Data Driven Inverse Kinematics of Soft Robots using Local Models

Fredrik Holsten², Morten Pol Engell-Nørregård², Sune Darkner¹, and Kenny Erleben¹

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.

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
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 takes 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

$$\alpha \equiv [\alpha_0 \cdots \alpha_{P-1}]^T. \quad (1)$$

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, $\alpha$, 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
determine the correspondence we search for the permutation, \( \pi \), of detected calibration points in coordinate system \( B \), as:

\[
\pi^* = \arg \min_{\pi} \frac{1}{C} \sum_{n=1}^{N} \left\| \mathbf{p}_n^A - \left( \mathbf{R}(\pi) \mathbf{p}_n^B + \mathbf{t}(\pi) \right) \right\|^2 \tag{2}
\]

where \( (\mathbf{R}(\pi), \mathbf{t}(\pi)) \) is the rigid transformation between calibration points \( \mathbf{p}_n^A \) from sensor \( A \) and \( \mathbf{p}_n^B \) 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 configuration, \( k \), a set of marker points,

\[ \mathcal{P}^k \equiv \{ \mathbf{p}_i^k = (x_i, y_i, z_i) | i \} \tag{3} \]

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 \( \mathbf{p}_{i+1}^k \) 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

\[
\mathbf{s}^k = \left[ \begin{array}{c} x_0^k \ y_0^k \ z_0^k \ x_{N-1}^k \ y_{N-1}^k \ z_{N-1}^k \end{array} \right]^T \tag{4}
\]

We may now state the inverse kinematics problem for a soft robot. We want to find the configuration parameter, \( \alpha^* \), 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 \( \beta \) be the intersection of the manifold created by the polynomial regression, such that \( \mathbf{W} = [\beta \ \mathbf{W}] \). Thus, \( \beta \) contains the zeroth ordered weights and \( \mathbf{W} \) contains all the higher ordered weights. Note also that the first element of \( \mathbf{b}(\alpha) \) is 1, \( \mathbf{b}(\alpha) = [1 \ \mathbf{b}(\alpha)^T]^T \).
\begin{align}
\mathbf{s}(\alpha) &= \mathbf{W} \mathbf{b}(\alpha), \\
&= \left[ \begin{array}{c}
\beta \\
\hat{\mathbf{W}}
\end{array} \right] \left[ \begin{array}{c}
1 \\
\mathbf{b}(\alpha)
\end{array} \right], \\
&= \beta + \hat{\mathbf{W}} \mathbf{b}(\alpha). \tag{11c}
\end{align}

If we denote shape displacements as \( \mathbf{u} = \mathbf{s} - \beta \), then the desired \((\ast)\) control parameters can be found by:
\begin{align}
\alpha^\ast &= \arg \min_{\alpha} \frac{1}{2} \| \mathbf{s}^\ast - \mathbf{s}(\alpha) \|^2, \\
&= \frac{1}{2} \| \mathbf{s}^\ast - (\beta + \hat{\mathbf{W}} \mathbf{b}(\alpha)) \|^2, \tag{12b}
\end{align}
\begin{align}
\| \mathbf{u}^\ast - \hat{\mathbf{W}} \mathbf{b}(\alpha) \|^2. \tag{12c}
\end{align}

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.
\begin{align}
\alpha^\ast(\mathbf{s}^\ast) &= \mathbf{W}^\dagger \mathbf{u}^\ast, \\
&= \mathbf{W}^\dagger (\mathbf{s}^\ast - \beta). \tag{13b}
\end{align}

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, \(d\),
\begin{align}
|\mathbf{W}| \equiv N \left( \sum_{i=1}^{d} \binom{P + i - 1}{i} + 1 \right). \tag{14}
\end{align}
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:
\begin{align}
K_{\min} \geq \sum_{i=1}^{d} \binom{P + i - 1}{i} + 1. \tag{15}
\end{align}

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 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:
\begin{align}
K_{\min} \geq |\mathcal{L}| \sum_{i=1}^{d} \binom{P + i - 1}{i}, \tag{16}
\end{align}
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,
\begin{align}
l^\ast &= \arg \min_{l \in \mathcal{L}} \| s_l(\alpha^\ast) - \mathbf{s}^\ast \|^2 \tag{17}
\end{align}
where \(\alpha^\ast\) is solved using (13b) and \(s_l\) 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 \(\alpha\), all the possible monomials are the solution space of:
\begin{align}
\prod_{p=0}^{P-1} (\alpha_p)^{L_p} \text{ s.t.: } \sum_{p=0}^{P-1} L_p = d, \quad \mathbf{L} \in \mathbb{Z}^P \tag{18}
\end{align}
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
\[
\left\{ \frac{P-1}{\prod_{p=0}^{P-1} \alpha_p L_p} \forall L \in Z^P, \sum_{p=0}^{P-1} L_p = d \right\} \tag{19}
\]
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 \( \sum_{p=0}^{P-1} \alpha_p L_p \)
can be found with the multinomial distribution. The coefficient of
the term \( \prod_{p=0}^{P-1} \frac{\alpha_p L_p}{L_p} \) is then \( \frac{d!}{P!} \). For instance,
on the expansion of \((\alpha_0 + \alpha_1 + \alpha_2)^d\), the term \( \alpha_1^2 \alpha_2 \alpha_3 \) will
have the coefficient \( \frac{d!}{2!1!1!} \). Moreover, \( d \)th ordered terms in the
polynomial regression are divided by \( d! \). We get,
\[
M'_\alpha = \left\{ \frac{P-1}{\prod_{p=0}^{P-1} \alpha_p L_p} \forall L \in Z^P, \sum_{p=0}^{P-1} L_p = d \right\}
\]
and
\[
b(\alpha) = \begin{bmatrix} M_1^{d} & M_2^d & \ldots & M_d^d \end{bmatrix} \tag{20}
\]
The problem of computing \( b(\alpha) \) is then reduced to finding
the set, \( L \), containing all vectors of length \( P \) that sums to \( d \).
This can be implemented efficiently using recursion.

When solving (12) numerically, \( b \) 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.](image)

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
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
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
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.

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: \( \alpha^*([\text{load}, s^*]) \).

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


