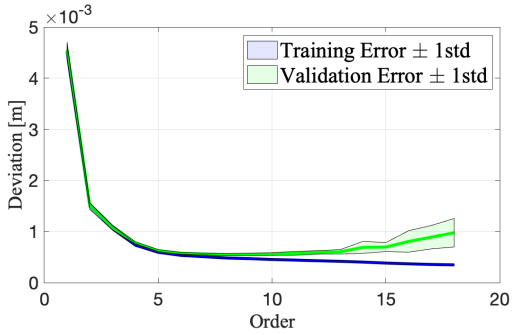
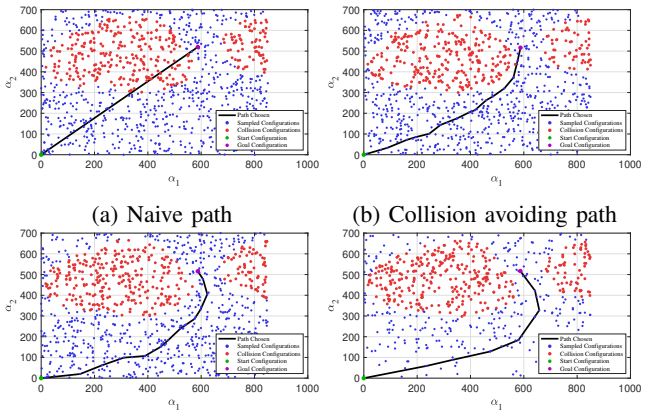


Fig. 6: Euclidean distance (m) between collected shape data and predicted shapes, using polynomial regression of increasing order. Observe that complex models tent to overfit to the data, as can be seen by the increasing validation error.

VII. Experiments

We found that we can achieve a fairly low validation error with the IK-models we have created. This is typically around 8 motor steps, which amounts to about 0.5mm of actuator cable. Nevertheless, there may be discrepancies between the modelled behaviour of the robot and the real world. To validate the model in a real world context, we created a system that detects the centroid of a small ball that can be used as input to the IK-model. We use a single marker for training of the robot and treat the centroid of the ball as $\mathbf{s}^*$. Using a cable driven silicone finger, capable of bending and translation, we deployed this scheme to minimize the distance between the tip of the finger and the centroid of a ball. Figure 2 depicts how it continuously tries to tip the ball over.

We created a cable driven grabber of silicone. Since the model only considers the start and the end shape of the robot, we found that the grabber tended to collide with the object of interest. This is shown in Figure 7a, where the path through the configuration space goes though the collision space. To overcome this problem, we implemented a variant of the probabilistic road map (PRM) algorithm to find a collision free path in the configuration space [20]. Random samples from the configuration space are treated as nodes in a graph where colliding configurations are removed. As observed in Figure 7b, the resulting path is indeed the shortest, but is dangerously close to collision. By associating weights to the edges that are inversely proportional to the distance to the closest colliding configuration, we obtain a safer path as shown in Figure 7c. If we sample less frequently the further away from the collision space, we get a path consisting of fewer samples and with low collision probability as illustrated in Figure 7d. The learning ability



(a) Naive path (b) Collision avoiding path

(c) Min. probability of collision (d) Importance sampled

Fig. 7: Planned paths through the configuration space. Using a road probability map with associated edge weights generates safer paths.

of our model depends on the number of local models and the order of the approximating polynomials. Regarding the order, there is a trade off between training error vs time and memory complexity, validation error and required data quantity. The same goes for the number of local models, but as we saw in (14), the number of unknowns in the system of linear equations (10), increase linearly with the number of local models, opposed to the power of the order.

We collected 2244 shape and configuration vectors from a soft robot with two actuators. We performed 5-fold cross validation on the two hyper parameters, number of models and order, to find an optimal combination. One that gives us a model that generalizes well on unseen data and does not require too much computational time. The experiment was

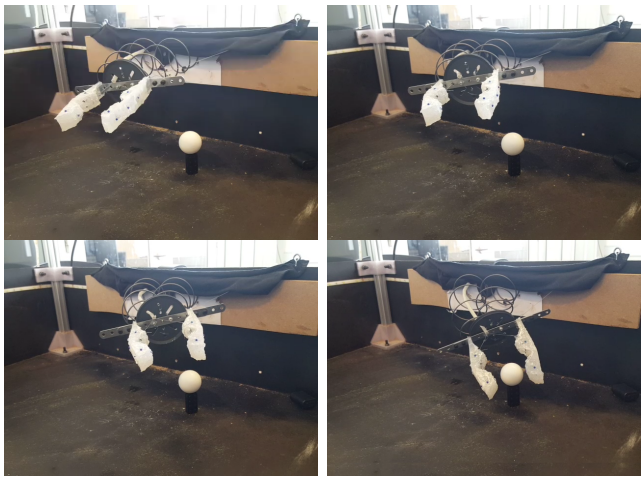


Fig. 8: Grabber choosing collision free trajectory using our RPM approach. This demonstrates that the fast IK approach of local models is feasible in a more complex motion planning scenario.

performed twice: once with each of the two model selection methods described in Section V.

Figure 9a and 9d illustrates that the number of local models as well as the model order are means of increasing the learning ability of the model. The exception is when the number of local models is so large that the regression tree struggles with classifying the correct local model labels.

From Figure 9b and 9e, it is clear that the minimum solution provides an overall lower validation error than when using regression trees. On the other hand, the execution time is much smaller when using regression trees, except when the models are linear, which can be seen in Figure 9c and 9f.

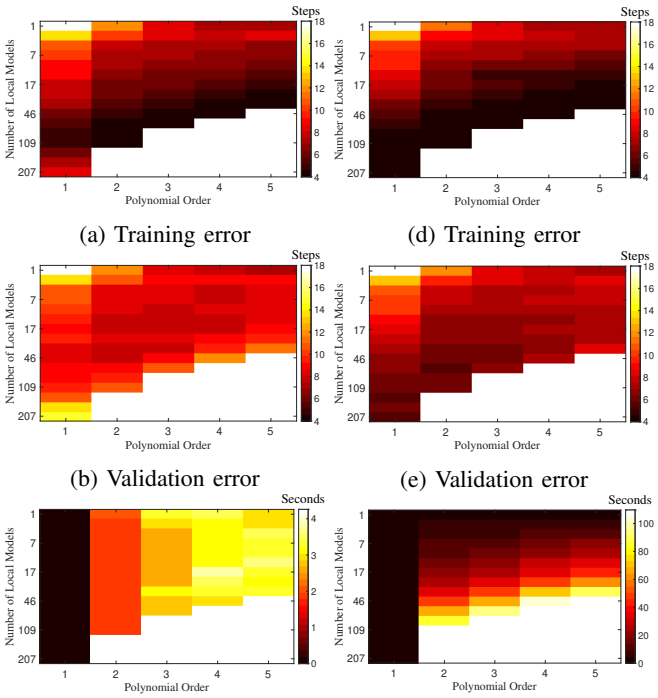
As shown in Figure 9b, we can reach the same low validation error with 50 linear local models as with 15 3rd order models. Using the linear models, is, however, about 30 times faster.

VIII. Discussion and Conclusion

A computationally efficient data driven approach for learning the non-linear relationship between configuration parameters and the shape of soft robots has been proposed. The inverse problem of finding optimal configuration parameters with respect to a desired goal shape is computed using numerical optimization. We have demonstrated that using multiple, low order local models reduces the time complexity, required data quantity as well as increases the accuracy of both the forward and inverse modelling. This is important for the scalability of the model with respect to robot complexity.

Our work is currently limited to a low number of control parameters, since the time needed for sampling the data set increase by an order of magnitude for each added control value. Further research may include importance sampling to avoid this time disadvantage.

Polynomial regression may become computationally infeasible for an extremely large number of control parameters.



(a) Training error (d) Training error

(b) Validation error (e) Validation error

(c) Execution time (f) Execution time

Fig. 9: Cross validation using minimum solution for model selection (left column) and using regression trees for model selection (right column). Observe that many local low order models are advantageous.

Future work may address sparse methods for estimating Jacobians and Hessians of the shape function approximation. This work is promising in the respect that it demonstrates that higher order models are likely not very computationally efficient.

The proposed method can in principle be extended to account for loads, although this has not been tested. By appending the relevant load to the shape vector in the data acquisition phase, we could obtain a model: $\alpha^*([load, \mathbf{s}^*])$.

Lastly, our work focuses on kinematics and uses visual markers. The point cloud data holds promising directions for learning dynamics and fitting a computational mesh to the point cloud data to increase the dimensionality and shape representation of the shape vectors further.

REFERENCES

[1] S. R. Toolkit, "Sdm fingers," https://softroboticstoolkit.com/book/sdm-fingers, 2018.

[2] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[3] F. Gao, Z. Wang, Y. Wang, Y. Wang, and J. Li, "A prototype of a biomimetic mantle jet propeller inspired by cuttlefish actuated by sma wires and a theoretical model for its jet thrust," *Journal of Bionic Engineering*, vol. 11, no. 3, pp. 412–422, 2014.

[4] Z. Zhang, T. M. Bieze, J. Dequidt, A. Kruszewski, and C. Duriez, "Visual servoing control of soft robots based on finite element model," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2895–2901.

[5] A. Rodríguez, E. Coevoet, and C. Duriez, "Real-time simulation of hydraulic components for interactive control of soft robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4953–4958.

[6] G. Runge, M. Wiese, L. Günther, and A. Raatz, "A framework for the kinematic modeling of soft material robots combining finite element analysis and piecewise constant curvature kinematics," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, April 2017, pp. 7–14.

[7] H. Jin, E. Dong, S. Mao, M. Xu, and J. Yang, "Locomotion modeling of an actinomorphic soft robot actuated by sma springs," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, Dec 2014, pp. 21–26.

[8] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, pp. 467 EP –, 05 2015.

[9] G. Gerboni, A. Diodato, G. Ciuti, M. Cianchetti, and A. Menciassi, "Feedback control of soft robot actuators via commercial flex bend sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 4, pp. 1881–1888, Aug 2017.

[10] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2550–2555.

[11] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, July 2017.

[12] J. Bosman, T. M. Bieze, O. Lakhal, M. Sanz, R. Merzouki, and C. Duriez, "Domain decomposition approach for fem quasistatic modeling and control of continuum robots with rigid vertebras," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4373–4378.

[13] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Learning the inverse kinetics of an octopus-like manipulator in three-dimensional space," *Bioinspiration & Biomimetics*, vol. 10, no. 3, 2015.

[14] J. Chen and H. Y. K. Lau, "Learning the inverse kinematics of tendon-driven soft manipulators with k-nearest neighbors regression and gaussian mixture regression," in *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, April 2016, pp. 103–107.

[15] A. J. Taylor, R. Montayre, Z. Zhao, K. W. Kwok, and Z. T. H. Tse, "Modular force approximating soft robotic pneumatic actuator," *International Journal of Computer Assisted Radiology and Surgery*, Aug 2018.

[16] F. Holsten, "Soft robotics ai," Master's thesis, University of Copenhagen, 2018.

[17] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, Sept 1987.

[18] F. Holsten and K. Erleben, "pysoro," https://github.com/erleben/pySoRo, 2018.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[20] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 8 1996.