Inverse Kinematics (IK) solves the problem of mapping from the Cartesian space to the joint configuration space of a robotic arm. It has a wide range of applications in areas such as computer graphics, protein structure prediction, and robotics. The problem is commonly solved analytically based on the structure of the robotic arm or numerically by approximation through recursive methods, e.g., Jacobian-based methods. Over the past decade, data-driven methods have also been exploited. Unfortunately, these approaches to IK become inadequate for high Degree-of-Freedom (DoF) robotic arms. Theoretically, the redundant DoFs of such robotic arms can provide an infinite number of solutions to IK for reaching a given target position. The huge solution space could be exploited for more flexible operations of high-DoF robotic arms. The problem is that existing approaches are confined and normally produce only one joint solution for a target position. This paper presents the first work that solves high-DoF IK by generating multiple distinct joint solutions to reach any given target position in the working space. The proposed data-driven approach can be applied to any robotic arms without knowing detailed kinematics information. It not only obtains multiple distinct joint solutions for a target position, but also solves the high-DoF IK problem within a millisecond and achieves subcentimeter distance errors with very sparse training data.

1 Introduction

A key computation in robot control is solving the inverse kinematics (IK) problem. The problem is to find the joint parameters that can move the end effector of a robotic arm to a target position, perhaps with a certain orientation. The IK problem has been studied extensively since the 1980s [1, 3, 5, 22, 23, 27, 29, 30], and has found applications in such diverse areas as computer graphics, protein structure prediction, robotics, etc. The problem is commonly solved analytically based on the structure of the robotic arm [1, 3, 23, 27, 30] or numerically by approximation through iterative calculations [5, 22, 29], e.g., Jacobian-based methods. Recently, we have also seen data-driven approaches to solving the IK problem [2, 7, 8, 10, 12, 15, 21, 24, 26, 28]. Unfortunately, existing works become insufficient for high Degree-of-Freedom (DoF) robotic arms, which have attracted growing interests recently [13, 14, 18, 19, 31].

One distinct feature of high-DoF robotic arms, i.e., arms with more than six DoFs, is the redundant DoFs. It opens up new possibilities for robot controls. For example, high-DoF robotic arms can approach objects from different angles with different postures, perform tasks with the most efficient posture, maneuver to avoid obstacles, or mimic human postures more closely. Theoretically, the redundant DoFs can provide an infinite number of joint solutions in IK to reach a given target position, though many solutions give similar postures. Hence, to better exploit the redundant DoFs, it is essential to obtain multiple “distinct” joint solutions for a target position. As far as we know, the problem has not been systematically studied before, and existing works cannot be extended easily to solve this problem. Without loss of generality, we assume that joints of the robotic arms consist of rotational joints only. Thus, we will use “angle solutions” instead of “joint solutions” in the following discussions.
The analytical methods can solve the IK problem fast and comprehensively by deriving a closed-form expression for a robotic arm. However, the structure of the arm has to obey certain kinematics rules and cannot be too complicated. Unfortunately, high-DoF robotic arms are too complex to derive closed-form expressions for IK. The numerical methods approximate the target position iteratively from a starting point. The iterative approximation may take a longer time to compute the IK, and it produces only one angle solution. On the other hand, for high-DoF robotic arms, different starting points may generate different angle solutions to the given target position. Therefore, to obtain multiple angle solutions, we can simply perform the iterative approximation with different starting points. The problem, however, is that there is no guarantee that the solutions are sufficiently distinct and the total computation time is prolonged.

Most data-driven methods solve the IK problem using Deep Neural Networks (DNNs) to learn the mapping function from target positions to angle solutions \[2, 7, 8, 10, 12, 15, 26, 28\]. To compare with the numerical methods, the data-driven methods have shorter computation time and can directly compute the joint angles to reach the target position. In other words, the initial position of the robot is immaterial and it can reach the target position by simply turning the joints to the designated angles. However, the mapping functions can only associate one target position to one and only one angle solution. To obtain multiple angle solutions, a simple extension is to build several DNNs, each mapping the target positions to different angle solutions, or alternatively to train a single DNN that maps target positions plus an integer group id to angle solutions. In either method, the problem is that, since every target position has countless angle solutions with varying characteristics to reach it, clustering them into a fixed number of groups may not best match the characteristics of the angle solutions, leading to larger errors in reaching the target position.

This paper presents the first comprehensive study to tackle the problem of solving IK for high-DoF robotic arms that finds multiple distinct angle solutions for given target positions fast. We adopt the data-driven approach for its fast computation time and use self-supervised learning for generating the rational indices by considering the characteristics of the angle solutions. The key insight is that this problem is essentially a clustering of the infinite number of angle solutions to a target position into groups and a way of characterizing the groups. We thus propose to learn a feature vector for the groups that can best characterize them to obtain the optimal clustering. The feature vector is referred to as the posture index, which can be learned from training data through techniques such as Auto-Encoder and Variational Auto-Encoder (VAE). Since a target position may have many posture indices, we propose to use the KD-tree and the dictionary data structure to track the known positions, i.e., those in the training data and their corresponding posture indices. It follows that, given a target position, we can find different angle solutions by selecting one of the posture indices and then feeding it to the joint angle calculation model to get the angle solution. The proposed method is thus called Selective Inverse Kinematics (SIK).

SIK can be applied to find multiple distinct angle solutions for a target position or a position plus an orientation. Hence, in the following discussions, the term "goal" will be used to refer to both, while "target position" refers to position only.

The main contributions of the paper are as follows:

- To the best of our knowledge, this is the first work that solves the IK problem for high-DoF robotic arms by providing multiple distinct angle solutions to reach a given goal with sub-centimeter errors. We only need to use a single DNN plus a KD-tree.
- The proposed SIK requires very sparse training data to achieve very high accuracy in reaching the target position. A technique called Regional Goal Alignment is proposed to solve the IK for goals not in the training data. Experimental results show that the accuracy can be within $\sim0.5$ cm with training data collected by every $30^\circ$ per joint.
- It has a fast computation time for real-time manipulations.
- It can be applied to any robotic arm without hardware constraints.
- The paper provides a very comprehensive study of the proposed SIK, including properties of posture indices, effects of the sparsity of the training data, different training strategies, positions as well as orientations as the goal, etc.

The remainder of the paper is organized as follows. Section 2 introduces the proposed SIK, and Section 3 shows our experiments and results. To better understand the contribution of the proposed approach, related works are described in Section 4. Conclusions are drawn in Section 5.

### 2 Approach

This section presents the proposed SIK method. Section 2.1 gives an overview of the method. In Section 2.2, we present the offline processing part of the method, including ground-truth dataset collection, joint angle calculation model...
Selective Inverse Kinematics: A Novel Approach to Finding Multiple Solutions Fast for High-DoF Robotic Arms

A P REPRINT

Figure 1: Overview of the proposed SIK approach.

training, and data dictionary preparation. Section 2.3 introduces the online processing part of the method, including posture index selection and Regional Goal Alignment (RGA), which extends posture indices for unseen target positions.

2.1 Overview

We denote the desired goal, which is a target position plus perhaps an orientation, of the robotic arm by \( g \). The set of angle solutions with respect to desired goal \( g \) is denoted by \( \mathbb{A}_g = \{ a^0_g, a^1_g, \ldots, a^n_g \} \), where \( a^i_g \) is a vector representing an angle solution for the given goal. Given the desired goal \( g \), ideally we want to find a function \( P(\mathbb{A}_g|g) \) to generate all angle solutions. However, obtaining the joint probability of \( \{ a^0_g, a^1_g, \ldots, a^n_g \} \) is not realistic due to the infinite number of angle solutions of high-DoF robotic arms. Therefore, we simplify this problem by adopting an alternative function \( P(\mathbb{A}_g|g,i^g) \), where \( i^g \) is an index referring to one solution in the solution set \( \mathbb{A}_g \). With this function, it is possible to obtain different solutions in the set \( \mathbb{A}_g \) by providing different indices. Now, there is no need to calculate the solutions all at once, because the index can be used to generate individual solutions.

In order to obtain multiple "distinct" angle solutions to the given goal, the indices must be constrained. Otherwise, the indices may refer to very similar angle solutions. Therefore, it is desirable to have the indices to refer to the angle solutions that are as diverse as possible. In summary, the primary objective of SIK is to find a conditional probability function \( P(\mathbb{A}_g|g,i^g) \) that can return one of the most likely angle solutions for a given index \( i^g \), and the difference of any two indices must have a positive correlation with the difference of the corresponding angle solutions.

To find the conditional probability function \( P(\mathbb{A}_g|g,i^g) \), we propose to train a joint angle calculation DNN model to perform that function, similar to most existing data-driven methods to solving the IK problem. Unlike previous works that use DNNs to learn the association from a desired goal to a single angle solution, our joint angle calculation model for solving the high-DoF IK will take an extra index as the input to refer to one angle solution from among the multiple solutions to reach the desired goal. This DNN can be easily understood as a black box with a control rod outside and many functions inside. Given a desired goal as input, the black box can generate different outputs by altering the rod to enable the mapping function that we want to switch on.

For the joint angle calculation model to work, we need some auxiliary mechanisms to provide it with an index from among a number of indices that allow the robotic arm to reach the given desired goal. Since the working space of a robotic arm is fixed and bounded, we thus propose to provide the indices with a dictionary that contains the posture indices. Since it is impossible and unnecessary to build a dictionary that contains all the desired goals in the working space, we will only store a partial set of data and rely on kernel regression, a technique called Regional Goal Alignment, to generate indices for desired goals not in the dictionary. The sparse set of data can also be used as the training data to train the joint angle calculation model and thus will be called the ground-truth dataset in the later discussions.

Fig. 1 shows an overview of our approach, which consists of the two parts described above. The desired goal is first used to look up the dictionary, which in turn outputs an index. The index is then fed into the joint angle calculation model together with the given desired goal to obtain an angle solution. The whole process can be divided into two phases, each consisting of a number of steps:

**Offline phase:**

**Step 1.** Ground-truth dataset collection:
Uniformly sample the angles of the robotic arm to collect angle solutions \( a^0_g \) to their corresponding desired goal \( g \).
Step 2. Joint angle calculation model training:
Use the ground-truth dataset as the training data to train the joint angle calculation model.

Step 3. Dictionary building:
Store all the desired goals seen in Step 1 with their corresponding indices, which are obtained in Step 2 as byproducts, in the dictionary.

Online phase:

Step 1. Dictionary lookup:
Given a desired goal, look up the dictionary to find the closest data point and retrieve its indices.

Step 2. Regional goal alignment:
Align the indices to match the given desired goal.

Step 3. SIK inference:
Select one index to apply to the joint angle calculation model, together with the desired goal, to obtain one angle solution to move the robotic arm to the desired goal.

2.2 Offline Processing

In this subsection, we describe the operations performed in the offline phase, including ground-truth dataset collection, joint angle calculation model training, and dictionary building.

Ground-truth dataset collection

The proposed SIK method follows a supervised learning strategy to learn the indices to the multiple angle solutions for a given desired goal. Therefore, the first step is to collect the ground-truth dataset, which can be used for the offline SIK neural network training and for the online index selection. The ground-truth dataset consists of joint angles of the arm, the coordinates (x, y, z) of the end-effector, and the optional orientation of the end-effector. Since the working space of the robotic arm is known and fixed, the ground-truth dataset can be collected offline. Theoretically, we can collect every possible angle solution to every position in the working space of every possible orientation and store the data in a huge table. In practice, this is unnecessary and inefficient. As will be shown later, we only need to sample a very sparse set of data points, and the errors by our SIK method can be reduced to sub-centimeters.

In this work, we propose to turn each joint of the robotic arm in turn by \( \pi x \) degrees to collect the ground-truth dataset. For each turn of a joint, we record the angles of all the joints and use forward kinematics (FK) to calculate the coordinates and orientations of the end-effector. In the end, we will collect a large number of tuples that map from angle solutions to the positions and orientations of the end-effector. By sorting these tuples using positions and orientations as the key, we will have another set of tuples that map from positions and orientations to angle solutions. The latter tuples constitute the ground-truth dataset of our SIK method. Our experiments show that turning the joint angles by \( \pi x = 30 \) degrees to collect a very sparse set of ground-truth dataset is sufficient to maintain the IK errors within a centimeter.

Joint angle calculation model

With the ground-truth data, we can train a neural network to implement the function \( P(a^x_\pi g, i^x_\pi) \) that maps from a given desired goal and an index to an angle solution. To simplify the following discussions, we will focus on target positions as the desired goal. The orientations will be discussed later.

Intuitively, we can directly use a sequence of integers as the index to generate angle solutions. Unfortunately, this can at best produce suboptimal solutions. For one reason, since there is an infinite number of angle solutions to a target position and the solutions are continuous, there is no rational labeling method to cluster the angle solutions and associate the clusters to the integers, and in addition keep the clustering consistent across all target positions. Any mechanical clustering of the angle solutions to a fixed number of groups will inevitably lead to suboptimal solutions. We need a better index that can cluster all the angle solutions to a target position and best match the characteristics of the clusters.

Our idea in the SIK is to learn an efficient index, called posture index, by self-supervised learning using the ground-truth dataset collected in the previous step. The idea is inspired by Auto-encoder [4], which is an unsupervised feature-learning scheme. In an auto-encoder, we use an encoder to compress the input and a decoder to reconstruct the compressed data. By calculating the reconstruction loss, the posture indices are related to the angle solutions, and they have a strong, consistent meaning to the postures of the robotic arm. With these properties, the SIK can exploit the characteristics of posture indices to realize how the type of the angle solution should be like, and then the SIK can generate the proper answer with the given desired goal.
Selective Inverse Kinematics: A Novel Approach to Finding Multiple Solutions Fast for High-DoF Robotic Arms

Fig. 2 shows the implementation of our approach. The green part is our main angle calculation model for inference, and the rest of the architecture aims to obtain the posture indices. In the training phase, the encoder part first takes target positions, and their angle solutions from the ground-truth dataset and extracts feature vectors that can characterize their clusterings as the posture indices. The decoder part then receives a target position and its posture indices, deconverting this information back to the angle solutions by minimizing the reconstruction loss. After training, the decoder part in the figure can be used to calculate the angle solutions during inference.

Adopting reconstruction loss is sufficient for retrieving the characteristics of the postures. The fundamental problem is that the learned representations may not be continuous, so the interpolation of the posture indices may also not exist. This is not a serious problem to our approach though, because we only have to know different indices for different postures instead of every latent posture lying on the latent space. Nevertheless, the continuity of the posture indices may still be desirable. For example, the model can learn some synthesized postures by interpolating two different postures. Hence, we propose a variant of the SIK, called PSIK, which encodes the posture indices more rationally by mapping them to probability distributions. PSIK is more suitable for training with a very sparse ground-truth dataset, because sparse data is insufficient for the SIK to infer the continuous angle solutions between ground-truth dataset points. By introducing probabilistic distributions, the posture index becomes a range instead of a point. Thus, for any given target position not in the ground-truth dataset, its posture indices must have a strong correlation with the adjacent ground-truth dataset points.

To implement PSIK, we adopt Variational Auto-encoder (VAE) [16,17,25]. The VAE has the important advantage of approximating posterior with continuous latent variables. Fig. 3 shows the architecture of PSIK. We first convert the target position and the angle solutions to a simple probability distribution, e.g., normal distribution, through an encoder. Then we sample latent variables, i.e., posture indices, from the distribution and convert the latent variables with the target position back to the angle solutions through a decoder. It should be noted that we do not really sample latent variables, because backpropagation cannot handle sampling. Therefore, the reparameterization technique is used to implement the idea [17]. The original equation in VAE [17] is modified slightly by adding extra conditions. The loss function is as follows:

\[
\text{Loss} = -\int q(z|g, x) \cdot \log p(x|g, z) \, dz + KL(q(z|g, x) \parallel \text{Nor}(0, 1)),
\]

where \( g \) is the target position, \( x \) is the corresponding angle solutions with respect to \( g \), and \( z \) is the posture index. The objective is to minimize the equation above.

To ensure the angle solutions generated by the posture indices that are sufficiently distinct, we can calculate their similarities, e.g., in terms of Euclidean distance, of posture indices and only keep dissimilar ones. Finding a suitable threshold \( \lambda \) to determine the dissimilarity of the posture indices depends on the nature of the robotic arm that determines the distribution of the angle solutions in its working space, and thus will be left for future study.
Dictionary building

The dictionary contains target positions in the collected ground-truth dataset and their corresponding posture indices. It can be queried with a target position and will output a posture index, which can be fed into the joint angle calculation model to generate an angle solution. The dictionary can be built after the joint angle calculation model is trained in the previous step. Specifically, we leverage the encoder part of the neural network and feed the target positions in the ground-truth dataset through the encoder once more to obtain their posture indices. We then store the posture indices that are sufficiently distinct (> λ) into the dictionary.

During inference, if the given target position is in the ground-truth dataset, we can simply look up the dictionary to select a corresponding posture index and feed it to the joint angle calculation model to obtain the angle solution. If the given target position is not in the dictionary, i.e., was not reached during ground-truth dataset collection, we will look through the dictionary to find the position that is nearest to the given target position and use its posture indices. The regional goal alignment technique discussed in the next subsection can then be used to adapt these posture indices for the given target position. The rationale behind this is that the postures to reach a certain target position can also reach the positions around that position with minor adjustments.

To efficiently find the nearest position in the dictionary, we build a KD-tree to track the geometric relationship of the positions in the ground-truth dataset. A KD-tree is a space-partitioning data structure for organizing geometric data. Fig. 4 shows the data structures. A KD-tree comprises the positions in the ground-truth dataset to facilitate the search for the nearest position for the given target position. The dictionary contains the positions in the ground-truth dataset with their posture indices.

2.3 Online Processing

As mentioned earlier, given a target position during runtime, we will first search the KD-tree for the nearest position and then look up the dictionary for the corresponding posture indices. The obtained posture indices need to be adjusted slightly to fit the given target position. This is done through the regional goal alignment technique discussed in Section 2.3 After the posture indices are properly aligned, one index is selected for feeding into the joint angle calculation model to obtain an angle solution.

Regional goal alignment

We apply kernel-weighted prediction to adapt the posture indices of the nearest position in the ground-truth dataset to fit the given target position. A predicted set of posture indices is obtained as a kernel-weighted combination of the posture indices in the ground-truth dataset. Specifically, we denote \( I_{\text{given}} \) as the set of posture indices corresponding to the given target position and \( I_{\text{nearest}} \) as the set of posture indices of the nearest position in the ground-truth dataset. The kernel function \( k(given, nearest) \) is a measure to generalize indices from \( I_{\text{nearest}} \) to \( I_{\text{given}} \). Any kernel function can be used here to predict the set of posture indices for the given target position. The approximation can be denoted as:

\[
I_{\text{given}} = k(given, nearest) \times I_{\text{nearest}}.
\]

If the ground-truth dataset is dense enough, which means the difference between the given target position and the nearest position in the ground-truth dataset is close to zero, the kernel function can be seen as an identity matrix. We call this neighboring approximation strategy Regional Goal Alignment (RGA).
In our experiments, we found that an identity matrix is sufficient for getting an excellent result with a sub-centimeter error, and we believe our ground-truth dataset, which was collected with a 30-degree interval, is not difficult to collect, process, and store. Hence, the effects of different kernel functions on the overall performance will be left for future study.

3 Experiments

This section is organized as follows. In Section 3.1 details of the experimental setup are given. In Section 3.2 the diversity of the multiple angle solutions to reach a given target position is examined. In Section 3.3 we estimate the overall position accuracy of the proposed approach. In Section 3.4 properties of the posture indices are studied, and in Section 3.5 the computation time of our approach is evaluated. Finally, Section 3.6 shows the results using the target position plus the orientation as the given goal.

3.1 Experimental Setup

We evaluate the proposed methods using a 7-DoF robotic arm, Franka Emika Panda Arm\(^2\). Joint 1, which is the axis nearest the base, controls the orientation of the whole arm. Joint 4 is an elbow joint, which gives more versatility to the end-effector. The first three axes (Joints 1, 2, and 3) determine the position of the elbow joint, and the remaining axes determine the pose of the end-effector. The rotation limits of the joints are as follows: Min/Max (degree) = A1: -166/166, A2: -101/101, A3: -166/166, A4: -176/-4, A5: -166/166, A6: -1/215, A7: -166/166. The experiments were conducted entirely in a virtual environment, the PyBullet\(^7\) physics simulator, for fast experiments and avoiding unexpected collisions and damages if a real arm were used. In the virtual environment, physical parameters of the robot arm are the same as those of the real robot, except the self-collision mechanism was turned off.

Our ground-truth dataset consists of the motor angles of the arm, coordinates (x, y, z) of the end-effector, and the orientation of the end-effector. The ground-truth dataset was collected every 30 degrees of each motor and thus is very sparse. The ground-truth dataset contains 6,967,296 data points with a total size of 589.5 MB. From the collected data, we identified 580,608 different positions, resulting in about 12 data points per position on average. Note that it does not mean there are only 12 postures per position. Since the ground-truth dataset is sparse, it is not possible to obtain all the angle solutions to the same target position. To cope with the sparsity and to generate a sufficient number of angle solutions to a target position for selection, we actually include all the data points that can reach within 1 cm from the given target position as the angle solutions to that position. Hence, in Section 3.2, we also considered the posture indices from the neighboring nodes that are close to the target position for selection.

To implement SIK, the encoder and decoder each had five fully connected layers. The encoder had 2048, 2048, 1024, 27.25, 24.57, and 4 neutrons in the five layers respectively, whereas the decoder had 512, 1024, 2048, 1024, 512, and 7 neutrons. Each layer was followed by an ELU activation function, except for the output layer. The posture index produced by the encoder consists of four floating-point numbers. We have tried different parameters to train the neural networks of SIK. It is found that the learning rate should be less than or equal to 0.0005, decreased after every 1000 training epochs, and the batch size should be 65536. Stable and satisfactory results can be obtained for the entire working space after 3300 epochs. In PSIK, the layers are almost the same as in SIK, but the output layer of the encoder generates two outputs: one stands for the mean, and the other stands for the variance.

After the neural networks were trained, the ground-truth data points were fed through the networks once more to generate the posture indices and build the dictionary, following the procedures described in Section 2.2. In the dictionary, the maximum values of the posture indices, represented with four floating-point numbers, using SIK are: 31.62, 23.00, 27.25, 24.57, and the minimum values are: -34.04, -21.83, -29.92, -26.02. The average values of the posture indices with SIK are: -0.61, 0.38, 0.36, -0.26. In PSIK, the maximum mean/variance of the posture indices are: 5.70, 5.31, 7.99, 4.44/0.20, 0.07, 0.19, 0.07, and the minimum mean/variance are: -5.00, -4.98, -6.78, -4.48/0.01, 0.00, 0.00, 0.00. The average values of the posture indices with PSIK are: -0.01, 0.02, 0.01, 0.00, 0.01, 0.01, 0.01, 0.01.

The proposed approach is compared against a baseline design, which is a modified iterative method with a random previous position. Note that numerical methods solve the IK problem by iteratively reducing the distance between the target position and the current position of the end-effector. Therefore, the final posture of the robotic arm depends on its starting point. On the other hand, data-driven methods produce the angle solutions directly and the starting point is immaterial. Therefore, to make the comparison between SIK and the baseline meaningful, we randomly sampled ten locations around the target position as the previous state to obtain different angle solutions with the baseline. More specifically, in the baseline method, the robotic arm first approaches a random position that is within 50 cm of the target

---

\(^2\)https://www.franka.de/

---
Table 1: The variance of the angle for each joint in the four target positions using SIK, PSIK, and the baseline, based on 10 different angle solutions. Joint 1 controls the base of the robotic arm, whereas Joint 7 controls the rotation of the end-effector (gripper).

| Desired goal | Joint 1 | Joint 2 | Joint 3 | Joint 4 | Joint 5 | Joint 6 | Joint 7 |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| SIK          |         |         |         |         |         |         |         |
| a            | 141.23  | 70.59   | 175.55  | 5.75    | 178.47  | 54.69   | 133.93  |
| b            | 194.92  | 56.36   | 188.62  | 15.60   | 213.78  | 147.51  | 114.54  |
| c            | 166.66  | 99.60   | 176.19  | 2.99    | 142.76  | 109.93  | 51.89   |
| d            | 110.25  | 57.55   | 255.89  | 1.25    | 335.27  | 131.81  | 126.34  |
| PSIK         |         |         |         |         |         |         |         |
| a            | 138.53  | 70.26   | 176.80  | 5.69    | 179.89  | 54.97   | 0.01    |
| b            | 205.18  | 56.01   | 159.10  | 17.50   | 249.05  | 121.48  | 71.94   |
| c            | 164.97  | 98.49   | 178.18  | 3.03    | 142.63  | 109.57  | 0.01    |
| d            | 172.87  | 55.37   | 249.41  | 1.28    | 341.17  | 131.19  | 1.50    |
| Baseline     |         |         |         |         |         |         |         |
| a            | 0.64    | 6.21    | 58.19   | 31.62   | 9.12    | 0.58    | 0.00    |
| b            | 182.98  | 176.04  | 47.71   | 0.00    | 38.85   | 0.01    | 0.00    |
| c            | 232.72  | 87.02   | 101.10  | 0.00    | 3.80    | 0.00    | 0.00    |
| d            | 2.03    | 6.07    | 38.30   | 41.55   | 2.45    | 1.63    | 0.00    |

...
Selective Inverse Kinematics: A Novel Approach to Finding Multiple Solutions Fast for High-DoF Robotic Arms

Table 2: Illustrations of the ten different angle solutions in the four target positions

| No. 1 | No. 2 | No. 3 | No. 4 | No. 5 | No. 6 | No. 7 | No. 8 | No. 9 | No. 10 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
| ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] |
| ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] |
| ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] |
| ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] | ![image] |

Table 2 shows more details of the four target positions and their distance errors using SIK and PSIK. The first column is the positions in meters, and the remaining columns show the Euclidean distance errors of the end-effector in centimeters. In SIK, the average distance errors are between 0.4 cm to 0.5 cm, whereas PSIK has average distance errors between 1.1 cm to 1.5 cm. On the other hand, the average distance errors with the baseline in the four positions are 0.40 cm, 0.69 cm, 0.19 cm, and 0.32 cm, respectively. The total average error is about 0.4 cm.

To sum up, the multiple angle solutions obtained from the proposed methods to reach a given target position are quite distinct, and the average distance errors of the end-effector are about 0.5 cm and 1 cm. SIK and PSIK can both produce distinct, controllable, and reproducible postures. The baseline, by contrast, can only obtain random solutions with minor variations. In terms of distance errors, the baseline seems to outperform PSIK. However, it should be noted that our ground-truth dataset is very sparse. The distance errors of PSIK can easily be reduced to sub-centimeter if denser data are collected, as shown in later experiments.

Finally, Fig. 6 demonstrates an application of SIK for obstacle avoidance. In this example, there is a cube on the table, and the arm has to move the end-effector from the initial point (left) to the destination (right). Since the proposed SIK provides several different angle solutions to reach the target position, we can choose a better one that can avoid obstacles. Compared to SIK, the baseline iterative method only tries to move the end-effector to the target position as
Table 3: The distance errors (cm) of the ten angle solutions in the four target positions using SIK and PSIK

| Goal (meter): (x, y, z) | No. 1 | No. 2 | No. 3 | No. 4 | No. 5 | No. 6 | No. 7 | No. 8 | No. 9 | No. 10 | Avg. |
|------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|------|
| SIK                    |       |       |       |       |       |       |       |       |       |        |      |
| (0.1357, -0.321, 0.222) | 0.22  | 0.25  | 0.54  | 0.22  | 1.6   | 0.29  | 0.56  | 0.19  | 0.61  | 0.48   |      |
| (-0.333, 0.246, 0.1999)| 0.35  | 0.25  | 0.48  | 0.4   | 0.36  | 0.38  | 0.42  | 0.34  | 0.35  | 0.37   |      |
| (-0.188, -0.168, 0.321)| 0.74  | 0.24  | 1.04  | 0.57  | 0.40  | 0.63  | 0.43  | 0.45  | 0.31  | 0.61   | 0.54 |
| (0.2020, 0.2021, 0.2022)| 0.21  | 0.29  | 0.31  | 0.47  | 1.08  | 0.30  | 0.46  | 0.40  | 0.46  | 0.71   | 0.47 |
| PSIK                   |       |       |       |       |       |       |       |       |       |        |      |
| (0.1357, -0.321, 0.222) | 0.98  | 0.65  | 1.03  | 0.82  | 1.70  | 1.28  | 3.24  | 0.63  | 2.72  | 1.43   |      |
| (-0.333, 0.246, 0.1999)| 0.93  | 1.51  | 1.46  | 2.44  | 1.32  | 0.40  | 0.74  | 1.48  | 1.13  | 1.83   | 1.32 |
| (-0.188, -0.168, 0.321)| 0.94  | 1.45  | 3.35  | 1.26  | 0.51  | 1.64  | 0.77  | 1.09  | 0.29  | 0.63   | 1.19 |
| (0.2020, 0.2021, 0.2022)| 1.03  | 0.27  | 1.04  | 2.18  | 2.05  | 1.31  | 1.18  | 1.39  | 2.94  | 1.17   | 1.46 |

Figure 6: An example of approaching the target position, in which the robotic arm attempts to grasp the green object without colliding with the cube. The first row shows the trajectory calculated by the baseline and the second row by SIK, which can avoid the obstacle if an appropriate posture index is provided.

close as possible, without considering the obstacles. To avoid obstacles, extra path planning schemes are needed that calculate intermediate positions to guide the end-effector to move around the obstacles.

### 3.3 Accuracy and Data Density

In the previous subsection, we randomly sampled four positions in the working space of the robotic arm and studied the resultant angle solutions in depth. In this subsection, we study the proposed methods considering the whole working space and examine the effects of the density of collected data.

We first evaluate the accuracy of the proposed methods over the entire working space. This is done by sampling 100 desired goals uniformly from the working space based on the ground-truth dataset that has a 30-degree per joint resolution. The average distance error is then computed. For SIK, the average distance error is found to be about 0.5 cm after 3300 epochs, while for PSIK, the error is 1.56 cm. As mentioned in Section 3.1, PSIK is more difficult to train. The KL divergence prevented PSIK from fitting the training dataset, which is similar to a regularization term in training, so the accuracy of PSIK is limited. The subcentimeter error by SIK can match that by the iterative approximation methods, such as the baseline, while independent of the starting point.

We note that the ground-truth dataset has different densities in different regions of the working space, due to the nature of the robotic arm and the way the data were collected, i.e., based on the angles of the joints. Thus, to further examine the performance of our approach in different regions of the working space, we divided the working space with different
sizes of circles centered around the base of the robot and sampled 100 random target positions on each of the circles. The resultant average distance errors are shown in Tables 4.

Table 4 shows that SIK has worse performance when the end-effector is near the body of the arm. It performs even worse than PSIK. The best region for SIK is around 20 cm to 50 cm, where SIK has a 0.5 cm average distance error. Compared with SIK, the average distance errors in PSIK are relatively uniform across different regions, because PSIK treats the ground-truth dataset as distributions during training and considers nearby data points together. This also allows PSIK to infer angle solutions around edge regions, i.e., the radius smaller than 10 cm, by extending the sparse data points. Unfortunately, both methods perform poorly on 25 cm radius due to the low data density.

As shown in the beginning of this subsection, the PSIK has an average 1.56 cm distance error over the entire working space using a ground-truth dataset with a resolution of 30-degree per joint. To show that the distance error can be reduced within one centimeter, ground-truth dataset with higher resolution should be collected and evaluated. Unfortunately, the hardware resources available to us prevented us from collecting data with a one-degree per joint resolution, which would result in more than 9e+16 data points and excessive time for training. Hence, we narrowed the working space down to 10 degrees per joint and collected a ground-truth dataset with a one-degree per joint resolution. The target positions for testing were generated by randomly selecting seven numbers between 0 to 10 as the angle degrees of the seven joints in the narrowed working space and then calculating the position of the end-effector using forward kinematics. The resulting average distance error using the PSIK method can be as small as 0.3 cm.

3.4 Posture Indices

Our approach represents the postures of the robotic arm numerically using the 4-dimensional posture indices. Intuitively, every element in the posture indices should affect some aspects of the final posture of the arm. Unfortunately, the unsupervised learning that we used to obtain the posture indices lacks semantic explanations, though it allows our approach to generalize to any DoF robotic arms. This section tries to exploit the properties of the posture indices and compares those produced by SIK and PSIK.

The first question we attempt to answer is: “What dimensionality should the posture index have?” Since high-DoF IK is very new in robotics, we cannot find any related literature to assist us in answering this question. Hence, we tried different dimensionalities from one to seven in the experiments, because we used a 7-DoF robotic arm as the target robotic arm and it is unreasonable if the dimension of the posture indices is greater than the number of the joints. Our experiments show that the models with a dimension smaller than three failed to fit the training data. The loss values of these models stopped decreasing in an early stage. On the other hand, the models with a dimension greater than or equal to three can successfully achieve a good comparable performance. The main difference among them is the speed to converge. The longer the indices are, the faster the loss value decreases. Since the models with 3-dimensional indices need more hyper-parameter adjustments and training time, we therefore use 4-dimensional posture indices in most experiments discussed in this section.

Next, we study the effects of changing different elements in the posture indices. Let us start with 4-dimensional posture indices. Table 5 shows the changes of the postures when different elements in a posture index are changed for the target position (-0.25, -0.25, 0.25). There are two different trends in the postures when the robotic arm tries to reach the given target position. One affects the orientation of the end-effector (elements 1 & 4), and the other affects the body of the robotic arm but keeps both the orientation and the location fixed (elements 2 & 3). More specifically, element 1 makes the end-effector rotate around the vertical line, whereas element 4 rotates it around the horizontal line. In other words, we can change the two elements to control the direction that the end-effector approaches the target position. Fig. 7 shows the results when we adjust both elements 1 and 4. As we can see, the end-effector moves along slant lines.

On the other hand, elements 2 and 3 of the posture indices change the slope of the robotic arm. They cause similar effects except for the side. Such effects are a result of the extra redundancy in high-DoF robotic arms. Note that robotic arms with DoF fewer than seven can only have one solution to reach a target position when they need to keep a fixed orientation.

We next study how the roles of the elements in the posture indices change when the dimensionality of the indices is changed. We examine 3-dimensional posture indices first and observe the effects of adjusting each of the three
Figure 7: Since we know which elements in the posture index control the orientation of the end-effector, we can adjust the elements to make the end-effector move along slant lines and still keep touching the target position.

Table 5: Attributes of the posture indices in target position (-0.25, -0.25, 0.25)

| Element 1 | Error (cm) |
|-----------|------------|
|           | 0.08 | 0.24 | 0.6 | 0.81 | 0.65 | 0.51 |
| Element 2 | Error (cm) |
|           | 0.06 | 0.15 | 0.18 | 0.08 | 0.49 | 0.6 |
| Element 3 | Error (cm) |
|           | 0.03 | 0.58 | 0.66 | 0.38 | 0.37 | 0.38 |
| Element 4 | Error (cm) |
|           | 0.03 | 0.11 | 0.18 | 0.19 | 0.42 | 0.86 |

It is found that two of the three elements affect the orientation. This implies that they are essential so that the end-effector can approach the target position from any direction in a 3D space. The remaining element alters the slope, which means that the two elements that do the same in 4-dimensional posture indices are now merged into one due to their similarity. This also explains why the dimension of the posture indices has to be at least three.

When the dimensionality of posture indices increases, the trained models can have more room to contain the two types of changing trends, which eases the difficulty of training. However, many elements in the posture indices may cause similar effects. In summary, the dimensionality of posture indices should be at least two (for affecting the orientation) plus the DoF of the target robotic arm minus six, i.e., the number of extra redundancies.

Finally, we give a brief qualitative comparison of SIK and PSIK, after we have examined various aspects of these two methods. Both SIK and PSIK can obtain distinct postures for given goals, and they have similar diversity in the angle solutions. With respect to accuracy, SIK is much better than PSIK and is easier to train, because PSIK has more constraints and requires the posture indices to obey probabilistic distributions. However, PSIK has very uniform distance errors across the entire working space. This stabilizes the overall performance and extends the ground-truth dataset smoothly to overcome the sparsity in collected training data.

Regarding the characteristics of the obtained posture indices, SIK generates arbitrary floating-point numbers, whereas PSIK produces normalized numbers. If the application is just to obtain different angle solutions, then the normalization
Table 6: Comparison of computing time. The computing time of RNEA is the time that RNEA calculates the same one-hundred desired goals.

| Method          | DoF | Computing time (sec) |
|-----------------|-----|----------------------|
| Random IW-PSO   | 7   | 1.6                  |
| baseline        | 7   | 0.007                |
| SIK             | 7   | 0.0044               |
| PSIK            | 7   | 0.0044               |

will not change the usage. We only need to retrieve the numbers from the dictionary and feed them into the joint angle calculation models to get the angle solutions. However, if the application requires changing the posture of the robotic arm from a given initial posture, then PSIK may be a better choice. The normalization of the posture indices means that all the numbers of the indices are bounded in a known range and the impacts of the postures are equally distributed in the range. Therefore, users have better controls in adjusting the elements in the posture indices.

3.5 Computing Time

To evaluate the computation time, we measure the search time, which is the time to find the positions in the KD-tree that are nearest to the given target position plus the time to look up the dictionary of the posture indices. We also measure the execution time of running the SIK/PSIK models to generate the angle solutions. The Python \texttt{time.clock()} function, which returns the current processor time in seconds, is used to estimate the computation time. It was assumed that the KD-tree and the corresponding dictionary of posture indices had been established. We randomly sampled 100 testing points across the entire working space for estimating the average computation time.

The results show that the search time is about 0.0004 seconds, and the SIK/PSIK models take about 0.004 seconds to obtain the solution. Since SIK and PSIK have the same number of neurons, their execution time to calculate the angle solutions are the same. To fairly compare with the baseline, we only take the time when the baseline calculates the final target position as the computation time. Table 6 shows the comparison of the three methods in terms of computing time. The proposed methods only need half of the computation time of the baseline to calculate an angle solution. This is mainly because we have already learned the relationship between the target position and the angle solutions through a delicate function approximation, which is conceptually equal to a closed-form equation. Therefore, the proposed data-driven approaches can save a lot of time.

3.6 Orientation as Goal

In this subsection, we examine SIK when the desired goals include target position as well as orientation. The implementation is the same as those presented in Section 3.1 except for the length of the inputs to the joint angle calculation models, which becomes six plus the size of the posture indices. As mentioned in Section 3.4, we found that the dimension of the posture indices has to be at least three, in which one element controls the slope of the arm and the other two control the orientation of the end-effector. Now, the goal requires orientation also, so the orientation of the end-effect is given. That means we cannot change the orientation for such goals. It follows that we may be able to reduce the length of the posture indices from three to one.

We evaluated the distance errors and the cosine similarity of the orientation of the end-effector in four different desired goals using 1-dimensional posture indices. The average distance error is 1.35 cm, and the average cosine similarity is 0.9983, which means the deviation of the orientation is about 3 degrees. This experiment validates that SIK can also be applied if the desired goal includes both target position and orientation. Table 7 illustrates the five postures generated from the five posture indices to reach the four target positions while maintaining the given orientations. We can easily see the similar effects as discussed in the previous sections considering only target positions. We also trained SIK using 3- and 4-dimensional posture indices. It turned out that the extra elements cause the same changing trend, i.e., slope adjustment, but not orientation.

4 Related Work

Inverse kinematics (IK) calculates the angle displacement for each joint of a robotic arm such that the end-effector can reach the given target position with the given orientation in the working space. In other words, IK attempts to map from the robot working space (a Cartesian space) to its joint configuration space. Traditional solutions to the IK problem
Table 7: Illustrations of the five different angle solutions in the four target positions considering both target position and orientation using 1-dimensional posture indices

| No. 1 | No. 2 | No. 3 | No. 4 | No. 5 |
|-------|-------|-------|-------|-------|
| ![Illustration a](image1) | ![Illustration b](image2) | ![Illustration c](image3) | ![Illustration d](image4) | ![Illustration e](image5) |

can be broadly categorized into two main classes: analytical (closed-form) methods [1, 3, 23, 27, 30] and numerical methods (Jacobian-based) [5, 22, 29].

Analytical methods use explicit mathematical formulations to solve the IK problem with closed-form expressions. They can precisely determine all possible IK solutions. The biggest shortcoming of analytical methods is the need to solve the algebraic formulas that are very complex and difficult. Another problem is that analytic IK solutions require full knowledge of the kinematic structure of the robotic arm, and so far the whole process is very difficult to automate. Numerical methods, on the other hand, obtain one IK solution through iterative approximation. They avoid the complex formulation process. However, numerical methods are time-consuming in the iterative process. The computation time of a numerical method can be 100 times slower than an analytical one.

In addition to the two traditional IK solutions, data-driven approaches [2, 7, 8, 10, 12, 15, 21, 24, 26, 28] have also shown great potentials in solving the IK problem recently. Most of them applied a neural network to map the target position to one angle solution. Unfortunately, this is inadequate for high-DoF robotic arms. The redundant DoFs lead to extra flexibility of high-DoF robotic arms that can exploit different angle solutions for the same given goal. Existing data-driven methods can only find one angle solution, which greatly limits the applications of high-DoF robotic arms.

In [2], an accurate solution for the IK problem by using a neural network is introduced. The main idea is that the neural network for solving the IK problem can be improved if the current state of the robotic arm is added to the input. Their training dataset can only have a unique joint configuration in both input and output sets because their model can only find one angle solution. The experiments showed high accuracy by their model in robot motion control. Compared with this work, our approach can find multiple angle solutions, and we do not need a specialized training dataset. Our model can find the angle solutions for any target position (plus the orientation) in the entire workspace, while their model only works by following specific paths.

In [7], a machine learning approach to solving the IK problem was proposed that can eliminate the need for developing the equations by hand. In [26], a work that can solve the IK of a 3-DoF robotic arm in 3-dimensional space was introduced, which does not need a large dataset for training and can reduce the neural network structure. In [15], an IK solution by combining Genetic Algorithm and Neural Network was proposed. The above three studies again solved the IK problem by providing only a single angle solution, and they need to put extra efforts into designing customized training datasets, e.g., space-filling curves. Furthermore, these works only considered low-DoF robotic arms (2-DoF and 3-DoF), and it is unclear how they may be extended to high-DoF arms.

The work in [12] introduced two approaches to the IK problem for the Tricept robot: one is based on the MLP neural network, and the other is based on the RBF neural network. The proposed approaches can only find one angle solution for a given goal and consider only low-DoF robots. The study in [8] discussed solving the IK problem of multiple robotic arms using Artificial Neural Networks and Adaptive Neuro-Fuzzy Inference Systems. Similarly, this work can
Selective Inverse Kinematics: A Novel Approach to Finding Multiple Solutions Fast for High-DoF Robotic Arms

A PREPRINT

only obtain one angle solution for a target position. Although it experimented with 4, 5, 6, and 7-DoF robotics, the accuracy is poor, over a few centimeters. By contrast, our approach has a low distance error (smaller than 0.5 cm).

Two solutions were proposed in [23] for the IK problem of an industrial parallel robot: a closed analytical form and a Deep Learning approximation model based on three different networks (MLP, LSTM, GRU). The training data was collected through a user-defined point cloud and a 3-DoF parallel robot, the IRB360, is used for experiments. Again, the work ignored the flexibility of high-DoF robotic arms by considering only the unique solutions confined by the point cloud. It is unclear how the proposed work performs for high-DoF robotic arms.

In [21], a new method for learning a mapping between redundant states and low-dimensional postures was proposed. The work considered a high-DoF, complex musculoskeletal robot and attempted to find sets of internal pressures of pneumatic artificial muscles to meet a target position. An autoencoder was developed to handle the high-dimensional data, and supervised learning was used to learn known low-dimensional corresponding vectors. The proposed model had a relatively high distance error (~10 cm), and the characteristics of the vectors learned from the autoencoder were not studied. In contrast to their work, we showed a comprehensive analysis of the posture indices in this paper, and our model can achieve very good performances in both speed and accuracy.

5 Conclusions

We present in this paper a new data-driven approach to the IK problem for high-DoF robotic arms without knowing the kinematics information. Our approach can obtain multiple angle solutions for a given target position (plus a target orientation) and has a low distance error (~0.5 cm) for any position in the entire working space of the robot. The proposed methods do not impose any special constraints on training data and thus can be applied to any DoF robotic arm. Compared with the latest data-driven methods, our work signifies a major step forward in solving the IK problem for high-DoF robotic arms. This is the first data-driven research to address multiple angle solutions in IK, and a comprehensive study of the various properties of the proposed method is provided.

References

[1] MA Ali, HA Park, and CSG Lee. Closed-form inverse kinematic joint solution for humanoid robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 704–709. IEEE, 2010.
[2] A R Almusawi, L C Dülger, and S Kapucu. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242). Computational intelligence and neuroscience, 2016.
[3] T Asfour and R Dillmann. Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), volume 2, pages 1407–1412. IEEE, 2003.
[4] Y Bengio. Learning deep architectures for AI. Now Publishers Inc, 2009.
[5] SR Buss and JS Kim. Selectively damped least squares for inverse kinematics. Journal of Graphics tools, 10(3):37–49, 2005.
[6] E Coumans and Y Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.
[7] A Csizsor, J Eilers, and A Verl. On solving the inverse kinematics problem using neural networks. In 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), pages 1–6. IEEE, 2017.
[8] J Demby’s, Y Gao, and G N DeSouza. A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network. In 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pages 1–6. IEEE, 2019.
[9] S Dereli and R Köker. Iw-psp approach to the inverse kinematics problem solution of a 7-dof serial robot manipulator. Sigma J Eng Nat Sci, 36(1):77–85, 2018.
[10] AV Duka. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. Procedia Technology, 12:20–27, 2014.
[11] R Featherstone. Rigid body dynamics algorithms. Springer, 2014.
[12] Jamal Ghasemi, R Moradinezhad, and Mir Amin Hosseini. Kinematic synthesis of parallel manipulator via neural network approach. arXiv preprint arXiv:1904.04668, 2019.
[13] RE Goldman, A Bajo, L S MacLachlan, R Pickens, SD Herrell, and N Samaan. Design and performance evaluation of a minimally invasive telerobotic platform for transurethral surveillance and intervention. IEEE transactions on biomedical engineering, 60(4):918–925, 2012.
[14] P Huang, F Zhang, L Chen, Z Meng, Y Zhang, Z Liu, and Y Hu. A review of space tether in new applications. Nonlinear Dynamics, 94(1):1–19, 2018.
[15] H Z Khaleel. Inverse kinematics solution for redundant robot manipulator using combination of ga and nn. Al-Khwarizmi Engineering Journal, 14(1):136–144, 2018.
[16] DP Kingma, DJ Rezende, S Mohamed, and M Welling. Semi-supervised learning with deep generative models. arXiv preprint arXiv:1406.5298, 2014.

[17] DP Kingma and M Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[18] T Liu, Z Mu, H Wang, W Xu, and Y Li. A cable-driven redundant spatial manipulator with improved stiffness and load capacity. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6628–6633. IEEE, 2018.

[19] T Liu, Z Mu, W Xu, T Yang, K You, H Fu, and Y Li. Improved mechanical design and simplified motion planning of hybrid active and passive cable-driven segmented manipulator with coupled motion. In IROS, pages 5978–5983, 2019.

[20] JYSM Luh, M Walker, and R Paul. Resolved-acceleration control of mechanical manipulators. IEEE Transactions on Automatic Control, 25(3):468–474, 1980.

[21] H Masuda, A Hitzmann, K Hosoda, and S Ikemoto. Common dimensional autoencoder for learning redundant muscle-posture mappings of complex musculoskeletal robots. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2545–2550. IEEE, 2019.

[22] Y Nakamura and H Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. 1986.

[23] S Neppalli, MA Csencsits, BA Jones, and ID Walker. Closed-form inverse kinematics for continuum manipulators. Advanced Robotics, 23(15):2077–2091, 2009.

[24] KD Polyzos, PP Groumpos, and E Dermatas. Solving the inverse kinematics of robotic arm using autoencoders. In Conference on Creativity in Intelligent Technologies and Data Science, pages 288–298. Springer, 2019.

[25] K Sohn, H Lee, and X Yan. Learning structured output representation using deep conditional generative models. Advances in neural information processing systems, 28:3483–3491, 2015.

[26] P Srisuk, A Sento, and Y Kitjaidure. Inverse kinematics solution using neural networks from forward kinematics equations. In 2017 9th international conference on Knowledge and Smart Technology (KST), pages 61–65. IEEE, 2017.

[27] D Tolani, A Goswami, and NI Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. Graphical models, 62(5):353–388, 2000.

[28] J S Toquica, P S Oliveira, W SR Souza, J M ST Motta, and D L Borges. An analytical and a deep learning model for solving the inverse kinematic problem of an industrial parallel robot. Computers & Industrial Engineering, 151:106682, 2021.

[29] CW Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Transactions on Systems, Man, and Cybernetics, 16(1):93–101, 1986.

[30] I Zaplana and L Basanez. A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis. Mechanism and machine theory, 121:829–843, 2018.

[31] F Zhang and P Huang. A novel underactuated control scheme for deployment/retrieval of space tethered system. Nonlinear Dynamics, 95(4):3465–3476, 2019.