Front. Mech. Eng. 2025, 20(2): 9
https://doi.org/10.1007/s11465-025-0825-7

RESEARCH ARTICLE

Path synthesis of spatial revolute–spherical–cylindrical–revolute mechanisms using deep learning

Xueting DENG, Anar NURIZADA, Anurag PURWAR (✉)

Department of Mechanical Engineering, Stony Brook University, New York 11794, USA
✉ Corresponding author. Email: anurag.purwar@stonybrook.edu (Anurag PURWAR)

ABSTRACT The design of single-degree-of-freedom spatial mechanisms tracing a given path is challenging due to the highly non-linear relationships between coupler curves and mechanism parameters. This work introduces an innovative application of deep learning to the spatial path synthesis of one-degree-of-freedom spatial revolute–spherical–cylindrical–revolute (RSCR) mechanisms, aiming to find the non-linear mapping between coupler curve and mechanism parameters and generate diverse solutions to the path synthesis problem. Several deep learning models are explored, including multi-layer perceptron (MLP), variational autoencoder (VAE) plus MLP, and a novel model using conditional $-\beta-$ VAE (c $-\beta-$ VAE). We found that the c $-\beta-$ VAE model with β = 10 achieves superior performance by predicting multiple mechanisms capable of generating paths that closely approximate the desired input path. This study also builds a publicly available database of over 5 million paths and their corresponding RSCR mechanisms. The database provides a solid foundation for training deep learning models. An application in the design of human upper-limb rehabilitation mechanism is presented. Several RSCR mechanisms closely matching the wrist and elbow path collected from human movements are found using our deep learning models. This application underscores the potential of RSCR mechanisms and the effectiveness of our model in addressing complex, real-world spatial mechanism design problems.

KEYWORDS spatial mechanism, neural networks, path synthesis, machine learning, deep learning, generative models

1 Introduction

Multi-degree-of-freedom (multi-DOF) robots, which can be programmed to produce paths with their end-effectors, are prevalent. Meanwhile, closed-loop, single-DOF mechanisms can generate high-order multivariate polynomials with only one actuator and do not require complex control and programming. Such mechanisms are light, easy to build and maintain, and less prone to mechanical failure. As a result, single-DOF mechanisms are widely used in daily life, such as in car window wipers, rehabilitation devices [1–4], and robots including flapping-wing robots. However, designing closed-loop single-DOF mechanisms, especially spatial mechanisms, is a difficult task because it requires specialized knowledge and access to design tools that can facilitate such design. Previous research efforts have been focused on generating a specific motion or functional relationship between input and output. In general, the motion synthesis problem is often simplified because the mechanisms can be decomposed using dyadic decomposition; however, the path synthesis problem does not benefit from such simplification. This work is concerned with the path synthesis of a single-DOF spatial mechanism called revolute–spherical–cylindrica–lrevolute (RSCR) (Fig. 1). Although spatial RSCR mechanisms have potential applications in functional, path, and motion synthesis, they have remained relatively underexplored. In addition, the planar RRRR mechanism can be viewed as a special spatial RSCR mechanism when all joints are in parallel planes and all rotational axes are perpendicular to the joint plane.

The goal of path synthesis is to find mechanisms whose coupler follows a desired path or a sequence of

Received May 2, 2024; accepted October 9, 2024; online April 27, 2025

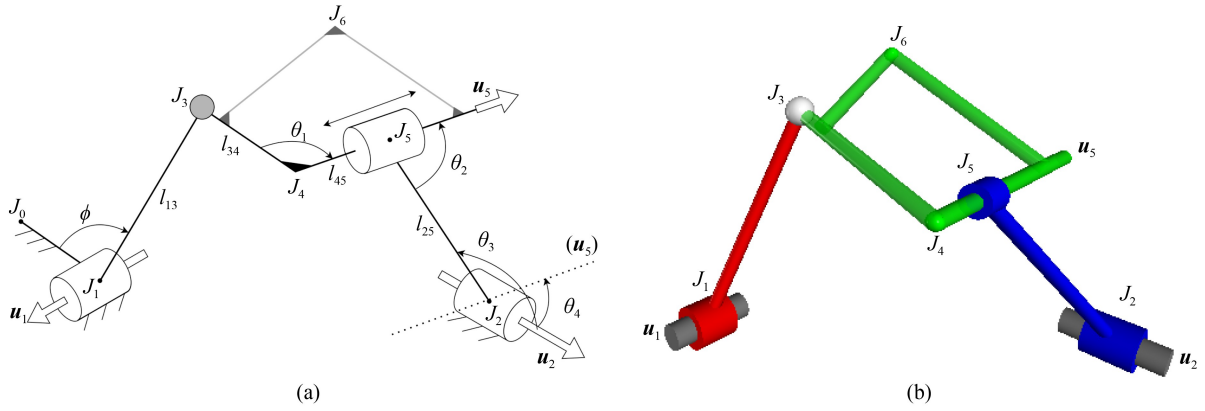


Fig. 1 RSCR mechanism that includes two fixed revolute (R) joints J_1 and J_2 with their respective rotational axes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, a spherical (S) joint J_3 rotating about $\boldsymbol{u}_1$ and connecting links l_{13} and l_{34}, a cylindrical (C) joint J_5 that slides along l_{45} and rotates about the rotational axis $\boldsymbol{u}_5$, and a welded joint J_4 for fixing the angle between l_{34} and l_{45}. Joint J_6 is another welded joint attached to the rigid body l_{345} and is also called the coupler point that generates the coupler curve. Joint J_0 is a point on the ground that only exists for expressing the input angle ϕ between vectors $\overrightarrow{J_1J_0}$ and $\overrightarrow{J_1J_3}$. Angles θ_1, θ_2, θ_3, and θ_4 are all constant angles. In particular, θ_1 is the angle between $\overrightarrow{J_4J_3}$ and $\overrightarrow{J_4J_5}$, θ_2 is the angle between $\overrightarrow{J_5J_2}$ and $\boldsymbol{u}_5$, θ_3 is the angle between $\overrightarrow{J_2J_5}$ and $\boldsymbol{u}_2$, and θ_4 is the angle between $\boldsymbol{u}_2$ and $\boldsymbol{u}_5$. (a) Kinematic structure of RSCR mechanism. (b) 3D model of RSCR mechanism.

points. For a single-DOF mechanism, the inverse process of determining the mechanism parameters of a desired path is challenging due to the non-linear relationships between the mechanism's parameters and its generated path. The direct analytical functions for a single-DOF mechanism, particularly for spatial RSCR mechanisms, are multivariate and of higher order [5,6]. Bagci [7] and Thompson [6] first analyzed the kinematic properties of RSCR spatial mechanisms, laying the groundwork and providing foundational formulas for synthesis problems. Chiang et al. [8] derived a motion synthesis method for accommodating four precision poses and relaxed specifications. Ananthasuresh and Kramer [9] and Watanabe et al. [5] worked on the branch identification of RSCR mechanisms. Ananthasuresh and Kramer [9] also classified RSCR mechanisms into cone, cylinder, and one-sheet hyperboloid types based on their geometry properties. Shrivastava and Hunt [10] highlighted that RSCR mechanisms can achieve dwell motion, a feature critical in many mechanical systems. Wang and Guo [11] demonstrated that RSCR mechanisms and other closed-chain mechanisms can be applied to the problem of stator blade adjustment in aero-engines. Most of these works focused on the function and motion synthesis problems and utilized special configurations of RSCR mechanisms to simplify calculations. For example, Osman and Segev [12] and Osman et al. [13] assumed that the two rotational axes of the revolute–cylindrical (RC) dyad are parallel, but Huang and Youm [14] assumed those two axes intersect. As two spatial lines, the two rotational axes of the RC dyad can be parallel, intersecting, or neither intersecting nor parallel. For parallel or intersecting configurations, the motion of coupler is constrained to a cone or a cylinder instead of a general one-sheet hyperboloid. Such simplification significantly reduces the complexity of calculations.

Deriving analytical relationships for path synthesis in spatial RSCR mechanisms and subsequently obtaining solutions are challenging tasks. With the advent of deep learning, its potential applications in engineering design problems have become evident [15]. These models can be used to 1) discover complex approximate relationships between inputs and outputs and 2) instantaneously generate many design solutions once the training is completed. Early in 2001, Vasiliu and Yannou [16] employed a multi-layer perceptron (MLP) deep learning model for the closed-path synthesis of planar four-bar linkages. Later, Galán-Marín et al. [17] focused on crank–rocker planar four-bar and improved the results produced by MLPs by changing the representation of the path from raw Cartesian coordinates to wavelets [18]. With the recent surge in deep learning, models with more complex structures than MLP, especially deep generative models, have proven effective in mechanism synthesis [19–26]. Variational autoencoder (VAE) [27] is known for extracting data features and thus can be applied to explore path features. The encoder of a VAE can map the input path to a low-dimensional latent space representation where similar paths can be searched in the neighborhood of the latent representation [19,20,28]. A method similar to the VAE also proved effective for the path synthesis of spatial 5-spherical–spherical (5-SS) mechanisms [21]. Instead of generating new mechanisms, the standard

VAE functions as a library, finding paths similar to the desired one from the database used for training. To overcome this defect, researchers proposed a pipeline that combines the VAE encoder and an MLP to extract features and generate new mechanisms [22]. They also examined the influence of various path representations on VAE performance. Meanwhile, the generative adversarial network was applied to planar four-bar synthesis problems, integrating kinematic and dynamic conditions [23]. Regenwetter et al. [15] published a review of the application of deep generative models in engineering design. Purwar and Chakraborty [24] recently discussed potential future research directions in robot mechanism design with deep learning.

Inspired by Nurizada et al. [29], this work introduces a novel generative model utilizing a conditional $-\beta-$VAE ($\mathrm{c}-\beta-$VAE) for the path synthesis of spatial RSCR mechanisms. $\mathrm{c}-\beta-$VAE is a modified version of the VAE for supervised learning that allows the model to generate reconstructed data with given conditions (c). This model also introduces an adjustable loss weight β. Our approach involves using a spatial path as the condition of $\mathrm{c}-\beta-$VAE, with the respective mechanisms serving as inputs to the model. The aim of the model is to predict mechanisms based on the provided spatial paths as the condition. For path synthesis after training, a desired path is provided, and the model combines this path with multiple random samplings from its trained latent space to generate several possible mechanisms that approximate the desired path. Inspired by the work of Nurizada and Purwar [22], we train several models with simple MLP architectures and VAE plus MLP architectures using the same dataset to demonstrate the effectiveness of our model relative to existing ones. By comparing the results of three different models, we establish the superiority of the novel model $\mathrm{c}-\beta-$VAE. This work also briefly describes the generation of a path database for RSCR mechanisms including a simulation algorithm and path normalization.

The remainder of the paper is organized as follows. Section 2 briefly introduces the structure of RSCR mechanisms, followed by the generation and processing of the database used in this work. Section 3 discusses the rationale for using deep learning and introduces the architectures of the employed deep learning models. Section 4 shows the analysis of the results from all architectures utilized. Finally, Section 5 presents a practical application of RSCR mechanisms in human upper-limb rehabilitation.

2 Database generation

Figure 1 shows the structure of an RSCR mechanism that includes two fixed revolute (R), a spherical (S) joint, a cylindrical (C) joint, and a welded joint. Joint J_6 is another welded joint attached to the rigid body l_{345} and is also called the coupler point that generates the coupler curve.

Deep learning requires a high-quality database with evenly distributed and sufficient data to achieve good results. The generation of such a database involves three main steps: implementing a simulator, collecting data from the simulator, and normalizing the collected data. This section briefly outlines these steps. Please see Deng et al. [30] for additional details.

2.1 Simulator

The simulator is implemented using the geometrical constraints of the spatial RSCR mechanism and a gradient-based optimization algorithm. The coordinates of J_3 are calculated from the input angle ϕ and l_{13}. Some geometrical constraint equations below are referenced from Javier and Eduardo [31]. In particular, the length constrains of rigid links l_{34} and l_{25} are presented as

$$f_1=\sqrt{(x_3-x_4)^2+(y_3-y_4)^2+(z_3-z_4)^2}-l_{34}=0, \tag{1}$$

$$f_2=\sqrt{(x_2-x_5)^2+(y_2-y_5)^2+(z_2-z_5)^2}-l_{25}=0, \tag{2}$$

where (x_a, y_a, z_a) are the coordinates of J_a. The dot product gives the constant angular relationships between links and axes:

$$f_3 = u_{43x}u_{5x}+u_{43y}u_{5y}+u_{43z}u_{5z}-l_{34}\cos\theta_1=0, \tag{3}$$

$$f_4 = u_{52x}u_{5x}+u_{52y}u_{5y}+u_{52z}u_{5z}-l_{25}\cos\theta_2=0, \tag{4}$$

$$f_5 = u_{25x}u_{2x}+u_{25y}u_{2y}+u_{25z}u_{2z}-l_{25}\cos\theta_3=0, \tag{5}$$

$$f_6 = u_{2x}u_{5x}+u_{2y}u_{5y}+u_{2z}u_{5z}-\cos\theta_4=0, \tag{6}$$

where the vectors $\overrightarrow{J_aJ_b}=(u_{abx}, u_{aby}, u_{abz})$ and $\boldsymbol{u}_a=(u_{ax}, u_{ay}, u_{az})$. Unfortunately, the dot product is unable to represent parallel relationships between $\boldsymbol{u}_5$ and $\overrightarrow{J_4J_5}$ because of the variable length of l_{45}. We use cross product to capture this geometrical constraint at $u_{5z}\neq 0$.

$$f_{7_1} = u_{45y}u_{5z}-u_{45z}u_{5y}=0, \tag{7}$$

$$f_{8_1} = u_{45z}u_{5x}-u_{45x}u_{5z}=0. \tag{8}$$

When $u_{5z}=0$, the two constraints change to

$$f_{7_2} = u_{45x}u_{5y}-u_{45y}u_{5x}=0, \tag{9}$$

$$f_{8_2} = u_{5z}=0, \tag{10}$$

where f_8 is automatically satisfied. The last

constraint equation is given by the unit vector u_5:

$$f_9 = \|\boldsymbol{u}_5\| - 1 = 0. \tag{11}$$

During the simulation, l_{13} is set to be the input link. The fixed joints J_1 and J_2 and their rotational axes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are assumed to be known. In this case, the coordinates of J_3 can be easily calculated while it is rotating about $\boldsymbol{u}_1$. The remaining unknown variables are J_4, J_5, and $\boldsymbol{u}_5$. The unknowns are set as a vector $\boldsymbol{\lambda} = (x_4, y_4, z_4, x_5, y_5, z_5, u_{5x}, u_{5y}, u_{5z})$. With nine constraint equations and nine unknowns, the solvable object function $F(\boldsymbol{\lambda})$ is

$$F(\boldsymbol{\lambda}) = \sum_{i=1}^{9} f_i^2(\boldsymbol{\lambda}) = 0. \tag{12}$$

For one simulation, the initial state of the mechanism, including the coordinates of all joints and vectors, is defined. During the simulation, a small angular increment is applied to input angle $\boldsymbol{\phi}$, which updates the coordinates of J_3. The numerical algorithm Powell's Hybrid Method (*hybr*) from *scipy.optimize.root* function in Python [32] is used to find the new root of F for the updated J_3. The coordinates of J_6 are calculated according to the relative coordinates of J_3 and J_4 as they form a rigid body.

2.2 Data collection

For deep learning training, we first need to create a database of mechanisms. We begin by setting up a $7 \times 7 \times 7$ mesh grid shown in Fig. 2 and locating the joints and axes of rotations as follows:

1) The center of the $7 \times 7 \times 7$ mesh grid is the origin $(0, 0, 0)$, and the x-, y-, and z-values of nodes range from $[-3, 3]$ with a unit interval 1.

2) The fixed revolute joint J_1 is located at $(0, 0, 0)$, and the rotational axis is fixed at $\boldsymbol{u}_1 = (0, 0, 1)$;

3) Given that three non-collinear points define a unique plane in $\mathbb{R}^3$, without losing generality, three joints J_1, J_2, and J_3 are set on the same plane $z = 0$ but their x- and y-coordinates are allowed to vary among the nodes.

4) J_4 is set at the same coordinates as J_5 for simplification. Given that J_4 exists to maintain the constant angle θ_1, its exact coordinates do not matter. Although in practice, J_5 and J_4 cannot physically meet. In theory, once θ_1 is a constant, any J_4 that lies along the rotational axis $\boldsymbol{u}_5$ results in the same mechanism.

5) Rotational axes $\boldsymbol{u}_2$ and $\boldsymbol{u}_5$ are set in a unit spherical coordinate system $(1, \beta_2, \beta_5)$ where $\beta_{i=2,5}$ is chosen from $\left[0, \frac{\pi}{3}, \frac{2\pi}{3}\right]$. This step offers nine different choices of $\boldsymbol{u}_2$ and $\boldsymbol{u}_5$, and their values can be calculated from

$$\boldsymbol{u}_i = (\sin\beta_1 \cos\beta_2, \sin\beta_1 \sin\beta_2, \cos\beta_1), \; i = 2, 5. \tag{13}$$

6) Other joints and axes are chosen, and the coupler point J_6 is assumed to be located at the nodes of a $3 \times 3 \times 3$ mesh grid attached to the rigid link l_{345} as shown in Fig. 3.

The above rules aim to ensure the joints are located at all grid nodes to build a relatively evenly distributed database. Unfortunately, even with such a small mesh grid and the limited choices of $\beta_{i=2,5}$, the number of possible combinations of different mechanisms is $7^2 \times 7^2 \times 7^3 \times 3^2 \times 3^2 = 66706983$ with $3 \times 3 \times 3 = 27$ coupler curves for each combination. Moreover, the grid-based generation method provides a limited combination of link lengths, leading to numerous repeated calculations. To streamline the generation process and avoid duplicated mechanisms, we only select one combination for each unique link

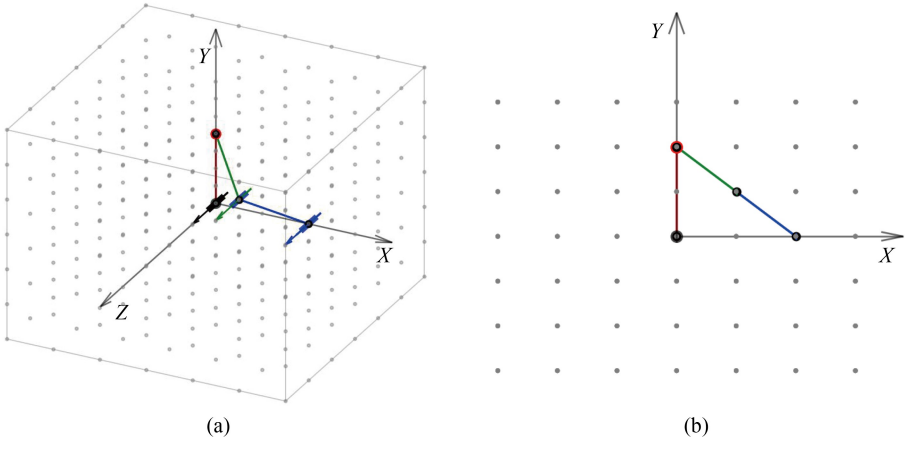


Fig. 2 Spatial $7 \times 7 \times 7$ mesh grid for chosen mechanism's initial state. The initial state does not include J_6, and J_4 is set at the same coordinates as J_5. (a) Spatial $7 \times 7 \times 7$ mesh grid. (b) Spatial $7 \times 7 \times 7$ mesh grid projection on the X–Y plane.

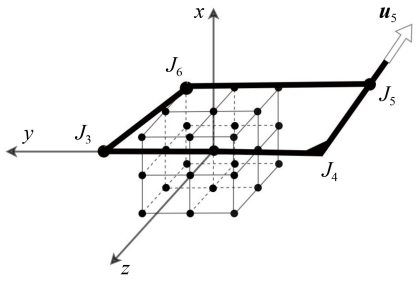


Fig. 3 Spatial $3 \times 3 \times 3$ mesh grid for choosing coupler points J_6. The spatial mesh grid is attached to the rigid body l_{345}. The origin of the mesh grid is the middle point of l_{34}, its y-axis aligns along $\overrightarrow{J_4J_3}$, its x-axis aligns along the direction of $\boldsymbol{u}_5 \times \overrightarrow{J_4J_3}$, and its z-axis is calculated following a right-handed coordinate system.

ratio $(l_{13}/l_{12}, l_{34}/l_{12}, l_{25}/l_{12})$. A closed-loop mechanism can be assembled in different ways even with the same links' lengths, lead to the generation of circuit defects [33] during synthesis. Although the grid-based method generates a limited number of different link ratios, initializing the joints' locations provides the benefit of avoiding the circuit defect. The density of the mesh grid is determined by considering the computing time, storage requirement, and the actual effect on the deep learning results. In the end, the algorithm produces 196912 mechanisms, each with 27 coupler curves. Given that this work focuses only on closed paths, the final database consists of 5316624 coupler curves with their corresponding mechanism after the open paths are removed.

Before the data are sent to the deep learning models, each raw coupler curve with Cartesian points obtained from simulator is converted to a third-degree B-spline interpolation curve [34]. Thus, each path has 100 data points that are uniformly sampled with the same time parametrization. The path is then translated to the center of mean, rotated, and reflected with principal component analysis [20,35–37] and independent component analysis [38]. Figure 4 shows a mechanism with its path before and after the normalization. The final database used in this work contains normalized paths with 100 evenly distributed data points each. The generated database is available at Kaggle website [39].

3 Deep learning models

The analytical functions for mapping from the RSCR mechanism parameters to its coupler curve have been derived [5,6]. However, an inverse function from the coupler curve to the mechanism parameters does not exist because multiple mechanism parameters could correspond to the same path. Deep learning excels at approximating the non-linear relationships between different data sets. As mentioned in Section 2, during data generation, mechanisms with the same link ratio are avoided to maintain a one-to-one mapping between the path and the mechanism. Furthermore, the mechanism parameters are represented by the initial coordinates of the joints and rotational axes to prevent circuit defects. Similar methods for planar linkage mechanisms have been validated in Refs. [16,17,22].

In our work, we leverage the capabilities of deep learning models to learn the complex relationship between the coupler curve and its corresponding RSCR mechanism parameters, essentially solving the closed-path synthesis problem for spatial RSCR mechanisms. We propose three distinct models for this purpose. Similar to Refs. [16,17], the first model utilizes an MLP to directly map the coupler curve to its mechanism parameters. Based on Nurizada and Purwar [22], the second model first trains a VAE to represent the features of the path data by latent space representation. It then incorporates a separate MLP to learn the mapping between the latent representations of the path and the respective mechanisms. Different from the first one, this model allows for the exploration of the latent space in the vicinity of the latent representation of the input, yielding multiple potential solutions, i.e., mechanisms,

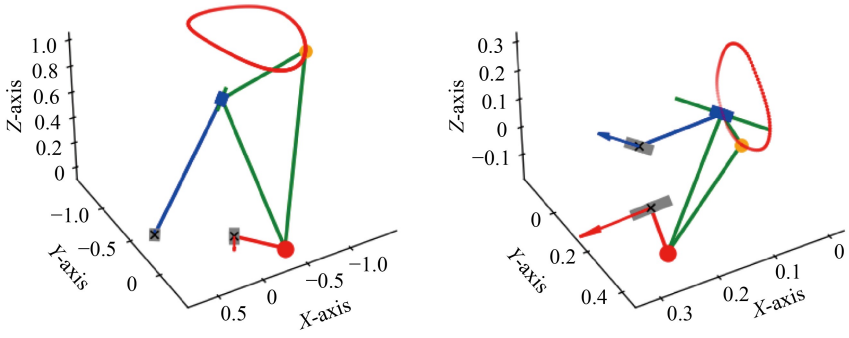


Fig. 4 RSCR mechanism and its path before (left) and after normalization (right).

for a single input path. The third model introduces a novel model $\mathrm{c}-\beta-\mathrm{VAE}$ that treats the coupler curve as a condition for reconstructing mechanism parameters. After training, $\mathrm{c}-\beta-\mathrm{VAE}$ learns a set of latent distributions that are conditioned on an input. For predicting mechanisms for the desired input path, data can be randomly sampled from the latent distributions and combined with the desired path before being fed to the decoder. Given that the random samples are drawn from the learned latent distributions, they are valid and ensure diverse output mechanisms for a single input path. Later in this section, we provide the theoretical background for each model. The results and comparisons of these methods are discussed in Section 4.

This work aims to train deep learning models that take a desired path as input and then predict one or several RSCR mechanisms capable of generating paths that approximate the input (Fig. 5). First, the raw input path is normalized to be represented as 100 3D Cartesian points flattened into a 1×300 vector. This vector corresponds to the normalized path coordinates $\boldsymbol{P}=(P_{1x},P_{1y},P_{1z},\ldots,P_{100x},P_{100y},P_{100z})$. The output data, representing the mechanisms, is encoded as 1×18 vectors $\boldsymbol{J}=(\boldsymbol{J}_1,\ \boldsymbol{J}_2,\boldsymbol{J}_3,\ \boldsymbol{J}_4,\ \boldsymbol{u}_2,\boldsymbol{u}_5)$ of the initial coordinates of RSCR mechanisms. The rotational axis $\boldsymbol{u}_1$ is always perpendicular to $\overrightarrow{J_1J_3}$, and J_5 coincides with J_4 for the initial coordinates. The rest of this section is devoted to the theoretical background needed to understand the proposed methods.

3.1 MLP

The MLP architecture is a fundamental model within the realm of deep learning. It comprises an input layer, one or more hidden layers, and an output layer. Each layer consists of multiple neurons, and every one of which is interconnected with neighbor layers' neurons through weighted connections. For this reason, the MLP is also called a "fully connected neural network". An exemplary architecture of an MLP is depicted in Fig. 6.

Assuming m neurons in the jth layer, the output value of the ith neuron in $(j+1)$th layer n_i^{j+1} can be calculated as

$$n_i^{j+1}=g_j\left(\sum_{k=1}^{m}w_k^j n_k^j+b_k^j\right), \tag{14}$$

where w_k^j and b_k^j represent the weights and bias of kth neuron in jth layer connected to the ith neuron in $(j+1)$th layer, respectively, and g_j denotes the activation function, introducing non-linearity into the output.

A critical aspect of deep learning is the utilization of a loss function to measure the model's accuracy and to train the model by calculating the difference between its predictions and the actual data. The objective of training is to minimize this loss function by fine-tuning the model's weights and biases. During training, the MLP undergoes forward-propagation, where data traverse through the network, followed by the adjustment of weights and biases based on the loss function's gradient through backpropagation [40].

For the evaluation of the difference between the predicted mechanism $\hat{\boldsymbol{J}}$ and the ground truth mechanism $\boldsymbol{J}$, the mean squared error (MSE) is chosen as the loss function:

$$L_{\mathrm{MSE}}=\frac{1}{18}\sum_{i=1}^{18}\left\|\hat{\boldsymbol{J}}_i-\boldsymbol{J}_i\right\|^2, \tag{15}$$

where $\hat{\boldsymbol{J}}$ denotes the predicted value of $\boldsymbol{J}$ from the model. The rectified linear unit activation function [41], coupled with batch normalization [42] and a dropout layer [43], is employed to enhance performance and counteract overfitting during training.

Our first method capitalizes on the approximating capabilities of MLPs to directly learn the mapping

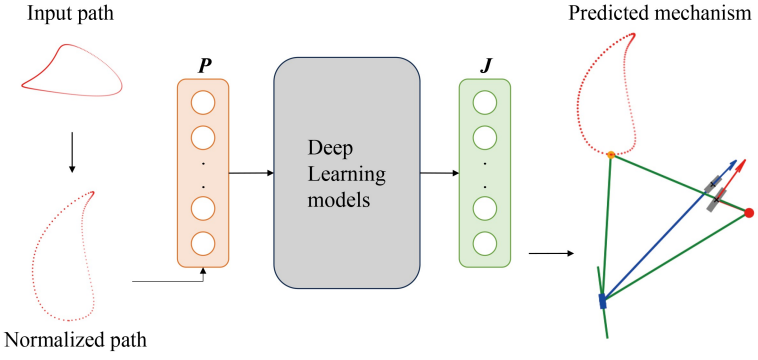


Fig. 5 General pipeline of deep learning. An input path will be normalized first and then sent to a deep learning model that predicts a mechanism or mechanisms capable of generating paths that approximate the input path.

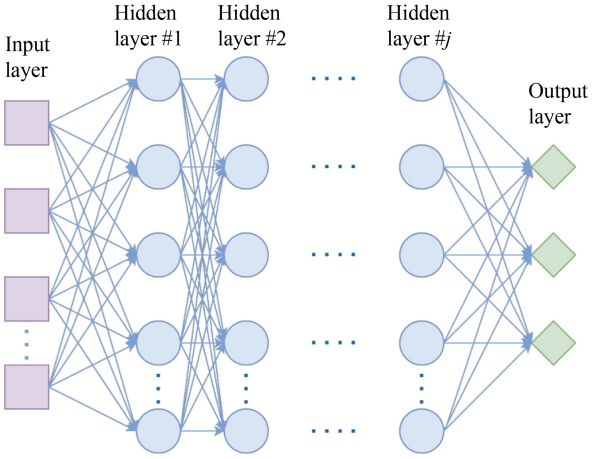


Fig. 6 Multi-layer perceptron: Each layer's neurons are fully connected to the neighboring layers' neurons with different trainable weights. An activation function is applied after each layer.

between the path and its respective mechanism, which serve as input and output to the MLP, respectively. We train several MLP architectures by varying the number of hidden layers and the number of neurons within each layer. The general architecture is shown in Table 1.

Table 1 MLP general architecture[a]

Layer	Layer dimension	Added functions
Input layer	300	Batch Norm + ReLU + Dropout
Hidden layer #1	n_1	Batch Norm + ReLU + Dropout
Hidden layer #2	n_2	Batch Norm + ReLU + Dropout
. . .	. . .	. . .
Hidden layer #N	n_N	Batch Norm + ReLU + Dropout
Output layer	18	Batch Norm + Dropout

a: n_j is the number of neurons in the jth layer. The detailed values of n_j and N are presented in the next section.

3.2 VAE

Introduced by Kingma and Welling [27], the VAE excels in data dimensionality reduction and reconstruction. It follows the classic autoencoder architecture, employing an encoder to reduce input data $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ to a low-dimensional latent space representation $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$. A decoder then reconstructs the latent space representation into output $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m)$ while maintaining the input's format. In generative tasks, the output $\hat{\boldsymbol{x}}$ is expected to be similar but not exactly the same as the input $\boldsymbol{x}$ for generating new data. Although this work primarily employs the encoder of the VAE to extract path data into latent space, a brief introduction of the full VAE is provided below.

To generate new data $\hat{\boldsymbol{x}}$, the true distribution of input $p(\boldsymbol{x})$ must be known. Unfortunately, $p(\boldsymbol{x})$ is often unknown. Instead, we assume a latent space representation $\boldsymbol{z}$ where $p(\boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{I})$ to indirectly calculate $p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})\, p(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}$. In practice, the latent space can be complex and high dimensional, making $p(\boldsymbol{x}|\boldsymbol{z})$ nearly zero for most $\boldsymbol{z}$. The key idea behind VAE is to sample $\boldsymbol{z}$ values that are likely to produce $\boldsymbol{x}$ with $p(\boldsymbol{z}|\boldsymbol{x})$ and to estimate $p(\boldsymbol{x})$ just from those samples [11]. This process leads to one of the most important equations in VAE:

$$\begin{aligned} &D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z}|\boldsymbol{x})\right] \\ &= \mathbb{E}_{z\sim q}\left[\log q(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}|\boldsymbol{x})\right], \end{aligned} \tag{16}$$

where $q(\boldsymbol{z}|\boldsymbol{x})$ is a new distribution that approximates $p(\boldsymbol{z}|\boldsymbol{x})$, and D_{KL} is the Kullback–Leibler divergence (KLD), a measure of the statistical distance between two distributions. According to Bayes' theorem, Eq. (16) can be transformed into

$$\begin{aligned} &\log p(\boldsymbol{x}) - D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z}|\boldsymbol{x})\right] \\ &= \mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z})\right]. \end{aligned} \tag{17}$$

Given that the value of KLD is always non-negative, the goal is to find the lower bound of $\log p(\boldsymbol{x})$ that involves minimizing $D_{\mathrm{KL}}[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z}|\boldsymbol{x})]$. On the right-hand side of Eq. (17), the term $\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right]$ corresponds to reconstructing $\boldsymbol{x}$ from $\boldsymbol{z}$, which is also the function of the decoder in VAE. The term $D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z})\right]$ relates to the encoder's role in making $q(\boldsymbol{z}|\boldsymbol{x})$ close to $p(\boldsymbol{z})$. To find the lower bound of $\log p(\boldsymbol{x})$, we should maximize $\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right]$ and minimize $D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z})\right]$. Ultimately, the VAE loss function is

$$L = -\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] + D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z})\right]. \tag{18}$$

Here, the prior distribution $p(\boldsymbol{z})$ is assumed to follow a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$. The learnable distribution $q(\boldsymbol{z}|\boldsymbol{x})$ follows $q(\boldsymbol{z}|x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with the VAE learning $\mu_i = f_{\text{encoder}}(x_i)$ and $\log\sigma_i^2 = f_{\text{encoder}}(x_i)$. With these assumptions, the loss function of VAE can be further simplified to

$$L = L_{\mathrm{MSE}}(\hat{\boldsymbol{x}}, \boldsymbol{x}) + \frac{1}{2}\left(-\log\boldsymbol{\sigma}^2 + \boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 - 1\right). \tag{19}$$

Finally, a technique called reparameterization trick is utilized for sampling the latent data $\boldsymbol{z}$. To ensure the sampling step remains differentiable, VAE samples a number ε from $\mathcal{N}(0, \mathbf{I})$ and calculates the latent data as

$$\boldsymbol{z} = \boldsymbol{\mu} + \varepsilon \times \boldsymbol{\sigma}. \tag{20}$$

A well-trained VAE can map similar data close to each other in the latent space $\boldsymbol{z}$. Given two sets of input $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the encoder outputs $q(\boldsymbol{z}|\boldsymbol{x}_1) \sim$

$\mathcal{N}\left(\mu\left(\boldsymbol{x}_1\right), \sigma_i^2\left(\boldsymbol{x}_1\right)\right)$ and $q\left(\boldsymbol{z}|\boldsymbol{x}_2\right) \sim \mathcal{N}\left(\mu\left(\boldsymbol{x}_2\right), \sigma_i^2\left(\boldsymbol{x}_2\right)\right)$, respectively. Owing to the similarity between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, distributions $q\left(\boldsymbol{z}|\boldsymbol{x}_1\right)$ and $q\left(\boldsymbol{z}|\boldsymbol{x}_2\right)$ will have similar means and variances to ensure that the latent variables $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ sampled from these distributions are close to each other.

In our application, the VAE has a two-fold task—it is used to reduce the dimension of the input path and map similar paths together. After training, the VAE can project an input path $\boldsymbol{P}$ with 300 dimensions into a low-dimensional latent space of n dimensions. We then train an MLP that learns the mapping between the latent representation of the path to their respective mechanisms (Fig. 7). Similar to the first method, we train several VAEs with varying latent dimensions to assess their impact on the reconstruction quality. The detailed architecture of VAE is shown in Table 2.

3.3 Attention mechanisms

Prior to the discussion of the last deep learning model, an important concept known as attention mechanism must be introduced. Renowned for their ability to focus on the most relevant parts of the input data for a given task, attention mechanisms have become one of the most prominent neural network architectures. Originally developed for natural language processing applications [44], attention mechanisms have been adapted across various fields, including computer vision [45].

The architecture of a commonly used attention mechanism called scaled dot-product attention comprises three main components, Query $(\boldsymbol{Q})$, Key $(\boldsymbol{K})$, and Value $(\boldsymbol{V})$. All $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ values are calculated by their own trainable single linear layer from the input data of the attention mechanism. Every query takes a different amount of information from the values by weighing them with their keys. This process allows the model to pay close attention to the data that has a great impact on the overall model. The dimensions of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are often chosen to be equal for simplicity but can be set to any value. Each component is calculated from its own trainable MLP and represents the input data from different perspectives. The output of the attention mechanism is computed by [44,46]

$$\text{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{21}$$

where the softmax function facilitates the conversion of the weighted scores into a probability distribution over the $\boldsymbol{V}$ vectors. In this context, $\sqrt{d_k}$ represents the dimensionality of the $\boldsymbol{K}$ vectors.

Self-attention is a foundational version of the attention mechanism, where the $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ vectors are all derived from the same input data. This process allows the model to focus on different parts of a single input, capturing various dependencies and relationships within that input. Multi-head attention is an enhanced version of the self-attention mechanism that incorporates multiple parallel attention mechanisms. By calculating several attentions in parallel, multi-head attention employs different sets of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ vectors to determine distinct sets of weights for the input data and thereby capture various aspects of the data.

Cross-attention is another variant where the attention mechanism is applied between two distinct

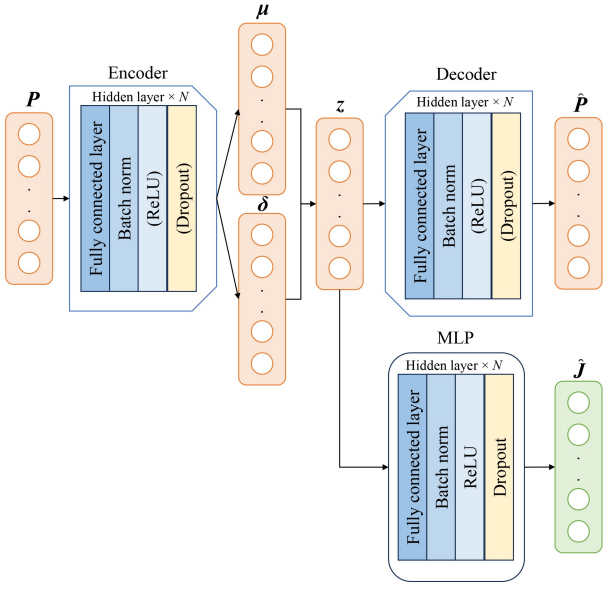


Fig. 7 Architecture of VAE plus MLP.

Table 2 VAE followed by MLP architecture

Layer/module	(Input, Hidden, Output) dimension	Description
Encoder MLP to $\boldsymbol{\mu}$	(300, –, Latent dim)	A single linear layer for calculating $\boldsymbol{\mu}$
Encoder MLP to $\log\boldsymbol{\sigma}^2$	(300, –, Latent dim)	A single linear layer for calculating $\log\boldsymbol{\sigma}^2$
Reparameterization trick	–	Latent space $\boldsymbol{z}$ calculation $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\varepsilon} \times \boldsymbol{\sigma}$
Decoder MLP	(Latent dim, –, 300)	A single linear layer for path reconstruction
MLP	(Latent dim, 8 × 4096, 18)	Separate MLP architecture with 8 hidden layers and 4096 neurons in each layer for mapping the latent space obtained from the encoder to the mechanism parameters

sets of inputs to integrate information across these sets. In cross-attention, $\boldsymbol{Q}$ is calculated from the first set, and $\boldsymbol{K}$and $\boldsymbol{V}$ are derived from the second set. This approach enables the model to effectively combine the two distinct sets and capture complementary information from both sources. For additional details on multi-head attention and cross-attention, please refer to Ref. [44]. In this work, we use all mentioned formats of attention to concentrate the data for $\mathrm{c}-\beta-\mathrm{VAE}$.

3.4 Conditional variational autoencoder ($\mathrm{c}-\mathrm{VAE}$)

$\mathrm{c}-\mathrm{VAE}$ [47], a variation of VAE designed for supervised learning, incorporates a condition alongside the input, allowing the model to generate reconstructed data with a given condition. The general architecture of $\mathrm{c}-\mathrm{VAE}$ is similar to that of VAE with an encoder, latent space, and a decoder. The three variables in $\mathrm{c}-\mathrm{VAE}$ are input $\boldsymbol{x}$, condition $\boldsymbol{y}$, and latent space data $\boldsymbol{z}$. The encoder processes the merged data of the $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate $\boldsymbol{z}$, which is then merged with $\boldsymbol{y}$ and fed into the decoder. The decoder's output, $\hat{\boldsymbol{x}}$, is a reconstruction of the input while satisfying the given condition. The loss function of $\mathrm{c}-\mathrm{VAE}$ is an extension of the regular VAE and is given by

$$L = -\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y})\right] + D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})\,||p(\boldsymbol{z}|\boldsymbol{y})\right]. \tag{22}$$

$\beta-\mathrm{VAE}$ is introduced to encourage a disentangled representation of the latent space [48]. As a hyper-parameter, β controls the balance of reconstruction loss and the KL divergence by scaling the KL divergence term. The loss of $\beta-\mathrm{VAE}$ is given by

$$L = -\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] + \beta\cdot D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x})\,||p(\boldsymbol{z})\right]. \tag{23}$$

In this work, we employ a combination of $\beta-\mathrm{VAE}$ and $\mathrm{c}-\mathrm{VAE}$ as $\mathrm{c}-\beta-\mathrm{VAE}$. Thus, the loss function of the final model is

$$L = -\mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y})\right] + \beta\cdot D_{\mathrm{KL}}\left[q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})\,||p(\boldsymbol{z}|\boldsymbol{y})\right]. \tag{24}$$

Our $\mathrm{c}-\beta-\mathrm{VAE}$ model takes the initial coordinates of mechanism $\boldsymbol{J}$ as input and expected (reconstructed) output. The normalized path coordinates $\boldsymbol{P}$, which serves as the condition. During training, the MSE loss minimizes reconstruction error between input and output coordinates of mechanisms' joints, aiding accurate mechanism parameter reconstruction. The aim is to reconstruct mechanisms from the database and generate defect-free mechanisms for a given coupler curve by sampling from the latent space.

Figure 8 shows our third model's architecture, which also incorporates self- and cross-attention mechanisms. As mentioned in Section 3.3, self-attention evaluates the relevance of elements within the same input vector, and cross-attention calculates the correlations between distinct vectors.

The process begins by amalgamating the distinctive features of the input mechanism parameters with the path condition, standardizing the input mechanism $\boldsymbol{J}$ and the path condition $\boldsymbol{P}$ using MLPs to obtain $\mathbf{e}_{\boldsymbol{J}}$ and $\mathbf{e}_{\boldsymbol{P}}$, respectively. A cross-attention then takes $\mathbf{e}_{\boldsymbol{J}}$ as the first input for calculating $\boldsymbol{Q}$ and $\mathbf{e}_{\boldsymbol{P}}$ as the second input for calculating $\boldsymbol{K}$ and $\boldsymbol{V}$. This cross-attention is applied to fully integrate information across $\mathbf{e}_{\boldsymbol{J}}$ and $\mathbf{e}_{\boldsymbol{P}}$. The cross-attention is followed by a self-attention. Alatent vector $\boldsymbol{z}$ is then generated by MLPs after the self-attention block. The latent vector $\boldsymbol{z}$ is taken into a second cross-attention again with $\mathbf{e}_{\boldsymbol{P}}$ and then sent to another self-attention block. In the end, a single linear layer projects the results from the self-attention block to $\hat{\boldsymbol{J}}$.

During inference, the exploration of the latent space within specified path conditions allows for the generation of mechanisms closely aligned with the desired path. The number of self-attention blocks, akin to the layer count in the MLP model, serves as a hyper-parameter. Optimal block numbers minimize loss until a threshold is reached, beyond which the training efficacy diminishes. The detailed architecture of $\mathrm{c}-\beta-\mathrm{VAE}$ is presented in Table 3.

4 Results and discussion

After the models' architectures are detailed in Section 3, the results for all the architectures employed are presented in this section. Our dataset comprises 5316624 data paths in total, with 90% of the data used for training, 5% for validation, and 5% for testing. All the models discussed in this paper were trained on the same dataset. To evaluate the robustness of our model, we added 20% random paths out of the original test set to the test set. These paths are generated by RSCR mechanisms with randomly picked joint locations and rotational axes without the discrete mesh grid nodes. We employed the Hausdorff distance [49] to assess similarity between two paths. Assuming A and B are two sets of points, the Hausdorff distance is given by $H(A,B)=\max\{h(A,B), h(B,A)\}$, where $h(A,B)=\max_{a\in A}\{\min_{b\in B} d(a,b)\}$ and $d(a,b)$ is any metric between the two sets of points. Experimental studies indicated that a Hausdorff distance of 0.027 is indicative of satisfactory similarity between two paths, and this threshold is employed to evaluate satisfactory results.

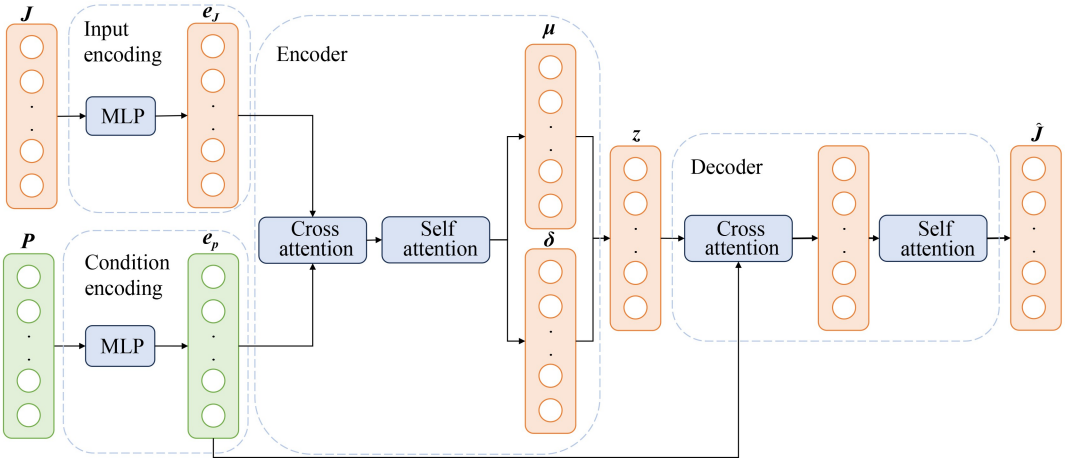


Fig. 8 Architecture of $\mathrm{c}-\beta-\mathrm{VAE}$.

Table 3 Detailed architecture of $\mathrm{c}-\beta-\mathrm{VAE}$

Layer/module	Input dim	Output dim	Description
Mechanism's joint encoding	18	2048	An MLP block for input mechanism $\boldsymbol{J}$ encoding
Path condition encoding	300	2048	An MLP block for input (condition) path $\boldsymbol{P}$ encoding
Encoder cross-attention block	2×2048	2048	Cross-transformer block for encoder attention
Encoder self-attention block	2048	2048	Self-transformer block for encoder
Mean vector calculation	2048	2048	A single linear layer for calculating $\boldsymbol{\mu}$
Logvar vector calculation	2048	2048	A single linear layer for calculating $\log(\boldsymbol{\sigma}^2)$
Reparameterization trick	2×2048	2048	Reparameterization trick for latent vector calculation
Decoder cross-attention block	2×2048	2048	Cross-transformer block for decoder attention
6 decoder self-attention blocks	2048	2048	List of self-transformer blocks for decoder
Mechanism's joint predictor	2048	18	A single linear layer for predicting mechanism $\hat{\boldsymbol{J}}$

Table 4 shows only the most significant MLP models that we trained. Starting with a learning rate of 0.001 and employing an automatic scheduler for learning rate reduction, we trained each of these models for over 200 epochs until the loss plateaued. The initial training loss for all models began at $\sqrt{L_{\mathrm{MSE}}} \approx 0.62$. Every model contains several hidden layers, each with the same number of neurons in our case; the detailed architecture is shown in Table 1. For the test database evaluation, only the predicted mechanism that generates a closed path, the focus of this work, was considered valid. A predicted path was deemed satisfactory if its Hausdorff distance from the input path is less than 0.027. The table also includes the mean and median values of the Hausdorff distances across all the test paths.

The results presented in Table 4 indicates that deep MLP architectures with more neurons tend to perform better than the shallow ones with few neurons. However, models with a larger number of

Table 4 Comparison of MLP models (satisfactory: $H < 0.027$)

Number of hidden layers	Number of neurons (per hidden layer)	Train loss, $\sqrt{L_{\mathrm{MSE}}}$	Test loss, $\sqrt{L_{\mathrm{MSE}}}$	Test path			
				Closed path/%	Satisfactory/%	Mean	Median
3	64	0.4830	0.4833	64.2	6.8	0.0760	0.0667
3	2048	0.2603	0.3503	49.0	10.1	0.0726	0.0643
6	2048	0.2037	0.2968	54.3	12.6	0.0651	0.0554
8	2048	0.1619	0.2728	57.2	15.5	0.0641	0.0520
9	2048	0.1189	0.2591	61.0	21.7	0.0570	0.0482
8	4096	0.0898	0.2386	62.9	22.5	0.0595	0.0435
9	4096	0.0877	0.2386	65.6	27.9	0.0492	0.0291

parameters require long training times. The most effective model is an MLP with 9 hidden layers and 4096 neurons in each layer. The disparity between a low median Hausdorff distance and a higher mean Hausdorff distance suggests that this model performs well for certain types of input paths but is less effective for others. This characteristic is further illustrated in Fig. 9, which shows a comparison of the predicted paths (in green) against randomly chosen input paths from the test dataset (in red). For clarity, we do not show the mechanisms generating the predicted path.

Table 5 presents three of the VAEs with MLP models we have trained; for the detailed architecture, please see Table 2. All three models share the same encoder architecture, which consists of a single MLP layer mapping the input dimensions to the latent space dimensions and a decoder that reverses this

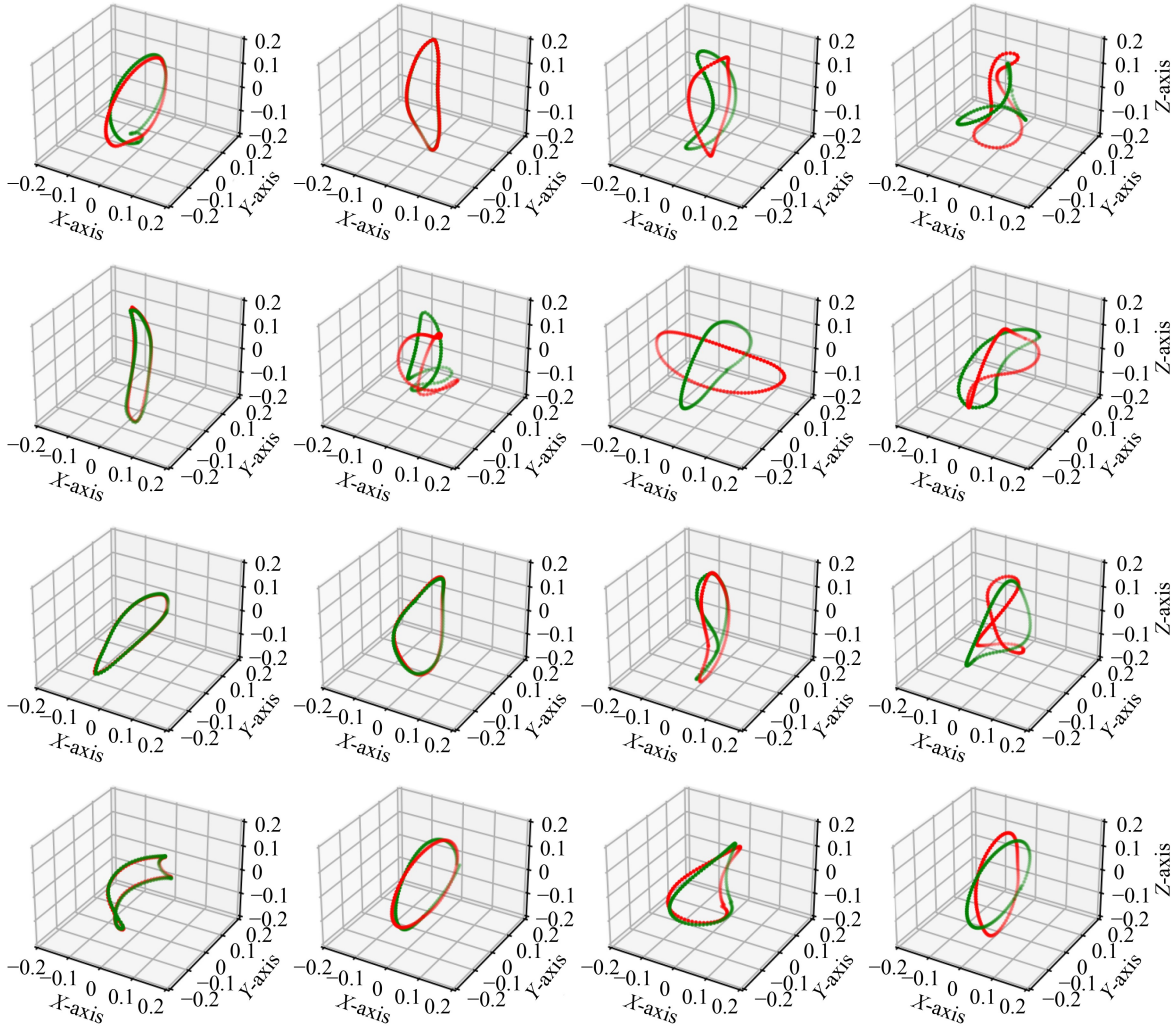


Fig. 9 Predicted paths (green) and input paths (red) from MLP 9 × 4096. The input paths are randomly chosen from the test dataset. The results show that this model is only sufficient for paths of certain shapes.

Table 5 VAE plus MLP model comparison. The MSE and KLD loss are the result from VAE model, and the mapping MSE loss is obtained from the following MLP model (satisfactory: $H < 0.027$)

Latent space dim	Train loss			Test loss			Test path			
	MSE	KLD	Mapping MSE	MSE	KLD	Mapping MSE	Closed path/%	Satisfactory/%	Mean	Median
64	0.9487	1.4950	0.1186	0.9165	1.4920	0.2250	65.3	27.2	0.0542	0.0383
128	0.4186	0.8926	0.1101	0.3695	0.8910	0.2208	65.8	30.0	0.0481	0.0307
256	0.3302	0.5364	0.1008	0.2372	0.5348	0.2177	68.2	35.1	0.0473	0.0258

mapping. The subsequent MLP architectures in these models are identical, each featuring 8 hidden layers with 4096 neurons per layer. We also explored various architectures with different encoder/decoder configurations and subsequent MLP architectures, discovering that the dimensionality of the latent space has the most significant impact on the results. Each VAE model was trained for over 400 epochs with an automatically learning rate reduction scheduler until the loss plateaued. Similarly, each MLP was trained for over 200 epochs.

Compared with the MLP only method, the VAE plus MLP architecture yields slightly better results, achieving a higher ratio of good paths and lower mean and median values of the Hausdorff distance. Although the MLP only method generates a single mechanism approximating the input path, the VAE plus MLP architecture can produce multiple predicted mechanisms, which is critical to support concept generation. During training, the latent space data are extracted directly as the input for the subsequent MLP. With the trained model, the input desired path is first mapped to a latent representation. Samplings around the neighborhood of this latent representation are then fed to the subsequent trained MLP, yielding multiple results. Figure 10 shows a comparison of the predicted paths in green with the randomly chosen input paths from test dataset in red. Figure 11(a) displays nine different predicted paths for the same input path in red, and Figure 11(b) shows nine different RSCR mechanisms corresponding to these nine predicted paths. These results demonstrate that the VAE plus MLP architecture performs better than the MLP only architecture and provides various mechanisms for the same input.

Table 6 presents some $\mathrm{c}-\beta-\mathrm{VAE}$ models we have trained, highlighting the impact of varying β values. We found that different β values impact the results.

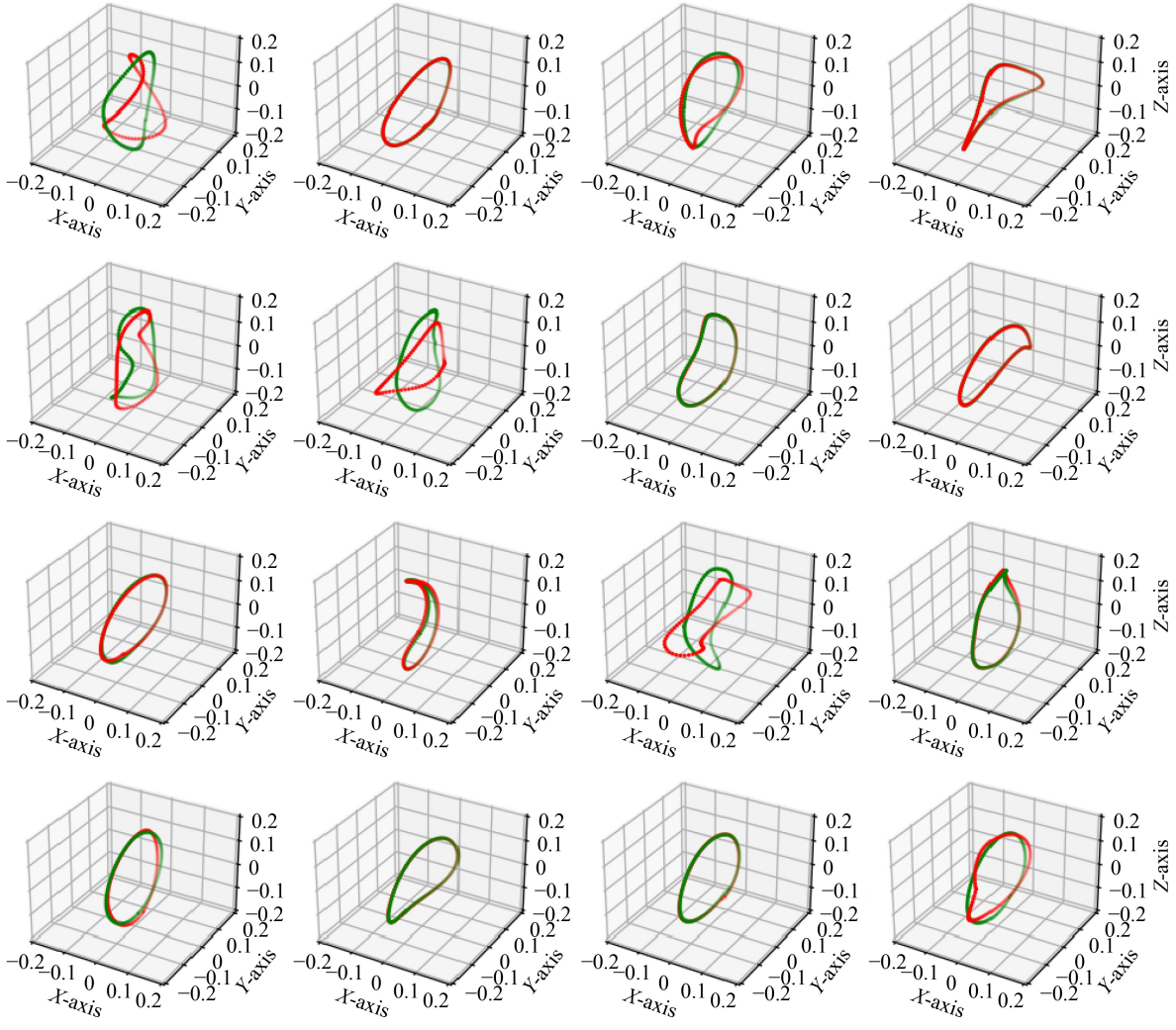


Fig. 10 Predicted paths (green) and input paths (red) from VAE plus MLP (latent space dim = 256) model. The input paths are randomly chosen from the test dataset.

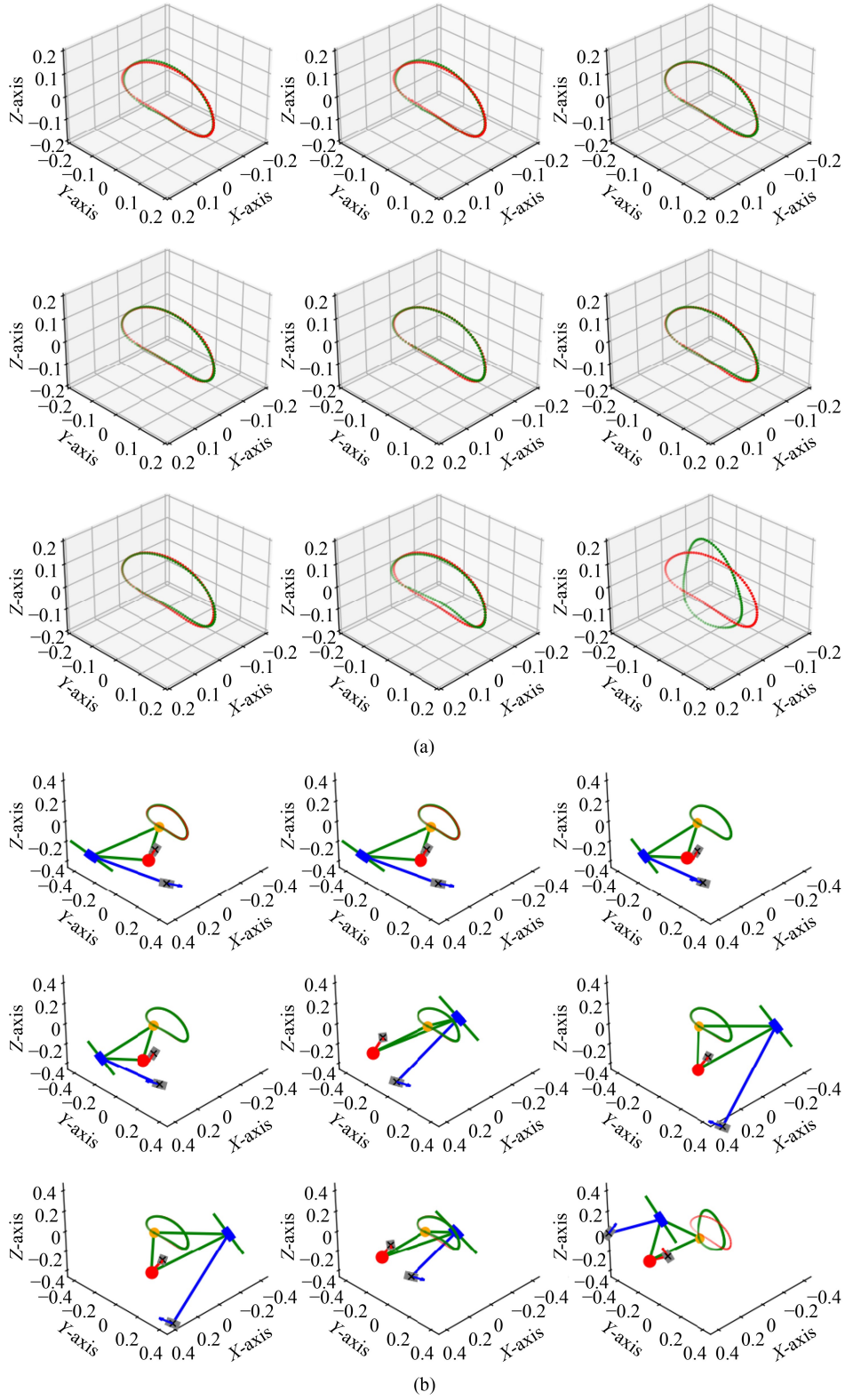


Fig. 11 Predicted paths (green) and mechanisms with the same input path (red) from VAE plus MLP (latent space dim = 256) model. (a) Predicted paths (green) with the same input path (red) from VAE plus MLP (latent space dim = 256) model. (b) Predicted paths (green) and their corresponding RSCR mechanisms.

The listed architectures utilize the exact same architecture in Table 3 except β. Compared with the other models discussed previously, $\mathrm{c}-\beta-\mathrm{VAE}$ achieves significantly better results. During testing, we took each test path as a condition, combined it with 30 random samples from the latent space, and sent it to the decoder to generate 30 different mechanisms. After decoding, the predicted path with the lowest Hausdorff distance from the desired path was selected as the result. The random samples were drawn from the learned latent distribution so that they are valid and introduce diverse outcomes. Among all the $\mathrm{c}-\beta-\mathrm{VAE}$ models, the best result is achieved with $\beta=10$, leading to a 78.9% ratio of good path. An additional test path database of 1000 paths generated by fully random RSCR mechanisms was used to assess the robustness of the best model. Shown in the third row of Table 6, the results of this evaluation indicate that even with randomly generated paths, the best model maintains a high performance and achieves a 70.8% ratio of good paths. Figure 12(a) illustrates a randomly selected result from the test path dataset, and Fig. 12(b) shows part of the result from the additional random test path dataset. Furthermore, Fig. 13 shows nine different predicted paths with their corresponding RSCR mechanisms, all derived from the same input path. The link lengths and rotational axes of these nine mechanisms are listed in Table 7 to illustrate their uniqueness from each other. While some of them, for example the 1st and the 4th mechanisms, are quite similar to each other, the majority display uniqueness. This finding further proves our model's capability to generate multiple, diverse RSCR mechanisms.

In this section, we presented the results from the three different models we explored. To verify the statistical difference between these results, McNemar's tests [50,51] were applied to compare the Hausdorff distances from the different architectures. When the Hausdorff distance is less than 0.027, the prediction is considered correct; otherwise, the prediction is considered wrong. With the use of the best model from each architecture, the p-value obtained between MLP and VAE plus MLP is 0.738, indicating no significant difference between their results. The p-value between $\mathrm{c}-\beta-\mathrm{VAE}$ and MLP is 2.75×10^{-11}, and that between $\mathrm{c}-\beta-\mathrm{VAE}$ and VAE plus MLP is 3.95×10^{-11}. These two p-values indicate are significant differences between the results of $\mathrm{c}-\beta-\mathrm{VAE}$ and the other two architectures. Among all the architectures, $\mathrm{c}-\beta-\mathrm{VAE}$ generally performs better, exhibiting lower losses, a higher closed-path ratio, and a greater satisfactory ratio. When the hyperparameter β is set to 10, $\mathrm{c}-\beta-\mathrm{VAE}$ demonstrates excellent performance across various paths. In addition, the $\mathrm{c}-\beta-\mathrm{VAE}$ architecture can generate multiple solutions for a single input path. Although the results between MLP and VAE plus MLP are similar, the latter still stands out due to its ability to generate multiple solutions.

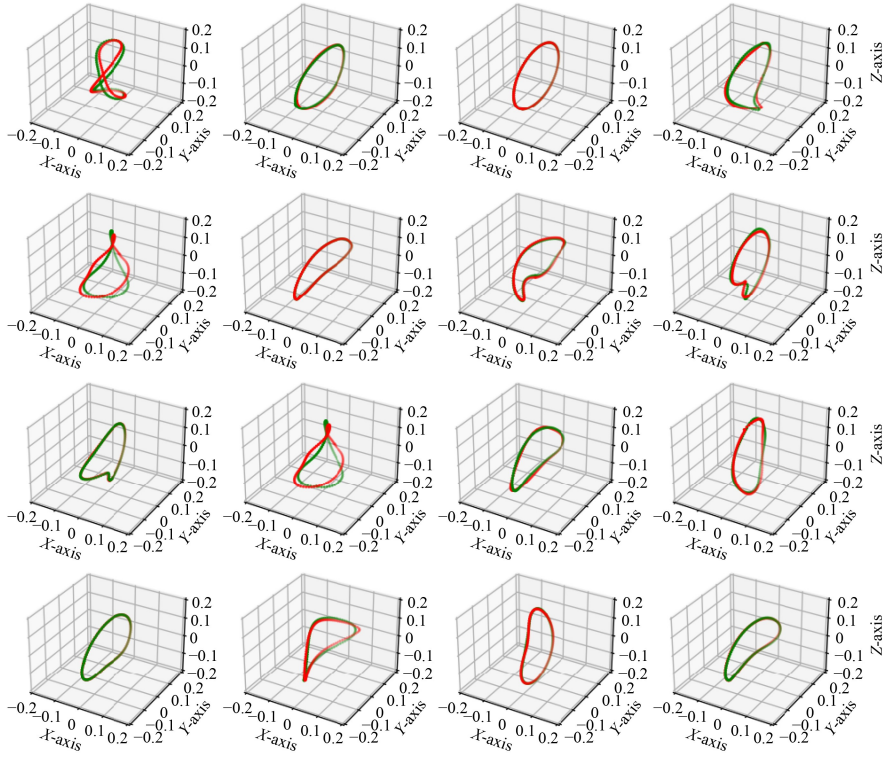
5 Application example

This section presents an example of the practical application of RSCR mechanisms. Zhao et al. [1] collected data on human upper-limb motion and designed a 1-DOF planar four-bar mechanism to approximate this motion. Considering the limitation of planar mechanisms, they projected the spatial upper-limb motion onto a neutral plane. The spatial RSCR mechanism can directly accommodate the spatial motion without the need for projection. By utilizing the original collected spatial data, the RSCR mechanism can provide additional natural solutions. Although this work focuses on path synthesis and only utilizes the path data given by Zhao et al. [1], the RSCR mechanism also holds potential for motion synthesis for future research. The raw path given by Ref. [1] is shown in Fig. 14, where the green path represents the trajectories of the wrist joint and the red path illustrates the trajectory of the elbow joint.

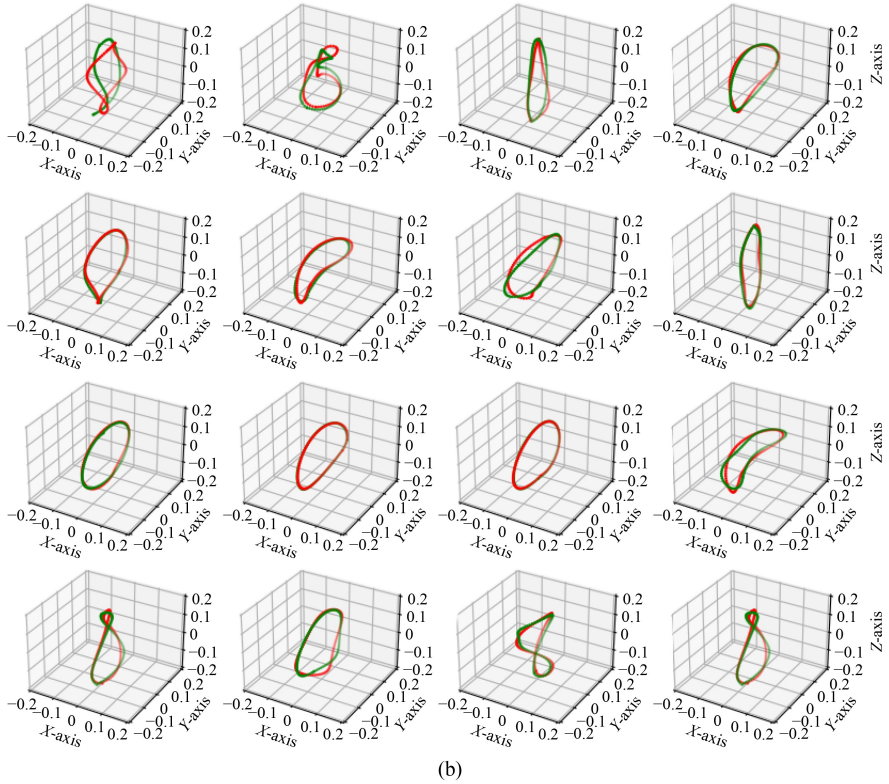
Following our pipeline, the raw trajectories were first converted into B-splines and then normalized. These normalized paths were fed into our optimal model, $\mathrm{c}-\beta-\mathrm{VAE}$ $(\beta=10)$, where they were copied 100 times, and each copy was combined with a random sample from latent space to ensure robust matching. The best matches were selected as the final design. Figure 15 displays the top matches for the

Table 6 $\mathrm{c}-\beta-\mathrm{VAE}$ model comparison (satisfactory: $H<0.027$)

β value	Train loss		Test loss		Test path			
	MSE	KLD	MSE	KLD	Closed path/%	Satisfactory/%	Mean	Median
0	0.0021	0.0107	0.0021	0.0106	100	23.2	0.0453	0.0417
10	0.0102	0.0025	0.0136	0.0025	100	78.9	0.0195	0.0125
10 (fully random path)	–	–	–	–	100	70.8	0.0249	0.0181
25	0.0237	0.0008	0.0136	0.0010	96.6	51.8	0.0527	0.0207
100	0.2426	0.00003	0.2417	0.000003	93.7	28.3	0.0486	0.0409



(a)



(b)

Fig. 12 Predicted paths (green) and input paths (red) from the $\mathbf{c}-\beta-\mathrm{VAE}$ $(\beta=10)$ model. (a) Predicted paths (green) and input paths (red) from the $\mathbf{c}-\beta-\mathrm{VAE}$ $(\beta=10)$ model. The input paths are randomly chosen in the test path dataset. (b) Predicted paths (green) and input paths (red) from the $\mathbf{c}-\beta-\mathrm{VAE}$ $(\beta=10)$ model. The input paths are randomly chosen in the random path dataset.

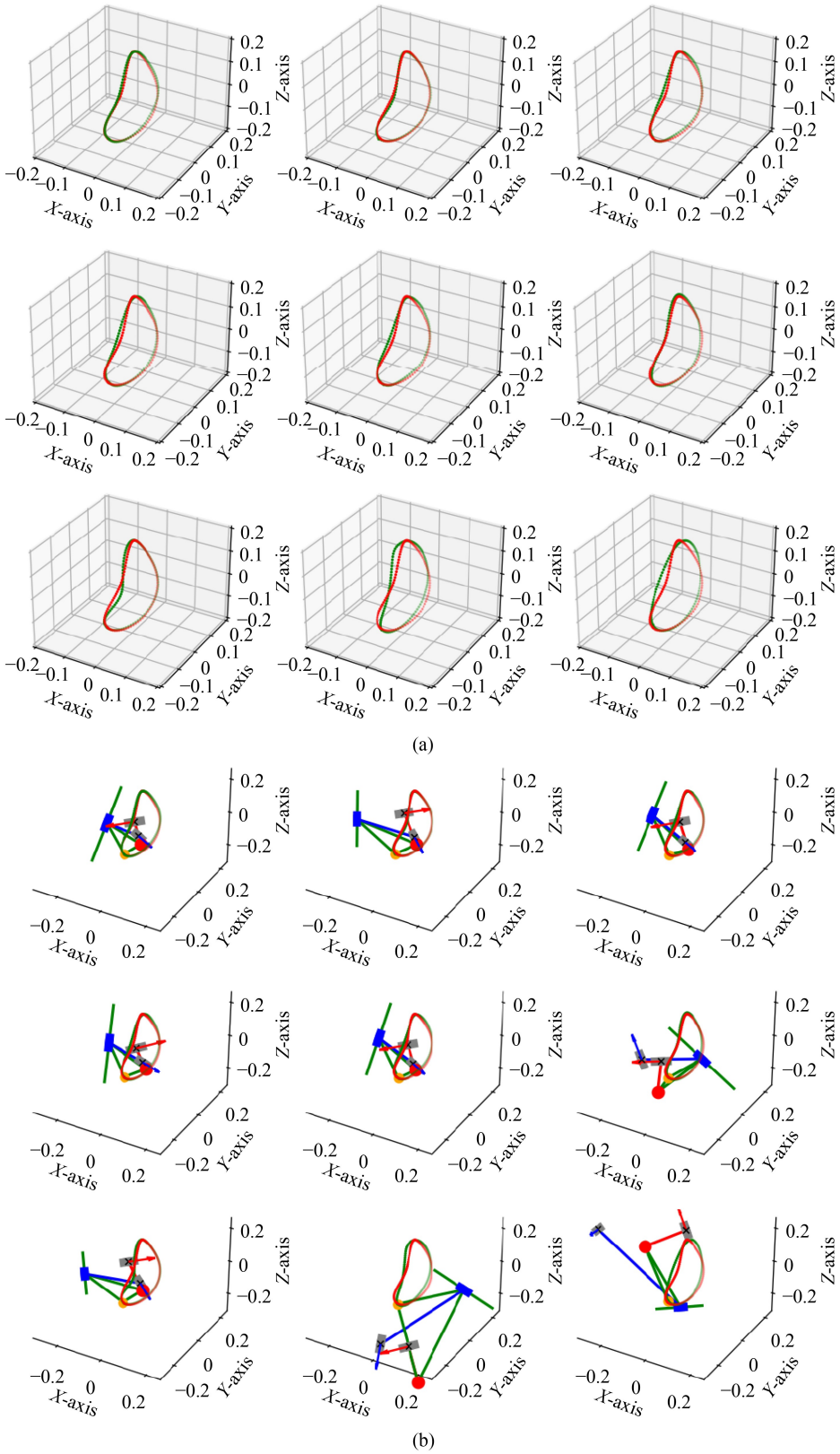


Fig. 13 Predicted paths (green) and their corresponding RSCR mechanisms with the same input path (red) from the $c-\beta-$VAE ($\beta=10$) model. (a) Predicted paths (green) with the same input path (red) from the $c-\beta-$VAE ($\beta=10$) model. (b) Predicted paths (green) and their corresponding RSCR mechanisms with the same input path (red).

Table 7 Comparison of nine different mechanism configurations

No.	l_{12}	l_{13}	l_{34}	l_{25}	$\boldsymbol{u}_2$	$\boldsymbol{u}_5$
1	0.6534	1.0516	1.7850	1.4995	(0.03, −0.88, 0.5)	(0.32, 0.88, 0.31)
2	0.9100	1.1620	1.9591	1.7991	(−0.02, −0.89, 0.5)	(0.84, 0.08, 0.48)
3	0.7523	1.0387	1.6738	1.4410	(−0.02, −0.87, 0.49)	(0.37, 0.88, 0.27)
4	0.6643	0.9774	1.8320	1.5670	(0.03, −0.84, 0.51)	(0.37, 0.87, 0.26)
5	0.7253	0.9682	1.6429	1.4435	(−0.04, −0.86, 0.47)	(0.42, 0.89, 0.26)
6	1.3059	0.7332	1.0906	1.1858	(0.4, −0.85, 0.22)	(0.01, −0.88, 0.47)
7	0.9860	1.2571	2.2178	2.0806	(0.03, −0.88, 0.52)	(0.87, 0, 0.49)
8	1.1896	0.4629	1.5074	1.0204	(−0.06, 0.84, 0.45)	(−0.53, −0.86, 0.27)
9	1.2338	0.6158	1.1143	1.9786	(−0.37, 0.85, 0.27)	(0.38, 0.85, 0.25)

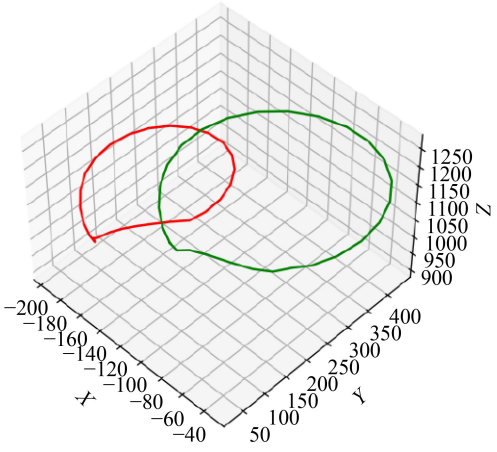


Fig. 14 Raw trajectories of wrist (green) and elbow (red).

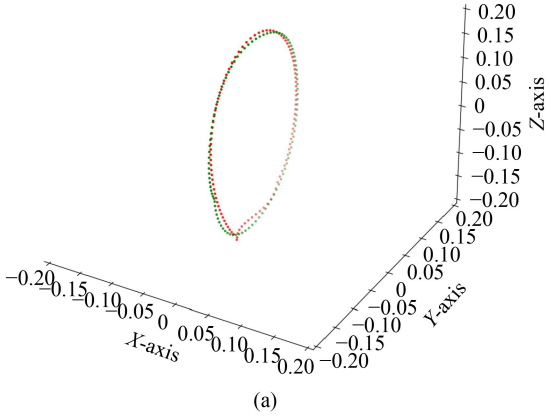
wrist and elbow paths, alongside the RSCR mechanisms that generate these optimal paths. Our model achieves excellent matches, with Hausdorff distances of 0.01031 and 0.0175 for the wrist and elbow paths, respectively. After scaling back, the design parameters for the practical application of the RSCR mechanisms are detailed in Table 8.

6 Conclusions and future work

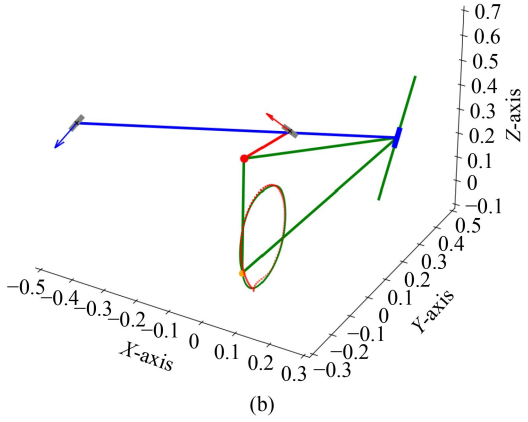
In this work, we detailed a comprehensive process for the path synthesis of spatial RSCR mechanisms using deep learning models. A dedicated simulator was developed to generate a robust database containing over 5 million normalized paths, providing a reliable foundation for deep learning training. We explored three distinct deep learning models: MLPs, VAE plus MLP, and a novel approach using $\mathrm{c}-\beta-\mathrm{VAE}$. Comprehensive evaluations proved $\mathrm{c}-\beta-\mathrm{VAE}$ with $\beta = 10$ to be the most effective model.

We also presented a practical application in human upper-limb rehabilitation, where our trained $\mathrm{c}-\beta-\mathrm{VAE}$ model successfully matched RSCR mechanisms with the wrist and elbow trajectories from real human movements. This demonstration highlighted the potential of RSCR mechanisms and our methods to make significant contributions in real-world scenarios.

Looking forward, we aim to extend these methods to diverse mechanism types and motion synthesis problems, continuing to refine the accuracy and applicability of deep learning models in mechanical design.



(a)



(b)

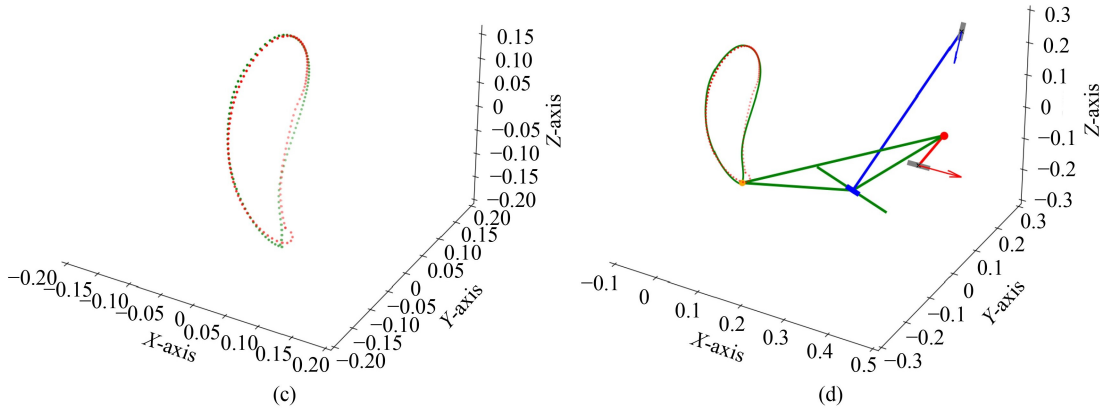


Fig. 15 Best matches for wrist and elbow trajectories found by $\mathrm{c}-\beta-\mathrm{VAE}$ ($\beta = 10$). (a) The best match found by our model (green) with the normalized wrist path (red). The Hausdorff distance between two paths is 0.01031. (b) RSCR mechanism corresponds to the best match path of wrist. (c) The best match found by our model (green) with the normalized elbow path (red). The Hausdorff distance between two paths is 0.0175. (d) RSCR mechanism corresponds to the best match path of elbow.

Table 8 Design parameters of RSCR mechanism for wrist and elbow path synthesis

Parameter	Wrist	Elbow
l_{12}	840.7165	234.5693
l_{13}	176.8662	65.1142
l_{34}	898.8414	158.2612
l_{36}	700.3672	291.2300
l_{25}	939.3346	308.1754
$\boldsymbol{u}_1$	(0, 0, −1.00)	(0, 0, −1.00)
$\boldsymbol{u}_2$	(0.38, −0.89, 0.26)	(0.86, −0.01, 0.51)
$\boldsymbol{u}_5$	(−0.31, 0.90, 0.30)	(0.35, −0.89, 0.28)

Nomenclature

Abbreviations

c−VAE	Conditional variational autoencoder
c−β−VAE	Conditional−β−VAE
DOF	Degree of freedom
FCNN	Fully connected neural network
GAN	Generative adversarial network
ICA	Independent component analysis
KL	Kullback–Leibler
KLD	Kullback–Leibler divergence
MLP	Multi-layer perceptron
MSE	Mean squared error
PCA	Principal component analysis
RC	Revolute–Cylindrical
ReLU	Rectified linear unit
RSCR	Revolute–Spherical–Cylindrical–Revolute
RRRR	Revolute–Revolute–Revolute–Revolute
SS	Spherical–Spherical
VAE	Variational autoencoder

Variables

a	A point in set A
A	Set of points
b	A point in set B
B	Set of points
$d(a,b)$	Euclidean distance between point a and point b
$\mathbf{e}_{\boldsymbol{J}}$	Vector that represents mechanism parameters after the original mechanism parameters encoded by MLP
$\mathbf{e}_{\boldsymbol{P}}$	Vector that represents path coordinates after the original path coordinates encoded by MLP
F	Object function for the simulator
$h(A,B)$	Maximum distance of set A to the nearest point in set B
$h(B,A)$	Maximum distance of set B to the nearest point in set A
$H(A,B)$	Hausdorff distance between sets A and B. It is defined by the maximum distance of a set to the nearest point in the other set
$\boldsymbol{J}$	Vector that represents mechanism parameters
$\hat{\boldsymbol{J}}$	Vector that represents the predicted mechanism parameters
$\boldsymbol{K}$	Vector represents Key in attention structure
l_{ik}	Link that connects joint J_i and J_k
$\boldsymbol{P}$	Vector that represents the path coordinates
P_{ix}	x-coordinates of the ith point in a path
P_{iy}	y-coordinates of the ith point in a path
P_{iz}	z-coordinates of the ith point in a path

$\boldsymbol{P}(\boldsymbol{x})$	Probability distribution of the input data $\boldsymbol{x}$
$\boldsymbol{P}(\boldsymbol{x}\|\boldsymbol{z})$	Probability distribution of the input data $\boldsymbol{x}$ with the condition of $\boldsymbol{z}$
$\boldsymbol{P}(\boldsymbol{z})$	Probability distribution of the latent space data $\boldsymbol{z}$
$q(\boldsymbol{z}\|\boldsymbol{x})$	Probability distribution of the latent space data $\boldsymbol{z}$ with the condition of the input data $\boldsymbol{x}$
$\boldsymbol{Q}$	Vector represents Query in attention structure
$\boldsymbol{V}$	Vector represents Value in attention structure
$\hat{\boldsymbol{x}}$	Reconstructed vector from the original input $\boldsymbol{x}$
$\boldsymbol{z}$	Vector represents the latent space of VAE
β	Hyper parameter to adjust the weight of loss function in a c$-\beta-$VAE architecture
$\mu(\boldsymbol{x}_2)$	Mean of $\boldsymbol{x}_2$
$\boldsymbol{\mu}$	Vector that represents the means of normal distributions
ϕ	Input angle of a RSCR mechanism from the actuator
$\mathcal{N}(0, \mathbf{I})$	Standard normal distribution with mean of 0 and standard deviation of $\mathbf{I}$
$\sigma_i(\boldsymbol{x}_2)$	Standard deviation of $\boldsymbol{x}_2$
σ	Vector that represents the standard deviations of normal distributions
ε	Random sample from $\mathcal{N}(0, \mathbf{I})$

Acknowledgements This work was financially supported by the National Science Foundation of USA (under Grant No. STTR phase II \#2126882) and the co-author/co-PI Dr. Purwar, who also holds stocks in Mechanismic Inc., USA. The research findings in this publication may or may not necessarily relate to the interests of Mechanismic Inc. The terms of this arrangement have been reviewed and approved by Stony Brook University, USA in accordance with its policy on objectivity in research. All findings and results presented in this paper are those of the authors and do not represent those of the funding agencies.

Conflict of Interest The authors declare no conflict of interest.

References

1. Zhao P, Zhang Y T, Guan H W, Deng X T, Chen H D. Design of a single-degree-of-freedom immersive rehabilitation device for clustered upper-limb motion. Journal of Mechanisms and Robotics, 2021, 13(3): 031006
2. Song W, Zhao P, Li X, Deng X T, Zi B. Data-driven design of a six-bar lower-limb rehabilitation mechanism based on gait trajectory prediction. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2023, 31: 109–118
3. Zhang Y, Deng X T, Zhou B, Zhao P. Design and optimization of a multi-mode single-DOF watt-I six-bar mechanism with one adjustable parameter. In: Proceeding of Advances in Mechanism, Machine Science and Engineering in China. Singapore: Springer, 2022
4. Deng X T, Purwar A. A matrix-based approach to unified synthesis of planar four-bar mechanisms for motion generation with position, velocity, and acceleration constraints. ASME Journal of Computing and Information Science in Engineering, 2024, 24(12): 121003
5. Watanabe K, Sekine T, Nango J. Kinematic analysis and branch identification of RSCR spatial four link mechanisms. JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing, 1998, 41(3): 450–459
6. Thompson J M. Computer aided design and synthesis of the RSCR spatial mechanism. Thesis for the Master's Degree. Blacksburg: Virginia Polytechnic Institute and State University, 1987
7. Bagci C. The RSRC space mechanism – analysis by 3 × 3 screw matrix, synthesis for screw generation by variational methods. Dissertation for the Doctoral Degree. Oklahoma State University, 1969
8. Chiang C H, Chieng W H, Hoeltzel D A. Synthesis of the RSCR mechanism for four precision positions with relaxed specifications. Mechanism and Machine Theory, 1992, 27(2): 157–167
9. Ananthasuresh G K, Kramer S N. Analysis and optimal synthesis of the RSCR spatial mechanisms. Journal of Mechanical Design, 1994, 116(1): 174–181
10. Shrivastava A K, Hunt K H. Dwell motion from spatial linkages. Journal of Engineering for Industry, 1973, 95(2): 511–518
11. Wang X, Guo W Z. The design of looped-synchronous mechanism with duplicated spatial assur-groups. Journal of Mechanisms and Robotics, 2019, 11(4): 041014
12. Osman M O M, Segev D. Kinematic analysis of spatial mechanisms by means of constant distance equations. Transactions of the Canadian Society for Mechanical Engineering, 1972, 1(3): 129–134
13. Osman M O M, Bahgat B M, Dukkipati R V. Kinematic analysis of spatial mechanisms using train components. Journal of Mechanical Design, 1981, 103(4): 823–830
14. Huang T C, Youm Y. Exact displacement analysis of

four-link spatial mechanisms by the direction cosine matrix method. Journal of Applied Mechanics, 1984, 51(4): 921–928
15. Regenwetter L, Nobari A H, Ahmed F. Deep generative models in engineering design: A review. Journal of Mechanical Design, 2022, 144(7): 071704
16. Vasiliu A, Yannou B. Dimensional synthesis of planar mechanisms using neural networks: application to path generator linkages. Mechanism and Machine Theory, 2001, 36(2): 299–310
17. Galán-Marín G, Alonso F J, Del Castillo J M. Shape optimization for path synthesis of crank-rocker mechanisms using a wavelet-based neural network. Mechanism and Machine Theory, 2009, 44(6): 1132–1143
18. Chui C K. An Introduction to Wavelets. San Diego: Academic Press, 1992
19. Deshpande S, Purwar A. A machine learning approach to kinematic synthesis of defect-free planar four-bar linkages. Journal of Computing and Information Science in Engineering, 2019, 19(2): 021004
20. Deshpande S, Purwar A. Computational creativity via assisted variational synthesis of mechanisms using deep generative models. Journal of Mechanical Design, 2019, 141(12): 121402
21. Sharma S, Purwar A. A machine learning approach to solve the alt–burmester problem for synthesis of defect-free spatial mechanisms. Journal of Computing and Information Science in Engineering, 2022, 22(2): 021003
22. Nurizada A, Purwar A. An invariant representation of coupler curves using a variational autoencoder: Application to path synthesis of four-bar mechanisms. Journal of Computing and Information Science in Engineering, 2024, 24(1): 011008
23. Lee S, Kim J, Kang N. Deep generative model-based synthesis of four-bar linkage mechanisms considering both kinematic and dynamic conditions. In: Proceedings of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Boston: ASME, 2023, V03AT03A016
24. Purwar A, Chakraborty N. Deep learning-driven design of robot mechanisms. Journal of Computing and Information Science in Engineering, 2023, 23(6): 060811
25. Nobari A H, Srivastava A, Gutfreund D, Xu K, Ahmed F. LInK: learning joint representations of design and performance spaces through contrastive learning for mechanism synthesis. 2024, arXiv preprint arXiv:2405.20592
26. Yim N H, Ryu J, Kim Y Y. Big data approach for synthesizing a spatial linkage mechanism. In: Proceedings of IEEE International Conference on Robotics and Automation. London: IEEE, 2023, 7433–7439
27. Kingma D P, Welling M. Auto-encoding variational Bayes. 2013, arXiv preprint arXiv:1312.6114
28. Nurizada A, Dhaipule, R, Lyu Z, Purwar A. A dataset of 3M single-DOF planar 4-, 6-, and 8-bar linkage mechanisms with open and closed coupler curves for machine learning-driven path synthesis. ASME Journal of Mechanical Design, 2025, 147(4): 041702
29. Nurizada A, Lyu Z, Purwar A. Path generative model based on conditional β-variational auto encoder for four-bar mechanism design. Journal of Mechanisms and Robotics, 2025, 17(6): 061004
30. Deng X, Nurizada A, Purwar A. Synthesizing spatial RSCR mechanisms for path generation using a deep neural network. In: Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. ASME, 2024
31. Javier G d J, Eduardo B. Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge. New York: Springer, 2011, 72–74
32. Virtanen P, Gommers R, Oliphant T E, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt S J, Brett M, Wilson J, Millman K J, Mayorov N, Nelson A R J, Jones E, Kern R, Larson E, Carey C J, Polat İ, Feng Y, Moore E W, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero E A, Harris C R, Archibald A M, Ribeiro A H, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 2020, 17: 261–272
33. Chase T R, Mirth J A. Circuits and branches of single-degree-of-freedom planar linkages. Journal of Mechanical Design, 1993, 115(2): 223–230
34. Piegl L, Tiller W. The NURBS Book. 2nd ed. Berlin: Springer, 1997
35. Lyu Z J, Purwar A. Design and development of a sit-to-stand device using a variational autoencoder-based deep neural network. In: Proceedings of ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. St. Louis: ASME, 2022, V007T07A027
36. Jolliffe I T. Principal Component Analysis. 2nd ed. New York: Springer, 2002
37. Yu S C, Chang Y, Lee J J. A generative model for path synthesis of four-bar linkages via uniform sampling dataset. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2023, 237(4): 811–829
38. Sener S, Unel M. Geometric invariant curve and surface normalization. In: Campilho A, Kamel M, eds. Image Analysis and Recognition. Berlin: Springer, 2006, 445–456
39. Purwarlab. RSCR Mechanisms. 2025, available at www.kaggle.com/datasets/purwarlab/rscr-mechanisms website
40. Haykin S. Neural Networks: A Comprehensive Foundation. 2nd ed. Upper Saddle River: Prentice Hall, 1998

41. Agarap A F. Deep learning using rectified linear units (ReLU). 2019, arXiv preprint arXiv:1803.08375
42. Ba J L, Kiros J R, Hinton G E. Layer normalization. 2016, arXiv preprint arXiv:1607.06450
43. Srivastava N, Hinton G , Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 2014, 15(1): 1929–1958
44. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Red Hook, NY: Curran Associates Inc., 2017, 6000–6010
45. Han K, Wang Y H, Chen H T, Chen X H, Guo J Y, Liu Z H, Tang Y H, Xiao A, Xu C J, Xu Y X, Yang Z H, Zhang Y M, Tao D C. A survey on vision transformer. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(1): 87–110
46. Sutton R S, Barto A G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge: MIT Press, 2018
47. Sohn K, Lee H, Yan X C. Learning structured output representation using deep conditional generative models. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2015, 3483–3491
48. Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A. β-VAE: Learning basic visual concepts with a constrained variational framework. In: Proceedings of International Conference on Learning Representations (ICLR 2017). 2017
49. Rote G. Computing the minimum Hausdorff distance between two point sets on a line under translation. Information Processing Letters, 1991, 38(3): 123–127
50. McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika, 1947, 12(2): 153–157
51. Raschka S. Model evaluation, model selection, and algorithm selection in machine learning. 2020, arXiv preprint arXiv:1811.12808