Configuration-Dependent Robot Kinematics Model and Calibration

***Abstract*— Accurate robot kinematics is essential for precise tool placement in articulated robots, but non-geometric factors can introduce configuration-dependent model discrepancies. This paper presents a configuration-dependent kinematic calibration framework for improving accuracy across the entire workspace. Local Product-of-Exponential (POE) models, selected for their parameterization continuity, are identified at multiple configurations and interpolated into a global model. Inspired by joint gravity load expressions, we employ Fourier basis function interpolation parameterized by the shoulder and elbow joint angles, achieving accuracy comparable to neural network and autoencoder methods but with substantially higher training efficiency. Validation on two 6-DoF industrial robots shows that the proposed approach reduces the maximum positioning error by over 50%, meeting the sub-millimeter accuracy required for cold spray manufacturing. Robots with larger configuration-dependent discrepancies benefit even more. A dual-robot collaborative task demonstrates the framework's practical applicability and repeatability.**

Keywords: Robot Calibration, Kinematics Model, Configuration Dependent Calibration, Absolute Accuracy, Product-of-Exponentials

I. INTRODUCTION

Industrial robot arms are increasingly utilized in manufacturing processes demanding high accuracy, such as cold spray [1], deep rolling [2], and surface grinding [3]. To ensure that the robot tool center point (TCP) follows the desired path using joint control, accurate robot kinematic parameters are needed. Using kinematics parameters directly from the vendor data sheet often results in TCP errors beyond the vendor specification. The kinematics error may be caused by geometrical and non-geometrical factors [4]. Geometrical factors, such as misalignment in joint axes during robot production and mismatch in encoder position and joint angle readings, are typically not dependent on the arm configuration (e.g., out-stretched vs. folded). The non-geometrical factors, such as link and joint flexibility, motor backlash, etc., depend on the robot configuration due to the different loading conditions on the robot joints.

Robot calibration is the process of using external TCP measurements (considered as the ground truth) and robot joint angle readings to fit a kinematic model, so that joint angles can predict the TCP location sufficiently accurately during runtime. Fig. 1 shows an example of the external TCP measurement using a motion capture system from our robot welding testbed.

There are four key steps in robot calibration: modeling, measurement, identification, and compensation [5]. The critical aspect of this process lies in effectively parameterizing the robot kinematic model. Common approaches include the Denavit-Hartenberg (DH) parameterization [6] and the

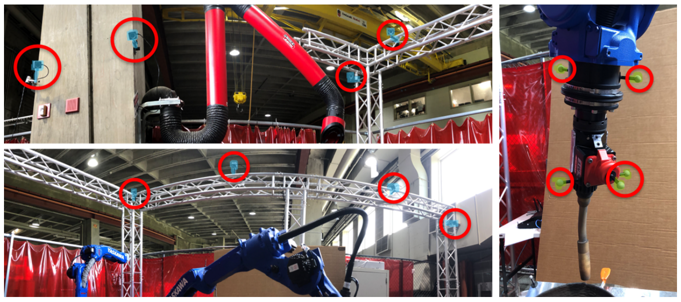


Fig. 1: Experimental robot calibration setup using the OptiTrack motion capture system.

Product-of-Exponentials (POE) model [7]. Although the DH model is widely adopted in both industrial and academic settings, it suffers from non-smooth parameterization and the presence of singularities, particularly in configurations with parallel joint axes [8]. To mitigate these issues, [9], [10] introduced an additional parameter for parallel or near-parallel joint axes, a strategy adopted by most subsequent works using the DH parameters, such as [11]–[13]. However, not only this approach is non-minimal, it is ad-hoc and further complicates the calibration process [14], [15]. The POE model offers a smooth and continuous parameterization, but is non-minimal. Recent works [16] imposes additional constraints to remove the redundancy in the POE parameterization for robot calibration.

Calibration for the geometrical factors has long been studied [7], [9] and is now routinely applied. Compensating for the non-geometric factors to achieve high accuracy over the entire workspace has received comparatively less attention. Many calibration methods for non-geometric factors focus on identifying joint flexibility [17]–[19]. These papers model joints as linear torsional springs and estimate the compliance coefficients of each joint. The Jacobian matrix of the tool offset with respect to the compliance coefficients is derived for optimization. Zhou et al. [17] simultaneously calibrates the geometric factors and the joint compliance coefficients. Kamali et al. [18] developed a cable system to apply various load to the robot to increase data collection efficiency. The parameters are optimized using damped least-square at different poses and loads. Although estimating the joint compliance is a promising approach, it requires the ability to measure the joint torques or the wrench applied on the tool, which may not be readily available. Several studies focus on model-free, configuration-dependent calibration methods, where the nominal kinematic parameters are first calibrated, followed by a data-driven model relating the residual position and orientation errors as a function of the joint configuration and tool payload [11], [20]–[22]. The configuration to residual error function is mostly approximated by a multi-layer feedforward Neural Network (NN) or radial basis functions (RBF). The NN prediction can further be implemented on top of the joint stiffness model to compensate for other non-geometric factors [19], [23], [24]. With extensive training, these black-box models can increase the tool location accuracy,

In this work, we adopt the POE model for robot parameterization. By requiring the identified model to be close to the nominal POE parameters, we remove the redundancy in the parameterization. Using local gradient search with data collected from clusters of robot configurations, we identify a set of POE parameters *near* the nominal parameters at each cluster. The full kinematics is constructed through interpolation of these local kinematic models. We found that the most effective interpolation scheme is to use a set of Fourier basis functions parameterized on the robot shoulder and elbow angles. This basis is motivated by the expression of the joint gravity load [25]. For comparison, we also train a multi-layer NN directly from the collected data. The performance is comparable, though the identification of the Fourier-basis model is much more efficient as no extensive training is required. We demonstrate our method on two separate 6R Motoman robots in our welding testbed. An OptiTrack motion tracking system is used for the tool frame measurements. Two sets of data are collected, one for model identification and one for performance evaluation. The test results show that the Fourier-basis method is able to reduce the maximum error by more than 50% to within 1 mm. To summarize, the main contribution of this paper are:

- A configuration-dependent kinematic calibration framework based on local minimal POE model identification and interpolation, which improves accuracy across the full workspace.
- A Fourier basis function approximation strategy parameterized by shoulder and elbow joint angles, offering an efficient and physically motivated alternative to pure data-driven methods such as neural networks.
- Experimental validation on two 6R industrial robots, demonstrating significant improvements in accuracy in reducing the maximum position error by more than 50% compared to the nominal model. The calibrated models were further evaluated in a dual-robot collaborative contact test, confirming their practical applicability.

The paper is organized as follows. Section II describes the robot kinematics and kinematic model calibration. Section III introduces our configuration-dependent kinematic model and the corresponding calibration frameworks. In section IV, we show the experimental results and demonstrate the enhanced accuracy using the proposed methodology.

II. Robot Kinematics Model and Calibration

A. POE Forward Kinematics Model

Consider an n-joint articulated robot arm with all revolute joints as shown in Fig. 2. Denote the reference (world) coordinate frame by an orthonormal frame and the origin: $(\mathcal{E}_0, \mathcal{O}_0)$. The ith frame, $\mathcal{E}_i$, is the rotation of the $i-1$th frame, $\mathcal{E}_{i-1}$, about the joint axis, $\vec{h}_i$, over the joint angle q_i. The joint axis in the $i-1$th frame (and the ith frame) is a constant vector, denoted by h_i. Denote $\mathcal{E}_i$ represented in $\mathcal{E}_j$ by the rotation matrix R_{ji}. The relative rotation between two consecutive frames is the rotation about the joint axis:

$$R_{i-1,i} = \mathsf{R}(h_i, q_i) \tag{1}$$

where $\mathsf{R}(h_i, q_i) = e^{h_i^{\times} q_i}$, $(\cdot)^{\times}$ is the skew symmetric matrix representing the cross product. The tool coordinate frame $\mathcal{E}_T$ represented in $\mathcal{E}_n$ is denoted by R_{nT}.

The origin of the ith frame, $\mathcal{O}_i$, may be arbitrarily chosen along h_i. Denote the link vector from $\mathcal{O}_{i-1}$ to $\mathcal{O}_i$ by $p_{i-1,i}$ which is a constant vector in $\mathcal{E}_{i-1}$. The same notation applies for the link vector from $\mathcal{O}_n$ to $\mathcal{O}_T$, denote by $p_{n,T}$. Define a homogeneous transformation from the $i-1$th frame to the ith frame as

$$T_{i-1,i} = \begin{bmatrix} R_{i-1,i} & p_{i-1,i} \\ 0 & 1 \end{bmatrix}. \tag{2}$$

Denote the robot tool frame (end effector position and orientation) by (p_{0T}, R_{0T}). Then the robot forward kinematics (mapping joint angles $\{q_i\}$ to the tool frame) is given by

$$T_{0T}(q) = T_{01}(q_1)T_{12}(q_2)\ldots T_{n-1,n}(q_n)T_{nT} \tag{3}$$

where q is a $n \times 1$ vector consisting of all joint angles. Note that T_{nT} is a constant transform:

$$T_{nT} = \begin{bmatrix} I & p_{nT} \\ 0 & 1 \end{bmatrix}. \tag{4}$$

The robot forward kinematics is completely parameterized

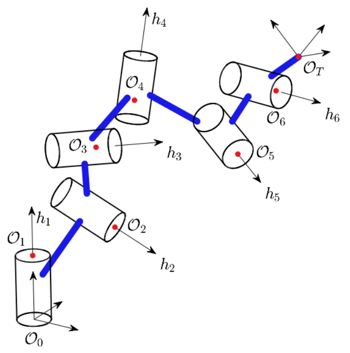


Fig. 2: Kinematic diagram of a 6R robot.

by the joint axes $\{h_i\}$ and link axes $\{p_{i-1,i}\}$. We can put them in the matrix form:

$$\mathbf{H} = \begin{bmatrix} h_1 & \ldots & h_n \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} p_{01} & \ldots & p_{nT} \end{bmatrix}. \tag{5}$$

The tool frame in forward kinematics (2) may be rewritten as below to highlight the dependence on the kinematic parameters:

$$\begin{aligned} T_{0T}(q; \mathbf{P}, \mathbf{H}) = T_{01}(q_1; p_{01}, h_1)\ldots \\ T_{n-1,n}(q_n; p_{n-1,n}, h_n)T_{n,T}(p_{nT}). \end{aligned} \tag{6}$$

The kinematic calibration problem is to use multiple measurements of the joint angles q and the corresponding tool frames T_{0T} to estimate $(\mathbf{P}, \mathbf{H})$. We assume the robot manufacturer has provided a set of nominal parameters:

$$\bar{\mathbf{H}} = \begin{bmatrix} \bar{h}_1 & \dots & \bar{h}_n \end{bmatrix}, \quad \bar{\mathbf{P}} = \begin{bmatrix} \bar{p}_{01} & \dots & \bar{p}_{nT} \end{bmatrix}. \tag{7}$$

We will first estimate $(\mathbf{P}, \mathbf{H})$ using data near a specified configuration (i.e., a given joint angle q). The process is then generalized to multiple configurations throughout the workspace. For an arbitrary configuration, we can interpolate from estimated $(\mathbf{P}, \mathbf{H})$ values in nearby configurations. We shall see that the interpolation algorithm affects the overall accuracy of the calibration result.

B. A Minimal POE Model

The POE parameterization is not minimal: The ith frame origin $\mathcal{O}_i$ may be placed anywhere on the ith joint axis, and the ith joint axis h_i is a unit vector, $\|h_i\| = 1$. To identify a unique POE model, we constraint the identified parameters to be *close* to the nominal parameters (Fig. 3). Denote $\bar{h}_i$ in the base frame as $(\bar{h}_i)_o := R_{0,i-1}\bar{h}_i$. Define $\bar{\pi}_i$ be the plane perpendicular to $(\bar{h}_i)_o$ and intersecting $\bar{\mathcal{O}}_i$, the ith origin in the nominal model. We require the identified origin $\mathcal{O}_i$ to be the intersection of the identified axis $(h_i)_o$ with $\bar{\pi}_i$. Let p'_{0i} be any vector from the robot base to the line $(h_i)_o$. Then $\mathcal{O}_i$ is given by the tip of the vector

$$p_{0i} = p'_{0i} + \frac{(\bar{p}_{0i} - p'_{0i})^T (\bar{h}_i)_o}{(h_i)_o^T (\bar{h}_i)_o} (h_i)_o. \tag{8}$$

Given the joint angle q, we can iteratively recover the constant vectors in $\mathbf{H}$ and $\mathbf{P}$ in (7) from $(h_i)_o$ and p_{0i}:

$$h_i = R_{0,i-1}^T (h_i)_o \tag{9a}$$
$$R_{0i} = R_{0,i-1} e^{h_i^\times q_i} \tag{9b}$$
$$p_{i-1,i} = R_{0,i-1}^T (p_{0i} - p_{0,i-1}) \tag{9c}$$

for $i = 1, \dots, n+1$, with $\mathcal{O}_{n+1} = \mathcal{O}_T$.

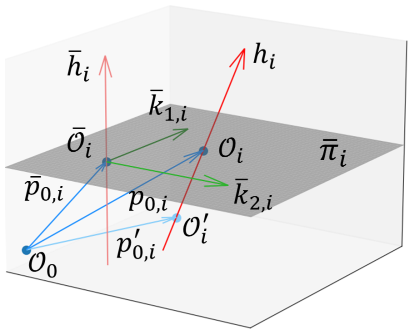


Fig. 3: Minimal POE parameterization based on a set of nominal POE parameters. Redundancy in the location of $\mathcal{O}_i$ is removed by constraining it to lie in the plane perpendicular to the nominal joint axis through the nominal joint origin.

C. Circular-Point Analysis (CPA)

First consider a simple and intuitive POE parameter identification method circular-point analysis (CPA) [26]. Move the robot to fixed location corresponding to the joint angle q. Then command the ith joint to K different values and record the corresponding TCP positions:

$$p_{0T}^{(k)} = p_{0i} + e^{(h_i)_o^\times q_i^{(k)}} (p_{iT})_o, \tag{10}$$

with $k = 1, \dots, K$. Fit these positions, $\{p_{0T}^{(k)}\}$, to a circle using a standard 3D circle fitting algorithm [27], [28] to find $(h_i)_o$ and p'_{oi} (normal and center of the circle). The identified POE parameters may then be extracted using (9). The CPA method is conceptually simple but each joint must move to multiple locations for a given q. For multiple configurations, the process could be time consuming.

D. Nonlinear Least Square (NLS) Estimate

An alternate approach is to estimate $(\mathbf{P}, \mathbf{H})$ by solving a nonlinear least square (NLS) problem. From Section II-B, we can parameterize $p_{i-1,i}$ and h_i by two parameters each:

$$h_i = \mathsf{R}(k_{1,i}, \theta_i)\mathsf{R}(k_{2,i}, \phi_i)\bar{h}_i \tag{11a}$$
$$p_{i-1,i} = \bar{p}_{i-1,i} + (v_i k_{1,i} + w_i k_{2,i}) - (v_{i-1} k_{1,i-1} + w_{i-1} k_{2,i-1}) \tag{11b}$$

where $(k_{1,i}, k_{2,i})$ are two orthogonal vectors in $\bar{\pi}_i$, and (θ_i, ϕ_i) parametrize h_i relative to $\bar{h}_i$ and (v_i, w_i) parameterizes the location of $\mathcal{O}_i$ relative to $\bar{\mathcal{O}}_i$. Note that since the base and tool origins are fixed, $v_0 = w_0 = 0$ and $v_{n+1} = w_{n+1} = 0$. Define

$$\begin{aligned} \Theta_{\mathbf{P}} &= [v_1, w_1, \dots, v_n, w_n]^\top \\ \Theta_{\mathbf{H}} &= [\theta_1, \phi_1 \dots, \theta_n, \phi_n]^\top \\ \Theta_{\mathbf{P},\mathbf{H}} &= \begin{bmatrix} \Theta_{\mathbf{P}} \\ \Theta_{\mathbf{H}} \end{bmatrix}. \end{aligned} \tag{12}$$

There are $4n$ total parameters. The metric for the NLS is

$$\min_{\Theta_{\mathbf{P},\mathbf{H}}} \frac{1}{2} \sum_{k=1}^{K} \left\| T_{0T}\left(q^{(k)}, \mathbf{P}(\Theta_{\mathbf{P}}), \mathbf{H}(\Theta_{\mathbf{H}})\right) - T_{0T}^{*(k)} \right\|^2. \tag{13}$$

where K is the total collected data samples, $T_{0T}^{*(k)}$ is the ground truth measurement at sample k. For the error norm, we choose a weighted combination of the position error and the orientation error:

$$\|T_1 - T_2\|^2 = \|p_1 - p_2\|^2 + a \left\| R_1 R_2^\top - I_3 \right\|_F^2 \tag{14}$$

where a is a weighting constant and the orientation error norm is equivalent to $8\sin^2(\frac{\theta}{2})$, θ is the equivalent angle of $R_1 R_2^\top$. We apply the gradient descent algorithm, starting with the nominal POE values, i.e., $\Theta_P = \Theta_H = 0$. The detail of the gradient computation is presented in the Appendix I.For a given configuration q, we collect a set of joint angles and task frame data near q and then estimate the POE parameters for that local data cluster.

III. Configuration Dependent Calibration (CDC)

The NLS method finds a set of POE parameters, $\Theta_{\mathbf{P},\mathbf{H}}$, around a given configuration using nearby data points. We may repeat the process throughout the robot workspace to obtain multiple sets of POE parameters, $\Theta_{\mathbf{P},\mathbf{H}}^{(\ell)}$, $\ell = 1, \dots, L$, corresponding to the centroid of the cluster $q^{(i)}$. To find the POE parameters at an arbitrary configuration q, we may apply an interpolation method (e.g., nearest neighbor, linear interpolation, cubic interpolation, etc.) based on the identified cluster parameters.

For elbow type of manipulators, $\Theta_{\mathbf{P},\mathbf{H}}^{(\ell)}$ shows a strong dependence of the shoulder and elbow angles, (q_2, q_3) due to the moment arm effect. Note that if we have longer extension from the robot wrist, there will be increasing dependence on the wrist angles. In this case, we may write the POE parameters as

$$\Theta_{\mathbf{P},\mathbf{H}}(q) = f_{\Theta_{\mathbf{P},\mathbf{H}}}(q_2, q_3). \tag{15}$$

We will first consider a basis function expansion approach to approximate $f_{\Theta_{\mathbf{P},\mathbf{H}}}$ as a linear combination of a chosen basis set, and use the estimated configuration-dependent kinematic parameters $(q^{(\ell)}, \Theta^{(\ell)}_{\mathbf{P},\mathbf{H}})$ to learn the expansion coefficients. We will also consider a direct blackbox approximation using a multilayer neural network, and train using the identified POE parameters.

A. Basis Function Approximation

Write the parameter vector $f_{\Theta_{\mathbf{P},\mathbf{H}}}$ (15) as a linear combination of m basis functions, collected in a column vector $\beta(q_2, q_3) \in \mathbb{R}^m$:

$$f_{\Theta_{\mathbf{P},\mathbf{H}}}(q_2, q_3) = A\beta(q_2, q_3) \tag{16}$$

where A is a constant coefficient matrix. Using the identified POE parameters from L clusters, we have

$$\underbrace{\begin{bmatrix}\Theta_{\mathbf{P},\mathbf{H}}(q^{(1)}) & \dots & \Theta_{\mathbf{P},\mathbf{H}}(q^{(L)})\end{bmatrix}}_{\mathbf{C}} = A\underbrace{\begin{bmatrix}\beta(q_2^{(1)}, q_3^{(1)}) & \dots & \beta(q_2^{(L)}, q_3^{(L)})\end{bmatrix}}_{\mathbf{B}}. \tag{17}$$

The least square estimate of A is:

$$A = \mathbf{C}\mathbf{B}^+ \tag{18}$$

where $\mathbf{B}^+$ is the Moore-Penrose pseudo-inverse of $\mathbf{B}$. Then given any q, we can compute $\Theta_{\mathbf{P},\mathbf{H}}$ using (16) and (18). One may use a generic basis set such as the radial basis functions (RBF) [29]. For an elbow-type manipulator, the configuration dependence is likely due to the gravity load on the elbow and shoulder joints. In this case, a reasonable basis is the Fourier basis based on the expansion of the gravity load [25] :

$$\beta(q_2, q_3) = \Big[1, s_2, c_2, s_3, c_3, s_{23}, c_{23}, \sin(2q_2), \cos(2q_2), \sin(2q_3), \cos(2q_3), \sin(2(q_2+q_3)), \cos(2(q_2+q_3))\Big]^\top \tag{19}$$

where $s_i = \sin(q_i)$, $c_i = \cos(q_i)$, $s_{ij} = \sin(q_i + q_j)$, $c_{ij} = \cos(q_i + q_j)$, and $\beta \in \mathbb{R}^{13}$.

To further reduce the dimension of the basis function, we perform singular value decomposition on the coefficient matrix, $A = U\Sigma V^\top$, and retain r ($r < m$) dominant singular values. We then have a reduced basis set:

$$\beta_r(q_2, q_3) := \begin{bmatrix} V_1^\top \\ \vdots \\ V_r^\top \end{bmatrix} \beta(q_2, q_3). \tag{20}$$

The A matrix may be replaced by a smaller matrix

$$A_r := \begin{bmatrix}\sigma_1 U_1 & \dots & \sigma_r U_r\end{bmatrix}. \tag{21}$$

The POE parameters in (15) may be approximated by

$$\Theta_{\mathbf{P},\mathbf{H}} = A_r \beta_r(q_2, q_3). \tag{22}$$

B. Low Dimension Representation with Autoencoder

Given a basis set β, there may be a lower dimensional basis that more compactly represent the parameter function $f_{\Theta_{\mathbf{P},\mathbf{H}}}$. The singular value decomposition approach in (20) finds a lower dimensional hyperplane where the training data are concentrated. A more general approach, without the linearity assumption inherent in SVD, is to apply the Autoencoder (AE) method. The encoder transforms the parameters $\Theta_{\mathbf{P},\mathbf{H}}$ to a low dimension latent vector $z \in \mathbb{R}^N$ as the representation of the original high dimensional data. The decoder lifts z to an approximate $\Theta_{\mathbf{P},\mathbf{H}}$. Denote the encoder as f_{en}, decoder as f_{de} and the decoded parameter as $\hat{\Theta}_{\mathbf{P},\mathbf{H}}$:

$$z = f_{en}(\Theta_{\mathbf{P},\mathbf{H}}), \quad \hat{\Theta}_{\mathbf{P},\mathbf{H}} = f_{de}(z). \tag{23}$$

The encoder and decoder are parametrized using neural networks. The encoder and decoder have similar but flipped network structure, visualized in Fig. 4. These networks are trained based on the loss function

$$\mathcal{L} = \frac{1}{L}\sum_{i=1}^{L}(\Theta^{(i)}_{\mathbf{P},\mathbf{H}} - f_{de}(f_{en}(\Theta^{(i)}_{\mathbf{P},\mathbf{H}})))^2. \tag{24}$$

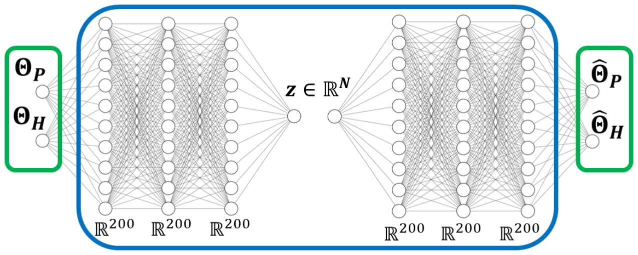


Fig. 4: Autoencoder (AE) structure. The encoder has the parameters $\Theta_{\mathbf{P},\mathbf{H}}$ as the input and latent vector z as the output with the decoder vice versa. They both has three hidden layers with 200 nodes and ReLU as the activation function.

C. Neural Network Function Approximation

We may use a multilayer neural network to directly approximate $f_{\Theta_{\mathbf{P},\mathbf{H}}}(q_2, q_3)$ in (15). The structure of the NN is showed in Fig. 5. The input layer has two nodes, i.e. the joint 2 and 3 angles. The hidden layers are fully connected layers with ReLU activation functions. The output layers is a fully connected layer with no output function, which become the linear combination of the hidden layers output.

The loss function is the mean square error between the predicted parameters and the calibrated ones.

$$\mathcal{L} = \frac{1}{L}\sum_{i=1}^{L}(\hat{\Theta}_{\mathbf{P},\mathbf{H}}(q^{(i)}) - \Theta_{\mathbf{P},\mathbf{H}}(q^{(i)}))^2 \tag{25}$$

where $\hat{\Theta}_{\mathbf{P},\mathbf{H}}(q^{(i)})$ is the predicted output from the NN. We use the Adam optimizer [30] to update the parameters based on the loss of each learning epoch with adaptive learning rate.

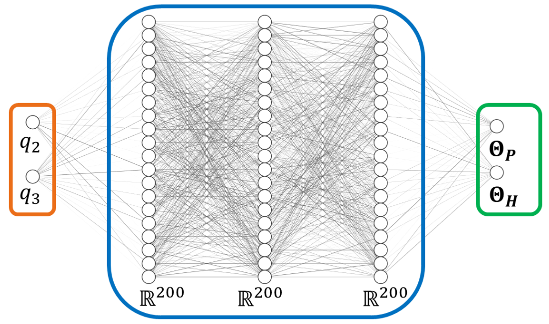
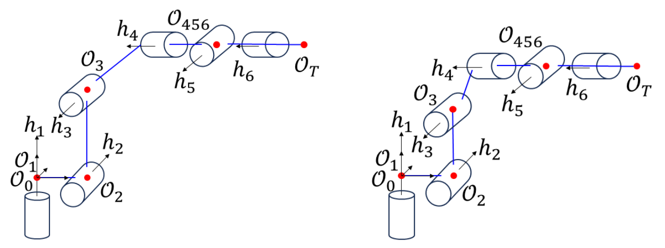


Fig. 5: Neural Network (NN) structure. The learned NN approximating the function $f_{\Theta_{\mathbf{P},\mathbf{H}}}(q_2, q_3)$ has one input layer (orange) taking the input of joint 2 q_2 and 3 q_3, three hidden layers (blue) with 200 nodes each and one output layer (green) predicting the error parameters $\Theta_{\mathbf{P},\mathbf{H}}$.

IV. Implementation and Evaluation

A. Experiment Setup

For the experimental verification of the calibration methods, we used Yaskawa MA2010 and MA1440 robots from our welding testbed. Both robots are the classic 6 rotational-axis (6R) industrial robot elbow arm with spherical wrists (joints 4, 5, and 6 have intersecting rotation axes). The vendor-provided POE model parameters and the zero configurations of these robots are shown in Fig. 6.



$$\mathbf{P} = \begin{bmatrix} 0 & 150 & 0 & 1082 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 760 & 200 & 0 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0 & 155 & 0 & 640 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 614 & 200 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) MA2010 (R1) (b) MA1440 (R2)

Fig. 6: Robot schematics in the zero configuration.

To measure the tool pose, T_{0T}, we use the OptiTrack motion capture system, with eight Prime^{x}13 [31] and the software Motive [32]. This system operates by detecting retro-reflective markers using IR-emitting cameras to estimate the distance to each marker. The stated accuracy for the marker distance is 0.2 mm. Using Small-Angle approximation, the orientation accuracy is approximately $\theta \approx \frac{0.2}{L}$, where L is the distance between the markers. In our setup, five markers are attached to the end effector of each robot. The orientation accuracy is around 0.05 degree. The 2D camera resolution is about 0.02-0.03 mm on a 2D plane enable by the sub-pixel centroiding technology. The setup of the robot and the motion tracking system is shown in Fig. 1. The motion tracking system has a total of 8 cameras. We collect data only after the system is properly calibrated and has reached thermal stability to ensure optimal accuracy.

For each robot, we collect a training set of tool frame data, T_{0T}, and the corresponding joint angles, q, and a test set for evaluation of the calibration outside of the training set. For MA2010, we specify 248 configurations with $q_2 \in [-55°, 50°]$ and $q_3 \in [-70°, 50°]$ as anchors for the parameter identification clusters. Around each anchor configuration, we collect 7 sets of tool frame data for a total of 1736 tool frame data points in 248 clusters in the training set. For MA1440, the training data consists of 280 configurations with $q_2 \in [-55°, 50°]$ and $q_3 \in [-70°, 60°]$ as cluster anchors for a total of 1960 tool frame data points in 280 clusters in the training set.

B. Evaluation

We apply the NLS and CDC methods on the training data to find the kinematics model and evaluate the result on the same training dataset. The NLS method finds one set of $(\mathbf{P}, \mathbf{H})$ to minimize all the tool errors. The CDC method finds one set of $(\mathbf{P}, \mathbf{H})$ for each cluster. We include an additional calibration case in our evaluation: a base-only calibration (Base), where only the base position (3 parameters) is optimized. This scenario corresponds to redefining the reference frame to eliminate any fixed base offset. In our manufacturing application of cold spraying, the performance requirement is less than 1 mm in position error and 1 degree in the surface normal error. To balance these two requirements in the objective function (14), we set the weighting factor to $a = 1/(8\sin^2(\frac{\pi}{360})) = 1641$. The error statistics for the training set is summarized in Table I. Both NLS and CPA improves on the vendor parameters, but CPA only works well on certain configurations and thus has a large standard deviation. Configuration dependency in CDC allows further reduction of the tool frame error over the training set. The base-only calibration reduces the mean position error but does not achieve the same level of improvement as NLS or CDC. Moreover, the orientation error remains unchanged, indicating that the error is not simply due to a fixed offset.

	R1 Position Error (mm)			R2 Position Error (mm)		
Method	Mean	Std	Max	Mean	Std	Max
Nominal	1.32	0.23	1.81	1.77	0.94	4.18
Base	0.41	0.20	1.24	1.59	0.89	3.49
CPA	0.49	0.25	1.20	1.59	0.79	3.29
NLS	0.25	0.14	0.86	0.41	0.28	1.53
CDC	**0.10**	**0.05**	**0.46**	**0.04**	**0.02**	**0.12**

	R1 Orientation Error (deg)			R2 Orientation Error (deg)		
Method	Mean	Std	Max	Mean	Std	Max
Nominal	0.07	0.02	0.12	0.12	0.04	0.26
Base	0.07	0.02	0.12	0.12	0.04	0.26
CPA	0.05	0.02	0.13	0.13	0.05	0.28
NLS	0.05	0.02	0.11	0.10	0.04	0.27
CDC	**0.04**	**0.02**	**0.10**	**0.09**	**0.05**	**0.25**

TABLE I: Tool frame position and orientation error statistics comparison on the training data. Boldface font highlights the best performance among all methods.

To evaluate the performance of the identified POE models beyond the training set, we use a testing dataset consisting of 1,500 robot poses, uniformly sampled from each robot's workspace. The testing dataset is representative of the entire workspace, as shown by the joint angle distribution in Fig. 7. The error from applying the nominal POE model to the testing dataset confirms the strong correlation between the position error in the clusters and the shoulder and elbow angles, as shown in Table II.

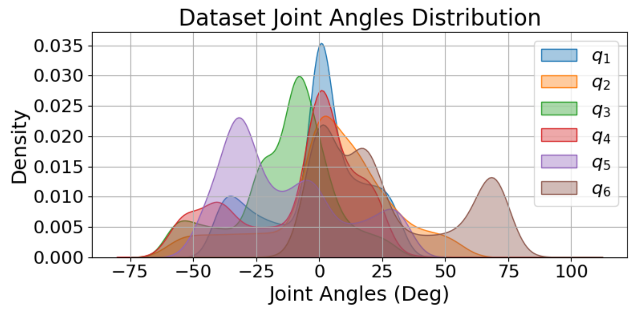


Fig. 7: Joint angle distribution in the testing dataset shows broad coverage of all joint angles.

	q_1	q_2	q_3	q_4	q_5	q_6
MA2010 (R1)	0.16	**0.83**	**0.82**	0.30	0.61	0.31
MA1440 (R2)	0.38	**0.77**	**0.44**	0.31	0.35	0.07

TABLE II: Correlation of the position errors with the joint angles show the strongest correlation with the shoulder and elbow joint angles (q_2, q_3).

We consider the following POE parameter identification methods for tool pose error comparison on the testing dataset:

- Non-Configuration-Dependent (one set of $(\mathbf{P}, \mathbf{H})$ parameters for all configurations)
  - Nominal $(\mathbf{P}, \mathbf{H})$: Vendor-provided parameters.
  - Base: Optimization of the robot base frame position, corresponding to a 3-parameter kinematic calibration.
  - CPA: Circular point analysis method (Secton II-C) around the zero configuration, $q = 0$.
  - NLS-0: Nonlinear least square method (Section II-D) around the zero configuration, $q = 0$.
  - NLS-1: Nonlinear least square method based on all training data.
- Configuration-Dependent Kinematics based on the interpolation from the NLS fit for each training cluster.
  - Nearest Neighbor: Nearest cluster.
  - Linear Interpolation: Linear interpolation based on nearby clusters.
  - Cubic Interpolation: Cubic spline interpolation based on nearby clusters.
  - RBF Interpolation: Radial basis function based on the training data (Section III-A)
  - FBF Interpolation: Fourier basis function based on the training data (Section III-A) The basis dimension $N = 13$ and $N = 7$ (reduced basis).
  - NN Interpolation: Neural Network approximation based on the training data (Section III-C)
  - AE Interpolation: Linear interpolation of the latent space vectors and decoding to $\Theta_{\mathbf{P},\mathbf{H}}$ (Section III-B). Latent space dimension $N = 6$.

For FBF, NN and AE, the number of parameters for each method is given in Table III.

Robots	FBF N=13	FBF N=7	NN	AE N=6
R1	312	168	87633	88420
R2	312	168	87633	88420

TABLE III: Number of parameters used in each method. For AE, only the number of parameters in the decoder, which used for estimation, is reported.

	R1 Position Error (mm)			R2 Position Error (mm)		
Method	Mean	Std	Max	Mean	Std	Max
Nominal	1.29	0.21	1.72	1.08	0.81	3.07
Base	0.53	0.25	1.27	1.17	0.62	2.93
CPA	0.44	0.20	0.99	1.44	0.61	3.19
NLS-0	0.41	0.21	0.99	1.68	0.71	3.20
NLS-1	0.92	0.79	3.46	1.71	1.20	5.18
Nearest	0.32	0.17	0.81	0.46	0.29	1.29
Linear	0.31	0.17	0.81	0.42	0.26	1.23
Cubic	0.31	0.17	0.83	0.43	0.27	1.33
RBF	0.31	0.17	0.83	0.47	0.44	4.90
FBF N=13	0.31	0.14	0.74	0.42	0.21	0.99
FBF N=7	0.32	0.16	0.81	0.44	0.19	0.98
NN	0.34	**0.13**	0.78	0.43	**0.15**	**0.77**
AE N=6	**0.31**	0.14	**0.73**	**0.42**	0.19	0.94
	R1 Orientation Error (deg)			R2 Orientation Error (deg)		
Method	Mean	Std	Max	Mean	Std	Max
Nominal	0.10	0.04	0.20	0.11	0.04	0.23
Base	0.10	0.04	0.20	0.11	0.04	0.23
CPA	**0.08**	**0.03**	0.17	0.11	0.04	0.20
NLS-0	0.08	0.03	**0.17**	0.11	0.05	0.22
NLS-1	0.19	0.09	0.39	1.00	0.58	2.24
Nearest	0.09	0.04	0.19	0.09	0.04	0.19
Linear	0.09	0.04	0.19	0.09	0.04	0.19
Cubic	0.09	0.04	0.19	0.09	0.04	0.19
RBF	0.09	0.04	0.19	0.09	0.04	0.31
FBF N=13	0.09	0.04	0.18	**0.09**	0.04	0.20
FBF N=7	0.09	0.04	0.18	0.09	**0.04**	**0.19**
NN	0.10	0.04	0.19	0.09	0.04	0.20
AE N=6	0.09	0.04	0.19	0.09	0.04	0.21

TABLE IV: Tool frame position and orientation error of multiple calibration methods applied to the test data. Boldface font highlights the best performance among all methods.

At each configuration in the testing dataset, we compare the computed tool pose using the identified POE parameters with the measured tool pose. Fig. 8a-8b show the tool frame position error of the methods applied to the two robots in each robot's base frame. Table IV summarizes the statistics of the tool frame position error for the different parameter identification methods. Paired t-tests and Wilcoxon signed-rank tests comparing configuration-dependent and non-configuration-dependent methods all show p-values near zero, indicating statistically significant improvements. From these results, we make the following observations:

- All calibration methods, except for NLS with fixed POE parameters, tend to improve on the nominal vendor-provided parameters. This highlights the importance of calibration.
- For R1, FBF and NN methods do not improve much over standard interpolation methods. For R2, the difference is

more pronounced. This may be explained by examining the variation of locally identified POE parameters $\Theta_{\mathbf{P},\mathbf{H}}$ from the nominal values over the training dataset, as shown in Table V. The higher standard deviation in R2 indicates that it has much more variations across the configurations than R1. This implies that standard interpolation methods would generalize poorly than FBF and NN methods.

	$\Theta_{\mathbf{P}}$ (mm)		$\Theta_{\mathbf{H}}$ (deg)	
Robots	Mean	Std	Mean	Std
R1	0.6653	0.0412	0.0353	0.0131
R2	0.6070	0.1392	0.1131	0.0429

TABLE V: Statistics of the deviation locally identified POE parameters from the nominal POE parameters over the training dataset. R2 shows much higher POE parameter variation between configuration.

- When the variation of $\Theta_{\mathbf{P},\mathbf{H}}$ is small between clusters (i.e., low configuration dependence), all interpolation methods perform reasonably well (as in the R1 case in Table IV). When the variation is large, interpolation using basis functions performs significantly better (as in the R2 case in Table IV).
- CPA works the best near the configuration where the process is applied. It also works reasonably well in other configurations but with degraded performance away from the nominal configuration (the zero configuration in our case).
- The NLS method with fixed POE parameters, whether using data around the zero configuration or all data from the workspace, works well on the training set (as in Table I) but does not generalize well away from the training set.
- CDC methods work well at the training data but the interpolation method affects its performance at the test configurations. This is most evident for the RBF interpolation method where large error could result between the training points.
- With the universal function approximation capability, the NN interpolation achieves the best accuracy among all interpolation methods, though the training process is time consuming.
- The Fourier basis interpolation motivated from the gravity load works comparably to NN interpolation. The chosen basis function has likely captured the functional dependence of the kinematics on configurations. Using only the dominant basis (7), we can reduce the number of basis functions by almost half, without sacrificing much performance. In addition, FBF requires the fewest parameters to approximate the kinematic model, resulting in greater training efficiency.
- The autoencoder method reduces the dimension of the basis vectors (latent space) and still achieves significant improvement of the tool pose prediction error. The dimension of the latent space (6) is about the same as the same as the dimension of the reduced Fourier basis (7).

b) **Fourier Basis Reduction** For MA2010 (Robot 1), Fig. 9 shows the SVD results of A. The left plot shows the log-scale of the singular values of A. Approximately seven singular values appear to be significant. The right plot shows a heat map of the magnitude of the V^T matrix. The first few rows contain the constant function basis $\mathbf{1}$, as well as $\cos(q_2)$ and $\cos(q_3)$ with larger values. This suggests that these basis functions have a more significant influence on the configuration-dependent kinematic models. As shown in Table IV, the performance of the reduced basis is comparable to using the full basis.

c) **Autoencoder Implementation** We implement both the encoder and decoder with three hidden layers. Each layer has 200 nodes and ReLU as the activation function with no output function, i.e. there's no activation function at the output layer and the last layer is a linear function. For training, the loss function is the reconstruction mean square error. Given a configuration q, we linearly interpolate from the identified POE to obtain the latent vector z and then decode z to obtain the parameter $\Theta_{\mathbf{P},\mathbf{H}}$. Table VI shows that with dimension $N = 6$, the decoded parameters have the best accuracy. Further examination of the structure of the latent space indicates a strong relationship between the latent vector distribution and joint 2 and 3.

N	2	6	12
Mean	0.33	0.34	0.32
Max	0.98	0.86	0.91

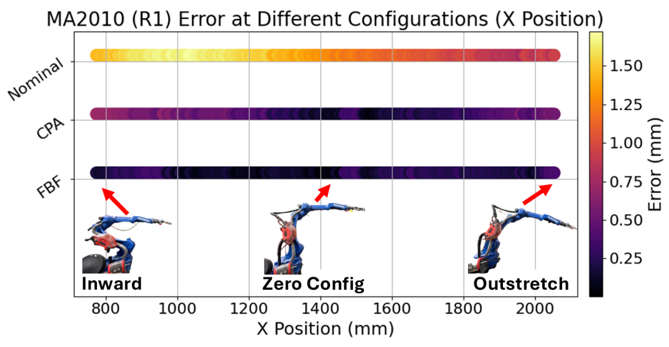
TABLE VI: Mean and max testing position error (mm) of different latent space dimension N.

d) **Neural Network Approximation** In [11], the authors used neural networks with 2, 4, or 6 hidden layers. To balance model capacity and overfitting, we experimented with various hidden layer sizes and depths to ensure sufficient expressiveness without excessive parameterization. Table VII shows the mean and max testing error of different hidden layers sizes and depths. We choose 3 hidden layers with 200 nodes each which yields the best results.

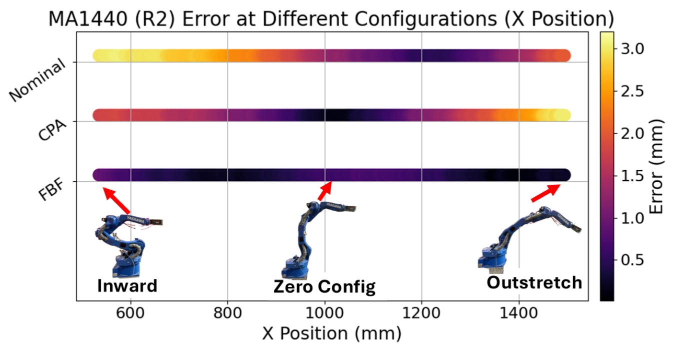
	[100,100]	[200,200]	[600,600]	[200,200,200]
Mean	0.34	0.27	0.27	0.25
Max	0.91	0.71	0.67	0.65

TABLE VII: Mean and max testing position error (mm) of different hidden layer structures.

e) **Dual-Robot Validation** For performance comparison without using the motion capture system, we command both robots to have a fixed relative tool position while changing the robot poses from inward to outstretched. This setup, shown in Fig. 10, simulates a dual-robot collaborative manufacturing task, such as R1 applying a cold spray tool to a workpiece held by R2. By recording the joint angles of both robots and applying different kinematic parameter sets, we compute the resulting position deviations from the known physical setup. Table VIII summarizes the mean, standard deviation, and maximum position errors for the nominal, CPA, FBF and AE models for five configurations. The result is consistent with performance evaluation using the motion capture system and confirms that the proposed

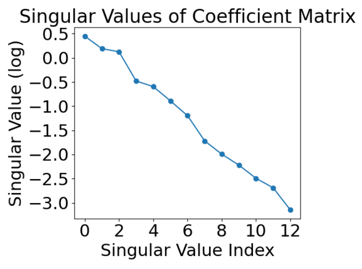


(a) MA2010 testing dataset position error.

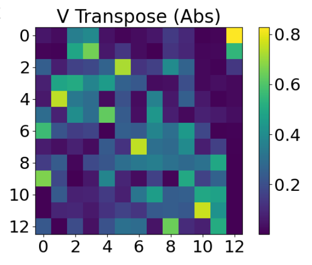


(b) MA1440 testing dataset position error.

Fig. 8: R1 (MA2010), R2 (MA1440) testing dataset position error. The X-Position represents the tool approach direction in the robot base frame, which corresponds to different robot configurations: from inward (left), through zero configuration (center), to outstretched (right). The color bars correspond to the three POE parameter sets. The colormap indicates the error magnitude in millimeters. For the nominal parameters and circular point analysis (CPA) method, the tool pose error varies with configuration. CPA, which is calibrated at the zero configuration, shows the lowest error near its calibration pose, but larger error at other configurations. Configuration-dependent methods such as the Fourier basis function (FBF) method maintains consistently low error across the entire workspace. R2 also shows higher color variation across configurations, leading to poor performance of standard interpolation methods.



(a) Singular value of Matrix A



(b) Matrix V^T

Fig. 9: Coefficient analyzation of the basis function through SVD. On the left is the singular value of matrix A in log scale. On the right is the matrix V^T.

CDC framework is effective for maintaining high relative positioning accuracy in a real dual-robot setup.

	Mean (mm)	Std (mm)	Max (mm)
Nominal	1.40	0.52	2.08
CPA	1.45	0.62	2.23
FBF N=7	0.52	0.21	0.81
AE N=6	0.61	0.16	0.88

TABLE VIII: Mean, standard deviation and maximum position error of the dual-robot validation. The result is consistent with the performance evaluation using the motion capture system.

V. Conclusion and Future Work

The paper introduces a configuration-dependent kinematic model and calibration framework designed to compensate for the inherent variability in serial robot kinematics due to non-geometric factors as a function of the robot pose. The robot kinematics are modeled using a minimal version of the Product of Exponentials parameters. Calibration is formulated as a non-linear least squares optimization, aimed at minimizing position and orientation errors across different collected configuration data clusters using a Jacobian iterative approach. We express the robot kinematics as a function of joint 2 and 3 angles and approximate the function using optimized kinematic parameters at sampled configurations. The results demonstrate that the accuracy of the configuration-dependent kinematic model surpasses that of a single universal kinematic model calibrated at one configuration. We are currently working on the calibration of two robot manipulators concurrently.

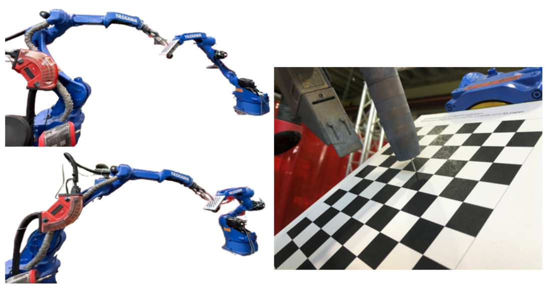


Fig. 10: To evaluate performance without using the motion capture system, two robots are jogged to five different configurations, ranging from inward to outstretched, while maintaining contact at the same physical point.

Appendix I
Gradient Computation

First consider the rotation matrix,

$$R_{0T} = R_{01} \dots R_{n-1,n} R_{nT} \tag{26}$$

where $R_{i-1,i} = e^{h_i^\times q_i}$ and R_{nT} is a constant matrix. From the Euler-Rodrigues Formula, we can write $R_{i-1,i}$ as

$$R_{i-1,i} = I + \sin(q_i)(h_i)^\times + (1 - \cos(q_i)) h_i^\times h_i^\times \tag{27}$$

It follows that

$$\frac{\partial R_{i-1,i}}{\partial \theta_i} = \sin(q_i)\left(\frac{dh_i}{d\theta_i}\right)^{\times} + (1-\cos(q_i))\left(\left(\frac{dh_i}{d\theta_i}\right)^{\times}(h_i)^{\times} + (h_i)^{\times}\left(\frac{dh_i}{d\theta_i}\right)^{\times}\right) \tag{28a}$$

$$\frac{\partial R_{i-1,i}}{\partial \phi_i} = \sin(q_i)\left(\frac{dh_i}{d\phi_i}\right)^{\times} + (1-\cos(q_i))\left(\left(\frac{dh_i}{d\phi_i}\right)^{\times}(h_i)^{\times} + (h_i)^{\times}\left(\frac{dh_i}{d\phi_i}\right)^{\times}\right) \tag{28b}$$

From (11a), we have

$$\frac{dh_i}{d\theta_i} = k_{1,i}^{\times}\mathsf{R}(k_{1,i},\theta_i)\mathsf{R}(k_{2,i},\phi_i)\bar{h}_i \tag{29a}$$

$$\frac{dh_i}{d\phi_i} = \mathsf{R}(k_{1,i},\theta_i)k_{2,i}^{\times}\mathsf{R}(k_{2,i},\phi_i)\bar{h}_i. \tag{29b}$$

Hence, we can compute the gradient of R_{0T} by (θ_i, ϕ_i):

$$\frac{\partial R_{0T}}{\partial \theta_i} = R_{0,i-1}\frac{\partial R_{i-1,i}}{\partial \theta_i}R_{i,T},\ \frac{\partial R_{0T}}{\partial \phi_i} = R_{0,i-1}\frac{\partial R_{i-1,i}}{\partial \phi_i}R_{i,T}.$$

The position kinematics is given by

$$p_{0T} = p_{01} + R_{01}p_{12} + ... + R_{0n}p_{nT}. \tag{30}$$

From (11b), we have

$$\frac{dp_{i-1,i}}{dv_i} = k_{1,i}, \quad \frac{dp_{i-1,i}}{dw_i} = k_{2,i} \tag{31a}$$

$$\frac{dp_{i,i+1}}{dv_i} = -k_{1,i}, \quad \frac{dp_{i,i+1}}{dw_i} = -k_{2,i}. \tag{31b}$$

It follows

$$\frac{dp_{0T}}{dv_i} = (R_{0,i-1} - R_{0,i})k_{1,i},\ \frac{dp_{0T}}{dw_i} = (R_{0,i-1} - R_{0,i})k_{2,i}$$

$$\frac{dp_{0T}}{d\theta_i} = R_{0,i-1}\frac{dR_{i-1,i}}{d\theta_i}p_{iT},\ \frac{dp_{0T}}{d\phi_i} = R_{0,i-1}\frac{dR_{i-1,i}}{d\phi_i}p_{iT}.$$

where $p_{iT} = p_{i,i+1} + \ldots + R_{in}p_{nT}$. Using the definition of the parameter vectors in (12), we have

$$\begin{aligned}\frac{\partial p_{0T}}{\partial \Theta_{\mathbf{P}}} &= \begin{bmatrix} \frac{dp_{0T}}{dv_1} & \frac{dp_{0T}}{dw_1} & \cdots & \frac{dp_{0T}}{dv_n} & \frac{dp_{0T}}{dw_n}\end{bmatrix}\\ \frac{\partial p_{0T}}{\partial \Theta_{\mathbf{H}}} &= \begin{bmatrix} \frac{dp_{0T}}{d\theta_1} & \frac{dp_{0T}}{d\phi_1} & \cdots & \frac{dp_{0T}}{d\theta_n} & \frac{dp_{0T}}{d\phi_n}\end{bmatrix}.\end{aligned} \tag{32}$$

For the orientation, let ψ be any 3-parameter representation of $SO(3)$ and J_ψ be the representation Jacobian, i.e., $\dot{\psi} = J_\psi \omega$. Then the gradient of ψ with respect to the parameter vectors is given by

$$\frac{\partial \psi}{\partial \Theta_{\mathbf{H}}} = J_\psi\big[(\frac{dR_{0T}}{d\theta_1}R_{0T}^{\top})^{\vee}, \ldots, (\frac{dR_{0T}}{d\phi_n}R_{0T}^{\top})^{\vee}\big] \tag{33}$$

and $\frac{\partial \psi}{\partial \Theta_{\mathbf{P}}} = 0$, where $(\cdot)^{\vee}$ converts a skew symmetric matrix to a vector (inverse of $(\cdot)^{\times}$).

References

[1] H. Chen, T. Fuhlbrigge, and X. Li, "Automated industrial robot path planning for spray painting process: A review," in *2008 IEEE International Conference on Automation Science and Engineering*, 2008, pp. 522–527.

[2] S. Chen, Z. Wang, A. Chakraborty, M. Klecka, G. Saunders, and J. Wen, "Robotic deep rolling with iterative learning motion and force control," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5581–5588, 2020.

[3] D. Zhu, X. Feng, X. Xu, Z. Yang, W. Li, S. Yan, and H. Ding, "Robotic grinding of complex components: A step towards efficient and intelligent machining – challenges, solutions, and applications," *Robotics and Computer-Integrated Manufacturing*, vol. 65, 2020.

[4] S. Kana, J. Gurnani, V. Ramanathan, S. H. Turlapati, M. Z. Ariffin, and D. Campolo, "Fast kinematic re-calibration for industrial robot arms," *Sensors*, vol. 22, no. 6, 2022.

[5] Z. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 377–385, 1987.

[6] S. A. Hayati, "Robot arm geometric link parameter estimation," in *The 22nd IEEE Conf. on Decision and Control*, 1983, pp. 1477–1483.

[7] K. Okamura and F. Park, "Kinematic calibration using the product of exponentials formula," *Robotica*, vol. 14, no. 4, p. 415–421, 1996.

[8] H. Zhuang, L. K. Wang, and Z. S. Roth, "Error-model-based robot calibration using a modified cpc model," *Robotics and Computer-Integrated Manufacturing*, vol. 10, no. 4, pp. 287–299, 1993.

[9] S. Hayati and M. D. Mirmirani, "Improving the absolute positioning accuracy of robot manipulators," *J. Field Robotics*, vol. 2, pp. 397–413, 1985.

[10] W. Veitschegger and C.-H. Wu, "Robot accuracy analysis based on kinematics," *IEEE Journal on Robotics and Automation*, vol. 2, no. 3, pp. 171–179, 1986.

[11] J.-C. Hsiao, K. Shivam, I.-F. Lu, and T.-Y. Kam, "Positioning accuracy improvement of industrial robots considering configuration and payload effects via a hybrid calibration approach," *IEEE Access*, vol. 8, pp. 228 992–229 005, 2020.

[12] J. H. Jang, S. H. Kim, and Y. K. Kwak, "Calibration of geometric and non-geometric errors of an industrial robot," *Robotica*, vol. 19, no. 3, p. 311–321, 2001.

[13] X. Chen and Q. Zhan, "The kinematic calibration of an industrial robot with an improved beetle swarm optimization algorithm," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4694–4701, 2022.

[14] F. C. Park and K. Okamura, *Kinematic Calibration and the Product of Exponentials Formula*. Dordrecht: Springer Netherlands, 1994, pp. 119–128.

[15] J. He, L. Gu, G. Yang, Y. Feng, S. Chen, and Z. Fang, "A local poe-based self-calibration method using position and distance constraints for collaborative robots," *Robotics and Computer-Integrated Manufacturing*, vol. 86, p. 102685, 2024.

[16] C. Li, Y. Wu, H. Löwe, and Z. Li, "Poe-based robot kinematic calibration using axis configuration space and the adjoint error model," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1264–1279, 2016.

[17] J. Zhou, H.-N. Nguyen, and H.-J. Kang, "Simultaneous identification of joint compliance and kinematic parameters of industrial robots," *International Journal of Precision Engineering and Manufacturing*, vol. 15, pp. 2257–2264, 11 2014.

[18] K. Kamali, A. Joubair, I. A. Bonev, and P. Bigras, "Elasto-geometrical calibration of an industrial robot under multidirectional external loads using a laser tracker," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4320–4327.

[19] Y. Wang, Z. Chen, H. Zu, X. Zhang, C. Mao, and Z. Wang, "Improvement of heavy load robot positioning accuracy by combining a model-based identification for geometric parameters and an optimized neural network for the compensation of nongeometric errors," *Complexity*, vol. 2020, pp. 1–13, 01 2020.

[20] C. Landgraf, K. Ernst, G. Schleth, M. Fabritius, and M. F. Huber, "A hybrid neural network approach for increasing the absolute accuracy of industrial robots," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 468–474.

[21] W. Zhu, G. Li, H. Dong, and Y. Ke, "Positioning error compensation on two-dimensional manifold for robotic machining," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 394–405, 2019.

[22] G. Zhao, P. Zhang, G. Ma, and W. Xiao, "System identification of the nonlinear residual errors of an industrial robot using massive measurements," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 104–114, 2019.

[23] H.-N. Nguyen, P.-N. Le, and H.-J. Kang, "A new calibration method for enhancing robot position accuracy by combining a robot model–based identification approach and an artificial neural network–based error compensation technique," *Advances in Mechanical Engineering*, vol. 11, no. 1, p. 1687814018822935, 2019.

[24] P.-N. Le and H.-J. Kang, "Robot manipulator calibration using a model based identification technique and a neural network with the teaching learning-based optimization," *IEEE Access*, vol. 8, pp. 105 447–105 454, 2020.

[25] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[26] B. Mooring, M. Driels, and Z. Roth, *Fundamentals of Manipulator Calibration*. USA: John Wiley & Sons, Inc., 1991.

[27] D. Umbach and K. N. Jones, "A few methods for fitting circles to data," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 6, pp. 1881 – 1885, Dec. 2003.

[28] M. Proxima, "Fitting a circle to a cluster of 3D points," *Meshlogic*, 2016. [Online]. Available: https://meshlogic.github.io/posts/jupyter/curve-fitting/fitting-a-circle-to-cluster-of-3d-points

[29] M. D. Buhmann, "Radial basis functions," *Acta Numerica*, vol. 9, p. 1–38, 2000.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[31] NaturalPoint, Inc., *PrimeX 13*, 2025. [Online]. Available: https://docs.optitrack.com/hardware/cameras/ethernet-cameras/primex-13

[32] ——, *MOTIVE*, 2025. [Online]. Available: https://docs.optitrack.com/motive