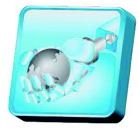
ASME Journal of Mechanisms and Robotics
Online journal at:
https://asmedigitalcollection.asme.org/mechanismsrobotics



Generative Design of Planar Four-Bar Motions

Amro Halwah
Department of Mechanical Engineering,
Computer-Aided Design and Innovation Lab,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: amro.halwah@stonybrook.edu

Anurag Purwar[1]
Department of Mechanical Engineering,
Computer-Aided Design and Innovation Lab,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: anurag.purwar@stonybrook.edu

This article introduces a novel deep generative approach to the approximate motion synthesis of planar four-bar mechanisms. Existing motion synthesis approaches for synthesizing four-bar mechanisms have generally focused on representing the motion as five or less discrete poses while not providing any guarantees on the practicality of the solutions, such as ones without circuit-, branch-, and order defects. This article presents a conditional variational autoencoder (C-VAE)-based approach for synthesizing practical mechanisms that can capture the overall motion requirement while generating a continuous motion approximately. The mathematical machinery in our work uses quaternions and kinematic mapping to represent the geometric constraints of dyads of four-bar mechanism as algebraic constraint manifolds in a three-dimensional projective space. The parameters describing the manifolds encode information about the type and dimensions of constituent dyads of four-bar mechanisms. These parameters are used as input to a C-VAE that uses an arbitrarily large number of input poses approximating a motion as the condition. A procedure is presented for normalizing input motions such that they are invariant with respect to rotation, scale, translation, and reflection. The trained model can generate several pairs of constraint manifolds from which four-bar mechanisms with different parameters can be retrieved. These mechanisms' coupler poses approximate the input's coupler poses to varying degrees of error. A method for filtering mechanisms by the quality of match of their output motion to the input motion is also proposed. Different metrics for quantifying the similarity of an output mechanism's motion with the input motion are presented for path and orientation separately. [DOI: 10.1115/1.4070416]

Keywords: planar four-bar linkages, motion synthesis, conditional variational autoencoder, machine learning, application of machine learning, mechanism synthesis and analysis

1 Introduction

Synthesis of planar four-bar mechanisms where the coupler can go through a set of given poses[2] is known as a motion synthesis problem. An overview of the development of mechanisms for motion generation is provided by Kerle et al. [1]. Despite extensive literature on four-bar and higher-order linkages [2–7], this problem is often narrowly defined in classical kinematic synthesis as one where the mechanism type, typically a four-bar with revolute joints, is preselected, and the motion is described by a limited number of discrete poses, usually no more than five [8]. These assumptions reduce the problem to one-dimensional synthesis and permit exact solutions. However, when more than five poses are specified, exact solutions no longer exist, and optimization methods are typically employed to find mechanism parameters that approximate the desired motion [9,10]. Most classical approaches overlook critical practical issues such as branch and circuit defects, common in single-degree-of-freedom mechanisms, as discussed by Chase and Mirth [11].

Ge et al. [12] introduced a novel approach to simultaneous type and dimensional synthesis of four-bar mechanisms using algebraic distance for the fitting of constraint manifolds associated with planar four-bar linkages. The key idea behind this method is to create a unified design equation for all types of constituent dyads of four-bar mechanisms, and then employ a two-step algorithm to find the dyad type and dimensions. The two-step algorithm greatly simplifies computation by first solving for eigen vectors of a set of linear equations using singular value decomposition (SVD) and then finding roots of a quartic polynomial. Zhao et al. [13], Li et al. [14], and Ge et al. [15] extended this approach to include six-bar, spherical, and spatial platform mechanisms. Prior to Ge's work, Hayes and Zsombor-Murrary [16] presented a similar method, but did not identify the fundamental constraint equations necessary for type synthesis. All of these methods use quaternions and kinematic mapping to find a unified representation of geometric constraints of various dyad types. Kinematic mapping-based motion synthesis approaches were originally pioneered by Ravani and Roth [17,18] and McCarthy's group [19,20].

Other approaches taken by researchers in the past have been based on analytical, graphical, or optimization techniques to find mechanisms for the motion synthesis problem. Levitskii [21] presented one of the first works in this area, wherein he used

[1]Corresponding author.
[2]In this article, a pose refers to both the position and the orientation of a moving object.

Contributed by the Mechanisms and Robotics Committee of ASME for publication in the Journal of Mechanisms and Robotics. Manuscript received April 5, 2025; final manuscript received November 8, 2025; published online December 16, 2025. Assoc. Editor: Ashis G. Banerjee.

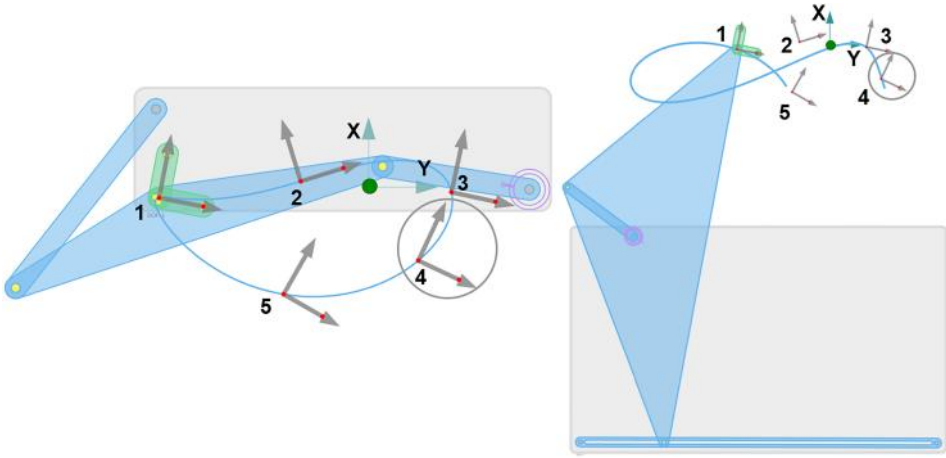


Fig. 1 Results of motion synthesis produced using MotionGen are displayed, with the initial five poses and the corresponding calculated mechanism on the left. On the right, the outcome from a minor adjustment to the (*x*, *y*) position of the circled pose by (−1.39, +0.08) is shown. Clearly, the mechanisms resulting from these inputs differ drastically with the first being a defect-free Grashof RRRR, and the second a non-Grashof RRRP with a circuit defect.

optimization to achieve an approximate solution. Modler and Lin [22] utilized geometric transformations to synthesize planar four-bar mechanisms for any unsharpened three-position problem, and showed how geometrical similarity transformation can be used to synthesize multiple mechanisms for the same motion generation task [23]. Zimmerman [24] presented a graphical method, using poles and rotation angles, for the exact synthesis of planar four-bar mechanisms. Yao and Angeles [25] formulated the motion synthesis problem as an optimization task using a least-squares criterion to find dyads that best match a given set of poses. Chen et al. [26] provide a comprehensive solution to the Burmester problem, integrating all four dyad types with any combination of prismatic (P) and revolute (R) joints and treating PR and RP dyads as RR dyads with one joint at infinity. Al-Widyan et al. [27] further refined the Burmester problem solution by enhancing numerical stability, addressing issues of ill-conditioning in traditional methods through least-square filtering and data-conditioning techniques. Simionescu [28] presents constraint equations for the synthesis of planar four-bar mechanisms from two and three prescribed positions, and provides optimization methods for finding the best transmission angles. Cervantes-Sánchez et al. [29] presented a vector-based method for the exact motion synthesis of four-bar linkages with prismatic and revolute joints, challenging previous claims that PR and RP dyads can achieve five specified poses and asserting that only four can be reached exactly. Most recently, Xu et al. [30] have introduced a bi-invariant approach using pole locations for the approximate motion synthesis of planar four-bar mechanisms.

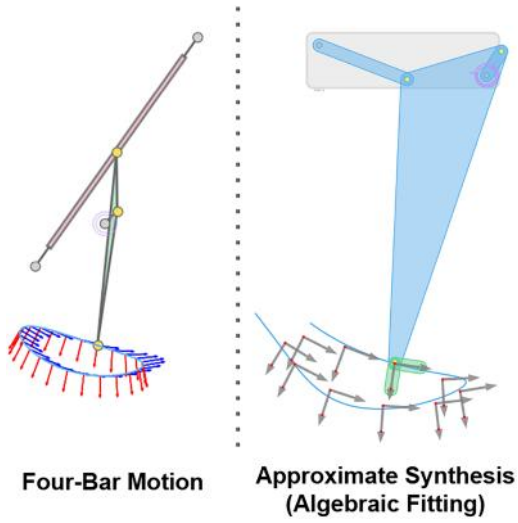


Fig. 2 Given a four-bar motion, shown on the left, ten poses are sampled and used as input for the motion synthesis algorithm [34] in MotionGen [32]. The resulting mechanism, shown on the right, does not come close to approximating the input motion.

Despite such advances, these methods have several shortcomings: (1) motion is still required to be represented as a set of few discrete poses and (2) solutions suffer from order, circuit, and branch defects [11] and are limited in number (at most four dyads). Restriction on the number and placement of poses forces mechanism designers to make highly restrictive choices about the relative importance of the poses. Typically, synthesis produces two dyads independently satisfying motion requirements, but when assembled together result in defects. In addition to that, the nonlinear relationship between poses and the mechanisms also presents a problem—minor modifications to inputs can lead to drastically different outcomes or sometimes no viable solutions. To illustrate this unpredictable behavior, we employ the motion synthesis function in MotionGen [31,32], which implements the algebraic fitting algorithm presented in Ref. [12]. Figure 1 shows how a slight variation in one of the five input poses can change the outcomes dramatically. A slight translation of the circled pose yields a crank-slider mechanism on the right side that is markedly distinct from an all revolute-jointed mechanism on the left. Moreover, the motions produced by both mechanisms are also very different from each other, and the new mechanism is a non-Grashof [33] one with a circuit defect, wherein three poses are on one circuit, while the other two poses are on the other circuit. Furthermore, for more than five poses, the generated mechanisms approximate the input poorly. For example, Fig. 2 shows ten poses sampled from an input motion of 360 poses and used as an input to the algebraic motion synthesis algorithm [12]. It can be seen that the outcome is neither the original mechanism nor an approximately similar motion.

In an attempt to tackle the circuit-defect issues, Deshpande and Purwar [34] proposed an autoencoder neural network (NN) to create a database of motions represented with discrete signatures.

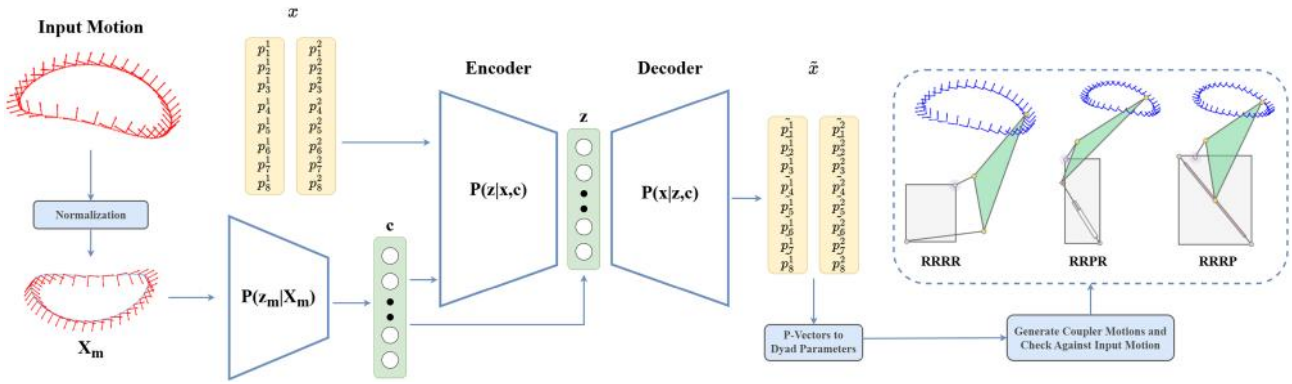


Fig. 3 This overview figure illustrates our proposed pipeline for the motion generation of planar four-bar mechanisms using a VAE for motion representation and a C-VAE for mechanism generation. The VAE is trained on normalized four-bar motions to encode the motion, X_m, to a lower dimensional space representation, c. The C-VAE model is trained on pairs of p vectors, x, representing mechanisms that have closed coupler motions, conditioned on the motion's latent representation, c. The condition is concatenated with a randomized latent vector, z, and passed through the decoder. This generates a pair of p vectors, $\tilde{x}$, from which dyad parameters can be computed to construct the four-bar mechanism. The coupler motions of generated mechanisms are computed and compared against the input motion using the L_2 norm of the difference between input path Fourier descriptors and output path Fourier descriptors. The same computation is also repeated for orientation, and the two L_2 norm values are used for computing motion similarity.

The reduced dimensionality, achieved by the autoencoder, coupled with a standard clustering algorithm allowed for a k-nearest neighbor search that returns the motion with the highest similarity score, along with its linkage parameters. In a subsequent work [35], they proposed a variational autoencoder (VAE) to learn the probability distribution of coupler motions, which can include incomplete information. It generates samples that mirror important features of the input, feeding these into existing computational synthesis methods. These approaches largely focused on motion representation followed by a database lookup for solutions rather than true generative design of mechanisms. Purwar and Ge [36] have presented a review of data-driven approaches for the kinematic synthesis of mechanisms, which includes a discussion of the aforementioned algebraic fitting algorithm as well as machine learning-driven approaches.

In this article, we present a novel generative design method based on conditional-variational autoencoder (C-VAE) NN to perform simultaneous type and dimensional synthesis of four-bar mechanisms for an arbitrary motion described by n-poses, where n can be much larger than the magic number five. The goal is to overcome the aforementioned limitations of existing synthesis approaches and produce a large number of defect-free and stable solutions without explicitly formulating and coding fundamental constraints needed to identify dyad types [37]. While the exact details of the method are presented later in the article, for now it suffices to look at Eq. (5), which presents a unified design equation representing the geometric constraints of dyads, and Eq. (6), which presents two quadratic fundamental constraints that relate dyad parameters p_i. These dyad parameters, although a redundant representation, encode information about *both* the type and dimensions of dyads.

We hypothesize that (1) these constraints could be learnt by a NN without explicitly specifying it and (2) a database of compatible dyads (that do not produce defects when put together) could be utilized for training purposes with the motion as a condition in a C-VAE. This approach has the potential to scale to higher-order platform mechanisms (planar, spherical, and spatial), where identifying fundamental constraints could be mathematically daunting due to a large number of parameters like p_i and nonlinearity in their relationship.

Figure 3 shows an overview of our proposed approach. We use our simulation algorithm [38] to generate a dataset of the three most common types of four-bar mechanisms and their motions represented as 360 poses. Motions are normalized to be invariant to affine transformations (rotation, translation, uniform scaling, and mirroring) and used to train a vanilla VAE. The latent representation of the VAE is used as the condition for a C-VAE to synthesize defect-free planar four-bar mechanisms. The C-VAE model is trained on a unified representation for four-bar planar dyads [37] called $\mathbf{p} = (p_1 \dots p_8)$ vectors. Specifically, the parameters of each mechanism are represented by a pair of $\mathbf{p}$ vectors, which are concatenated and used as input to a C-VAE. The generated mechanisms are filtered by the closeness of the match between their output motion and the given motion. To filter mechanisms based on the accuracy of the motion match, we calculate the Fourier descriptors (FDs) for the path and orientations separately. The L_2 norms of FD for path and orientation are used to quantify path and orientation matches. The L_2 norm of the joint location difference is used to quantify mechanism diversity.

The dataset, which contains 3,253,314 planar four-bar mechanisms, consists of three types of four-bar mechanisms (RRRR, RRRP, and RRPR), where R and P stand for revolute and prismatic joints, respectively. Once the training is complete, this method can produce mechanisms of varying parameters and types that closely approximate the given motion instantaneously. Since a $\mathbf{p}$ vector encodes the type of joints as well as the position of joints for a planar dyad [39], we show that the unified representation of a planar dyad is sufficient for representing both the mechanism type and parameters in deep generative models.

The rest of the article is organized as follows. Section 2 reviews the image space parameterization of a planar displacement using quaternions [18]. Section 3 reviews the generalized (G-) manifold of planar dyads, which represents geometric constraints of dyads in a unified form. Section 4 presents the VAE and C-VAE architectures along with the loss functions used for training. Section 5 presents a procedure for generating a dataset of planar four-bar mechanisms with closed coupler motions and the preparation of data for model training. Section 6 discusses the metrics used to quantify motion similarity and the conditions by which redundant mechanisms are filtered out. Section 7 presents the results of generated mechanisms using our proposed approach for motion generation, and finally, Sec. 8 compares the performance to the algebraic fitting algorithm in producing defect-free planar four-bar mechanisms.

2 Image Space Parameterization of Planar Displacement

A planar displacement is defined in the Cartesian space by a translation (d_1, d_2) and a rotation angle ϕ of a rigid body from a

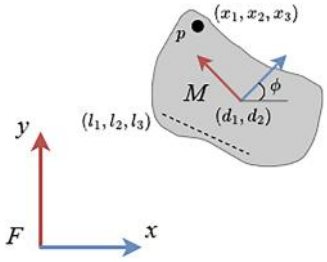


Fig. 4 A planar displacement of a rigid body in Cartesian space by a translation (d_1, d_2) and a rotation angle ϕ from a moving frame M to a fixed frame F. A point p on the rigid body has the homogeneous coordinates (x_1, x_2, x_3) and a line has the coordinates (l_1, l_2, l_3), all in the moving reference frame.

moving frame M to a fixed frame F, as shown in Fig. 4. The Cartesian space parameters (d_1, d_2, ϕ) can be represented in the image space by a planar quaternion $\mathbf{Z} = (Z_1, Z_2, Z_3, Z_4)$ [17,19] as

$$Z_1 = \frac{1}{2}\left(d_1 \cos\frac{\phi}{2} + d_2 \sin\frac{\phi}{2}\right), \quad Z_2 = \frac{1}{2}\left(-d_1 \sin\frac{\phi}{2} + d_2 \cos\frac{\phi}{2}\right)$$
$$Z_3 = \sin\frac{\phi}{2}, \quad Z_4 = \cos\frac{\phi}{2} \tag{1}$$

A planar displacement of point $\mathbf{x} = (x_1, x_2, x_3)$ from frame M to $\mathbf{X} = (X_1, X_2, X_3)$ in frame F can be represented by a homogeneous transform $[H]$ such that $\mathbf{X} = [H]\mathbf{x}$. Similarly, the planar displacement of a line $\mathbf{l} = (l_1, l_2, l_3)$ from frame M to $\mathbf{L} = (L_1, L_2, L_3)$ in frame F can be represented by a homogeneous transform $[\overline{H}]$ such that $\mathbf{L} = [\overline{H}]\mathbf{l}$. These homogeneous transforms can be given in terms of image space coordinates $\mathbf{Z} = (Z_1, Z_2, Z_3, Z_4)$

$$[H] = \begin{bmatrix} Z_4^2 - Z_3^2 & -2Z_3Z_4 & 2(Z_1Z_3 + Z_2Z_4) \\ 2Z_3Z_4 & Z_4^2 - Z_3^2 & 2(Z_2Z_3 - Z_1Z_4) \\ 0 & 0 & Z_3^2 + Z_4^2 \end{bmatrix} \tag{2}$$

$$[\overline{H}] = \begin{bmatrix} Z_4^2 - Z_3^2 & -2Z_3Z_4 & 0 \\ 2Z_3Z_4 & Z_4^2 - Z_3^2 & 0 \\ 2(Z_1Z_3 - Z_2Z_4) & 2(Z_2Z_3 + Z_1Z_4) & Z_3^2 + Z_4^2 \end{bmatrix} \tag{3}$$

where $Z_3^2 + Z_4^2 = 1$.

3 Generalized (G-) Constraint Manifold

In this section, we review the unified form of geometric constraints in the image space for RR, RP, and PR dyads. Each of these dyads has a different geometric constraint shown in Fig. 5. For an RR dyad, the constraint is that a point stays on a circle with its center and radius given by the homogeneous coordinates (a_0, a_1, a_2, a_3). For an PR dyad, a point stays on a line with the coordinate (L_1, L_2, L_3). For a RP dyad, a moving line (l_1, l_2, l_3) remains tangent to a circle with its center (a_1, a_2, a_3).

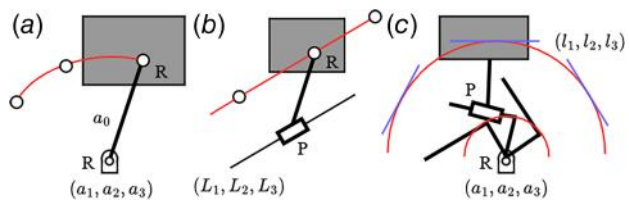


Fig. 5 Geometric constraints of (*a*) RR dyad, (*b*) PR dyad, and (*c*) RP dyad

It can be shown that all of these constraints (point on a circle, point on a line, and a line tangent to a circle) [12], when written using homogeneous transforms, reduce to a single quadratic equation given by

$$\begin{aligned} &-a_0(Z_1^2 + Z_2^2) + a_0x(Z_1Z_3 - Z_2Z_4) + a_0x_2(Z_2Z_3 + Z_1Z_4) \\ &+ a_1(Z_1Z_3 + Z_2Z_4) + a_2(Z_2Z_3 - Z_1Z_4) \\ &+ (a_2x_1 - a_1x_2)Z_3Z_4 - \frac{1}{2}(a_1x_1 + a_2x_2)(Z_3^2 - Z_4^2) \\ &+ \frac{1}{4}(a_3 - a_0x_1^2 - a_0x_2^2)(Z_3^2 + Z_4^2) = 0 \end{aligned} \tag{4}$$

This quadratic equation can be rewritten in terms of intermediate parameters p_i, $i = 1 \ldots 8$, as

$$\begin{aligned} &p_1(Z_1^2 + Z_2^2) + p_2(Z_1Z_3 - Z_2Z_4) + p_3(Z_2Z_3 + Z_1Z_4) \\ &+ p_4(Z_1Z_3 + Z_2Z_4) + p_5(Z_2Z_3 - Z_1Z_4) \\ &+ p_6Z_3Z_4 + p_7(Z_3^2 - Z_4^2) + p_8(Z_3^2 + Z_4^2) = 0 \end{aligned} \tag{5}$$

where the parameters are not independent and must satisfy the following relations:

$$\begin{aligned} p_1p_6 + p_2p_5 - p_3p_4 &= 0 \\ 2p_1p_7 - p_2p_4 - p_3p_5 &= 0 \end{aligned} \tag{6}$$

These eight parameters (or coefficients) can be referred to as a $\mathbf{p}$ vector and are related to the geometric parameters of a dyad by

$$\begin{aligned} p_1 &= a_0, & & \\ p_2 &= a_0x_1, & p_3 &= a_0x_2 \\ p_4 &= a_1, & p_5 &= a_2 \\ p_6 &= a_2x_1 - a_1x_2, & p_7 &= -(a_1x_1 + a_2x_2)/2 \\ p_8 &= (a_3 - a_0(x_1^2 + x_2^2))/4 & & \end{aligned} \tag{7}$$

where (a_0, a_1, a_2, a_3) are the homogeneous coordinates of the center of a circle in a fixed frame and a_0 is the homogenizing factor.

The geometric parameters of a dyad can be retrieved from a $\mathbf{p}$ vector using the following inverse relations:

$$\begin{aligned} l_1 : l_2 : l_3 &= p_2 : p_3 : 2p_8 \\ a_0 : a_1 : a_2 &= (p_2^2 + p_3^2) : (-p_3p_6 - 2p_2p_7) : (p_2p_6 - 2p_3p_7) \\ x_1 : x_2 : x_3 &= (p_6p_5 - 2p_7p_4) : (-p_6p_4 - 2p_7p_5) : (p_5^2 + p_4^2) \end{aligned} \tag{8}$$

Given a set of n poses in terms of Cartesian parameters (d_1, d_2, ϕ), we can write n linear equations using Eq. (5) in terms of unknown p_i. The resulting system of equations can be solved using SVD and candidate solutions substituted in the two quadratic constraints of Eq. (6), which produces up to four dyadic solutions. The dyad type can be determined by simply inspecting the signature of a dyad, which is the pattern of zeros in the $\mathbf{p}$ vector. For a PR dyad, $p_1 = p_2 = p_3 = 0$, and for a RP dyad, $p_1 = p_4 = p_5 = 0$. Thus, using Eq. (8) and the dyad's signature, we can find a pair of dyads, which when assembled together result in four-bar mechanisms. We note that this process is entirely driven by the given data without making any assumptions on the type of dyads.

4 Machine Learning Architectures for Representing Motions, Mechanisms, and Their Mappings

In this section, we present the machine learning (ML) architectures used in our method. First, we discuss the Kolmogorov–Arnold network (KAN) as a replacement for the multilayer perceptron (MLP) in our generative models. Then, we review VAE and present our proposed architecture of a VAE for four-bar motions. Finally, we review C-VAE and present its architecture and how it produces $\mathbf{p}$ vectors.

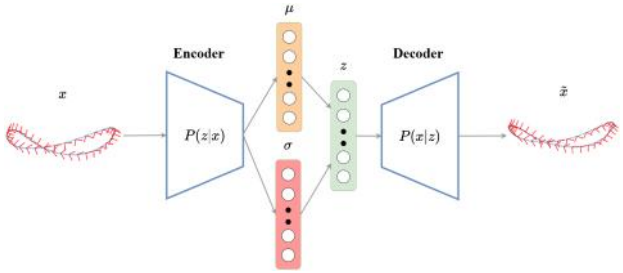


Fig. 6 In a VAE, input data *x* from a distribution *X* are passed through the encoder which outputs two vectors: the mean (μ) and the standard deviation (σ) for a Gaussian distribution in the latent space. A latent variable *z* is then generated using the reparameterization trick. This *z* is passed through the decoder to generate an output $\tilde{x}$ similar to the input. During training, the model tries to minimize both the reconstruction error between $\tilde{x}$ and *x*, and the divergence of μ and σ from the standard normal distribution.

4.1 Kolmogorov–Arnold Network. MLPs [40–42], also known as fully connected feedforward neural networks, are fundamental components of modern deep learning due to their universal approximation capabilities [42]. Given sufficient layers and parameters, MLPs can approximate any continuous function. However, their reliance on *fixed activation functions* and *dense weight matrices* often leads to parameter inefficiencies and reduced interpretability [43,44].

To address these limitations, KANs [45] have been proposed as an alternative to MLPs. KANs are grounded in Kolmogorov–Arnold representation theorem, which states that any multivariable continuous function can be decomposed into a finite sum of continuous functions of a single variable. Instead of using static weight matrices and predefined activation functions, KANs employ *learnable univariate spline functions* on each edge. This approach allows KANs to achieve function approximation with fewer parameters, enhanced adaptability to complex data patterns, and improved interpretability.

4.1.1 Structure of Kolmogorov–Arnold Network Layers. A typical KAN layer is represented as a matrix of univariate functions mapping an input dimension n_{in} to an output dimension n_{out}. Unlike MLPs, which separate linear transformations and nonlinear activation functions, KANs integrate *both operations within the learnable spline-based functions* assigned to each edge. This design enables adaptive, data-driven transformations, making KANs more efficient in modeling high-dimensional relationships. A learnable activation function in a KAN, $\phi(x)$, has two additive parts, a residual function $b(x)$ and the spline function given by

$$\phi(x) = w_b b(x) + w_s \text{spline}(x) \tag{9}$$

where w_s and w_b are trainable parameters that help control the magnitude of the activation function. In Eq. (9), $b(x)$ is given by

$$b(x) = \text{silu}(x) = \frac{x}{1 + e^{-x}} \tag{10}$$

and the spline function is given by

$$\text{spline}(x) = \sum_{i=0}^{G+k-1} c_i B_i(x) \tag{11}$$

where c_i are trainable, k is the degree of the spline, and G is the number of grid points, such that there are $(G + k)$ B-spline bases in total.

To maintain model stability and prevent overfitting, KANs incorporate a regularization loss, consisting of (1) L_1 regularization, which encourages sparsity by minimizing the absolute values of spline weights, and (2) entropy regularization, which ensures a balanced distribution of function complexity across edges, preventing over-reliance on a few dominant connections.

The L_1 norm of the spline function ϕ is defined as the average magnitude over its N_p input values $x^{(s)}$ given by [45]

$$\|\phi\|_1 \equiv \frac{1}{N_p} \sum_{s=1}^{N_p} \left| \phi\left(x^{(s)}\right) \right| \tag{12}$$

For a KAN layer Φ with n_{in} input nodes and n_{out} output nodes, the total L_1 norm is given by [45]

$$\|\Phi\|_1 \equiv \sum_{i=1}^{n_{\text{in}}} \sum_{j=1}^{n_{\text{out}}} \|\phi_{i,j}\|_1 \tag{13}$$

To promote an even distribution of function complexity across edges, KANs also include an entropy-based regularization term, defined as [45]

$$S(\Phi) \equiv -\sum_{i=1}^{n_{\text{in}}} \sum_{j=1}^{n_{\text{out}}} \frac{\|\phi_{i,j}\|_1}{\|\Phi\|_1} \log\left(\frac{\|\phi_{i,j}\|_1}{\|\Phi\|_1} \right) \tag{14}$$

This entropy regularization discourages excessive reliance on a small number of edges, thereby ensuring better generalization and smoother function approximations.

The final loss function for training a KAN is a combination of the mean squared error (MSE) for function approximation and the regularization terms:

$$\mathcal{L} = \text{MSE}(y, \hat{y}) + \lambda_1 \|\Phi\|_1 - \lambda_2 S(\Phi) \tag{15}$$

where λ_1 and λ_2 are hyperparameters controlling the strength of regularization, y is the input, and $\hat{y}$ is the output.

4.2 Variational Autoencoders. A VAE [46] is a type of generative neural network that is made up of two parts: an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation, known as the *latent space*, and the decoder reconstructs the input from this representation. This process allows the model to capture the salient information in the input and learn to generate data that are similar in distribution. The graphic in Fig. 6 shows the structure of a VAE.

A VAE approximates the true data distribution by learning a probabilistic encoder that maps input x to a latent distribution parameterized by a mean $\mu(x)$ and standard deviation $\sigma(x)$. Latent variables z are sampled from this distribution using the reparameterization trick:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{16}$$

This reparameterization enables gradient-based optimization by making the sampling operation differentiable.

The decoder network then reconstructs the input data from the sampled z. The quality of this reconstruction is quantified by the *reconstruction loss*, which measures how closely the generated output matches the original input:

$$\mathcal{L}_{\text{recon}} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] \tag{17}$$

Here, $q(z|x)$ denotes the approximate posterior distribution learned by the encoder, and $p(x|z)$ is the likelihood of reconstructing x given latent vector z. The expectation operator $\mathbb{E}_{q(z|x)}[\cdot]$ computes the average log-likelihood over samples from the latent space.

To encourage the structure in the latent space and prevent overfitting, a regularization term is added: the Kullback–Leibler (KL) divergence between the approximate posterior $q(z|x)$ and a prior $p(z)$, typically a standard Gaussian $\mathcal{N}(0, I)$. The KL divergence is defined as

$$\mathcal{L}_{\text{KL}} = \text{KL}(q(z|x) \parallel p(z)) \tag{18}$$

Table 1 Motion-VAE architecture with MLP layers and attention mechanism

Layer/module	Input dim	Output dim	Description
Encoder MLP layer 1	1080	512	Linear layer for input motion encoding
Encoder self-attention block 1	512	512	Self-transformer block for encoder
Encoder MLP layer 2	512	128	Linear layer for input motion encoding
Encoder self-attention block 2	128	128	Self-transformer block for encoder
Encoder MLP layer 3	128	64	Linear layer for input motion encoding
Encoder self-attention block 3	64	64	Self-transformer block for encoder
Mean vector calculation	64	64	Linear layer for calculating mean
Logvar vector calculation	64	64	Linear layer for calculating logvar
Reparameterization trick	2×64	64	Reparameterization trick for latent vector calculation
Decoder MLP layer 1	64	128	Linear layer for latent vector decoding
Decoder self-attention block 1	128	128	Self-transformer block for decoder
Decoder MLP layer 2	128	512	Linear layer for latent vector decoding
Decoder self-attention block 2	512	512	Self-transformer block for decoder
Decoder MLP layer 3	512	1080	Linear layer for latent vector decoding
Decoder self-attention block 3	1080	1080	Self-transformer block for decoder

Table 2 Motion β-VAE architecture with KAN layers

Layer/module	Input dim	Output dim	Description
Encoder KAN layer 1	1080	512	Linear layer for input motion encoding
Encoder KAN layer 2	512	128	Linear layer for input motion encoding
Encoder KAN layer 3	128	10	Linear layer for input motion encoding
Mean vector calculation	10	10	Linear layer for calculating mean
Logvar vector calculation	10	10	Linear layer for calculating logvar
Reparameterization trick	2×10	10	Reparameterization trick for latent vector calculation
Decoder KAN layer 1	10	128	Linear layer for latent vector decoding
Decoder KAN layer 2	128	512	Linear layer for latent vector decoding
Decoder KAN layer 3	512	1080	Linear layer for latent vector decoding

This term ensures that the latent representations remain close to a known, continuous distribution, facilitating smooth interpolation and generative sampling.

The full VAE loss function combines reconstruction and regularization terms:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}} + \beta \cdot \mathcal{L}_{\text{KL}} \tag{19}$$

Here, β is a weighting coefficient introduced in β-VAE [47], controlling the trade-off between reconstruction fidelity and latent space regularization. A higher β encourages more structured and disentangled latent factors, while a lower β prioritizes accurate reconstruction of the input.

In summary, VAEs are powerful generative models that encode inputs into a probabilistic latent space and decode from it to generate realistic data, balancing reconstruction accuracy with latent space regularity through principled probabilistic modeling.

4.3 Neural Network Architectures for Motion Variational Autoencoder. The first step in our method is to train a VAE with normalized motions of planar four-bar mechanisms. By creating a well-structured latent space, the VAE's encoder can be used to create a lower dimensional representation of a continuous motion discretized by 360 poses. This new representation preserves the salient features of the motion and allows for generating similar motions by sampling in the latent space using Eq. (16).

In an effort to create a motion VAE, we began with a combination of MLP and attention mechanisms, a technique proven by its efficacy across numerous domains [48–52]. This network architecture is presented in Table 1, which was obtained after several computational experiments. An MLP layer is composed of sequentially connected layers, followed by a batch normalization [53], rectified linear unit activation [54], and a dropout layer with a probability of 10% [55]. Each MLP layer is followed by a self-attention block which assesses the same input vector relevancy. The attention weight computation transitions the primary input vector into a query vector $\mathbf{Q}$, and a secondary vector into key $\mathbf{K}$ and value $\mathbf{V}$ vectors via distinct MLPs. Employing softmax$(\mathbf{Q}\cdot\mathbf{K}^T/\sqrt{d_k})$ [56] calculates attention scores, with attention-directed interactions encapsulated through the attentive aggregation of $\mathbf{V}$.

In an effort to reduce the latent space dimension, while forcing the models to cluster similar motions together and reconstruct input motions accurately, we replaced MLP layers with KAN layers [45]. This change allowed us to achieve better reconstruction while using a much lower latent dimension (from 64 to 10). We used a more efficient implementation of KAN [57] which calculates the L_1 regularization loss differently than is presented in the original work [45]. This architecture is presented in Table 2.

The KAN layers used in our VAE architecture all use a spline function of degree 3 and grid size of 5, which after some experimentation were found to be the minimum values necessary to

Table 3 Conditional-β-C-VAE architecture

Layer/module	Input dim	Output dim	Description
Encoder	26	50	KAN layer for input + condition encoding
Mean vector calculation	50	50	KAN layer for calculating mean
Logvar vector calculation	50	50	KAN layer for calculating logvar
Reparameterization trick	2×50	50	Reparameterization trick for latent vector calculation
Decoder	60	16	KAN layer for $\mathbf{p}$ vectors prediction

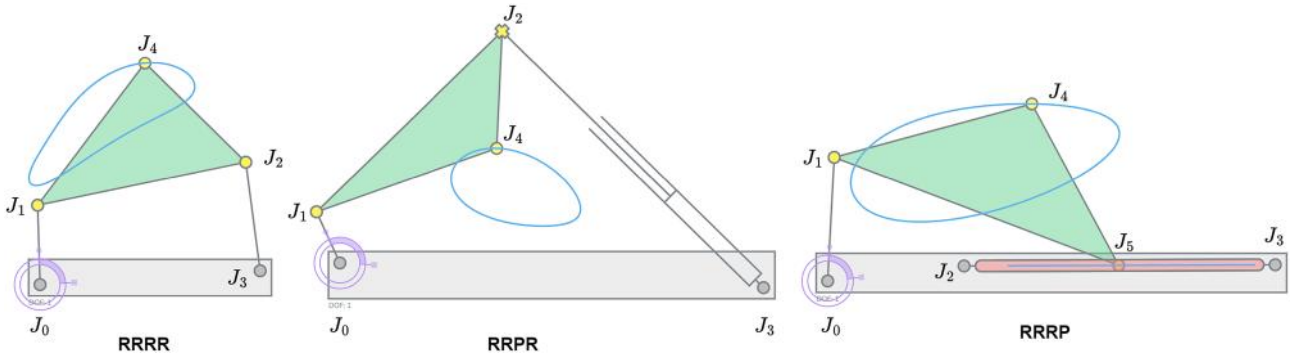


Fig. 7 The dataset includes three types of mechanisms: RRRR, RRRP, and RRPR. Joint J_0 is fixed for all mechanisms to the coordinate (1, 1), while the other joint coordinates (J_1, J_2, J_3, J_4) are generated randomly on a grid of 20 × 20 going from −10 to 10 on the *x* and *y* axes.

obtain good reconstruction. In the context of KANs, the grid size refers to the number of Bezier segments of the B-spline activation functions. A larger grid size means more segments, resulting in smoother and more accurate approximations, but contributes to computationally more expensive training. We experimented with different values of β and found that $\beta = 1000$ resulted in the best combination of accuracy and variation in output motions.

Both models were subject to an identical dataset training over 300 epochs with a batch size of 1024. The models were trained on an Ubuntu Linux computer with an AMD Ryzen Threadripper Pro 5955WX 16-Cores processor, two Nvidia RTX A6000 GPUs, and 258 GB of primary memory. The ML library used was PyTorch, and the python version was 3.12.3. The MLP + attention model and KAN model consisted of 576,849 and 248,999 trainable parameters, respectively, and both used a β of 1000. The MLP + attention model recorded a reconstruction loss of 0.358 and a KL divergence of 0.00611. The KAN model recorded a reconstruction loss of 0.822 and a KL divergence of 0.00113. Despite the lower reconstruction loss of the MLP + attention model, a visual inspection of the reconstruction showed that it was failing to preserve the smooth change in orientation of a four-bar motion seen by the KAN model. Furthermore, the KAN model is half the size of the other model and showed good reconstruction for a latent space dimension as low as 10.

4.4 Conditional Variational Autoencoder. C-VAE [58] is a variation of the VAE designed for supervised learning, whereby a label can be provided alongside the input to allow the model to generate data given additional information in the form of a condition. A C-VAE distinguishes itself from a traditional VAE through its architecture by incorporating both the condition and the input into the encoding process to produce a latent vector. Additionally, during the decoding phase, the condition is combined with this latent vector to facilitate the reconstruction of the input. The loss function for C-VAE is an adjusted version of Eq. (19) that includes the condition, denoted by c, given by

$$\mathcal{L}_{\text{cVAE}} = -\mathbb{E}_{q(z|x,c)}[\log p(x|z, c)] + \beta \cdot \text{KL}(q(z|x, c)\|p(z)) \quad (20)$$

The input and expected output of our C-VAE model is a concatenated vector of two **p** vectors encoding the geometric parameters of two dyads. The normalized motion of the coupler is used as the condition. The motion can be represented by an arbitrary number of poses. In this work, we chose to represent a motion with 360 poses of the form (x, y, ϕ).

While training the model, the MSE loss is used as a reconstruction term to help ensure the output **p** vector pair is reconstructed with minimal error. Since a dyad's parameters are encoded in a **p** vector, minimizing the MSE between the input and output **p** vectors contributes to more accurate reconstruction of the mechanism parameters obtained from an inverse computation of the **p** vector. We note that a goal of this article is to generate several defect-free mechanisms for a given motion, not merely reconstruct a mechanism from the database, and this is accomplished by sampling from the latent space.

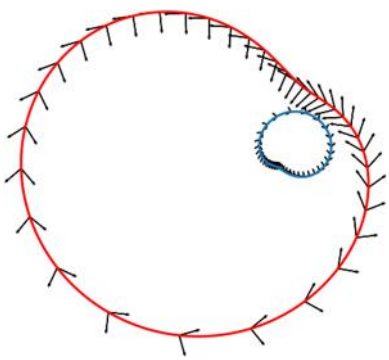


Fig. 8 The original larger motion undergoes scaling, rotation, and translation to yield the smaller normalized motion. The orientation of the coupler pose is shown for crank increments of 10 deg, and the orientation is visualized using frames.

4.5 Neural Network Architecture of C-VAE. Similar to our VAE, we use KAN layers throughout the C-VAE network where we base our architecture on the foundational design proposed by Sohn et al. [58], concatenating input and conditional vectors before their progression through an encoder of fully connected layers. The same concatenation method is applied in the decoder, combining the latent vector of dimension 10 from motion VAE with the latent sampling of the C-VAE for decoding. The full architecture for our C-VAE is outlined in Table 3.

Two **p** vectors representing the mechanism's parameters and type are concatenated and used as input to the C-VAE. The input motion is normalized using the procedure discussed in Sec. 5.2 and passed through the encoder of the motion VAE to get a latent vector representing it. The latent vector is used as the condition for the C-VAE.

Inference allows for latent space exploration within specified encoded motion conditions, generating mechanisms that generate a close match to the desired motion. We found that a grid size of 5 and a spline degree of 3 for KAN layers were sufficient, as increasing these values did not improve results and only increased model training time. Experimentation using different latent dimensions and β values showed that a β of 1000 and a latent dimension of 50 produced the best balance of accuracy and diversity in generated mechanisms. The model consisted of 253, 849 trainable parameters. It was trained over 300 epochs with a batch size of

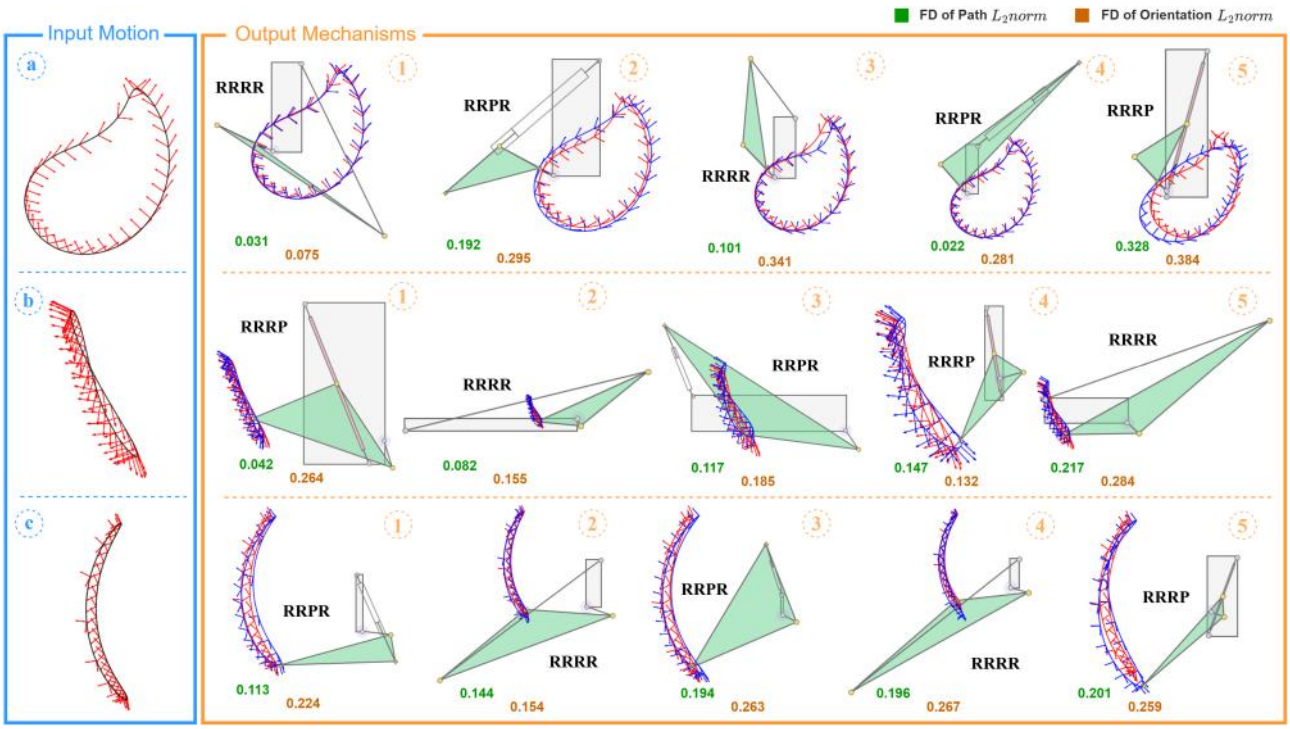


Fig. 9 Three input motions are shown along with five mechanisms generated by our method for each input motion. The coupler's poses, represented by frames, sampled every 15 deg of crank rotation. The coupler path of a generated mechanism is overlaid with the input motion to highlight their visual similarity. For each input motion, five mechanisms are filtered using the L_2 norm of the difference of Fourier descriptors for the path. For an input motion, the output mechanisms are presented in order of increasing L_2 norm value for the FD of the path, and the mechanisms are labeled by their types.

1024, achieving a reconstruction loss of 0.0684 and a KL divergence of 0.000123.

5 Dataset Generation

In this section, we present the process by which mechanism joints are generated, with the goal of the resulting mechanisms producing closed motions when the crank undergoes a full rotation. Then, we discuss the normalization process applied to four-bar motions before passing the normalized motion through the encoder of the VAE.

5.1 Generating Four-Bar Joint Combinations. In order to generate a dataset of four-bar mechanisms with different types, coupler motions, and link ratios, we produce arbitrary combinations of joint locations. Each type of four-bar mechanisms is generated by the same number of five joints ($J_0 \ldots J_4$); see Fig. 7. For all mechanisms, J_4 is the coupler point and its path and orientation are used to generate a motion.

The dataset generation process begins by fixing J_0, which we choose arbitrarily to be (1, 1), and varying the rest of the joint locations between the interval $[-10, 10]$, for generated (x, y) positions, to generate a large dataset of four-bar joint combinations. Three conditions are set to guarantee that we obtain joint locations for unique mechanisms that can achieve a closed motion. The first condition is that the mechanism satisfies the Grashof condition [33] to ensure the crank can undergo a full rotation. The second condition requires that no joints in the combination $(J_0, J_1, J_2, J_3, J_4)$ share the same coordinate, as it might result in a mechanism that is not a four-bar. The third condition ensures the dataset does not consist of large link lengths relative to one another. To achieve this, we set the condition that the ratio of the largest link to the smallest link in the mechanism is less than 4.

The joint combinations generated are used to simulate the motion of their respective mechanisms using a unified simulation algorithm [38]. The simulator provides the coordinates of the coupler point at each degree of rotation of the crank. So, for a closed coupler curve, the simulator produces 360 points that discretize the coupler curve. After the coupler curve is generated, we do a final check of the number of coupler points, and if it is less than 360, indicative of an open curve, the joints are discarded. The orientation ϕ of the coupler pose, at each coupler point along the path, is computed as follows:

$$\boldsymbol{V_1} = \frac{J_4 - J_1}{\|J_4 - J_1\|}$$

$$\phi = \arctan2(V_{1,y}, V_{1,x})$$

Overall, the dataset contains 1,033,913 RRRR mechanisms, 1,090,404 RRRP mechanisms, and 1,128,997 RRPR mechanisms. The $\mathbf{p}$ vectors for each generated mechanism are calculated using Eq. (7). The joint locations, original motion poses, normalized poses, and $\mathbf{p}$ vectors for all three types of mechanisms in our dataset are accessible.[3]

5.2 Motion Normalization Procedure. In this work, every coupler motion is normalized to make sure it is invariant with respect to rotation, uniform scaling, translation, and reflection. We start the process by centering the motion curve around (0, 0) by subtracting the mean $(\bar{x}, \bar{y})$ from each coupler point (x_i, y_i). We then follow a geometric normalization algorithm [59] that utilizes principal component analysis (PCA) whitening for scale and translation invariance, followed by independent component

[3] https://www.kaggle.com/datasets/purwarlab/rrrr-rrrp-rrpr

Table 4 Generated mechanisms' p-vectors for input motion (*a*) where the initial pose is (−1.25, 2.18, −0.43)

p-Vectors
p_1^1 : [−0.061465, −0.476217, −0.000076, 0.061416, 0.061417, −0.479849, 0.239475, −0.681469]
p_1^2 : [−0.000398, 0.027868, 0.000657, −0.004557, −0.002300, 0.101375, −0.018646, 1.060089]
p_2^1 : [−0.053312, −0.115589, 0.000084, 0.053056, 0.053227, −0.117420, 0.059554, 1.260212]
p_2^2 : [−0.005338, 0.004110, −0.006998, 0.016110, 0.036834, −0.841159, 0.147075, −0.328699]
p_3^1 : [−0.137992, −0.599171, 0.000009, 0.137959, 0.137977, −0.601774, 0.300738, −0.363536]
p_3^2 : [−0.010773, −0.039081, −0.018019, −0.102989, −0.003453, −0.114865, −0.871481, −0.335523]
p_4^1 : [−0.056328, −0.457369, 0.000028, 0.056349, 0.056333, −0.459932, 0.229675, −0.710901]
p_4^2 : [−0.003741, −0.027606, −0.000281, 0.007121, −0.014300, −0.127446, 0.185259, −0.922082]
p_5^1 : [−0.046438, −0.397906, 0.000079, 0.046383, 0.046402, −0.398912, 0.199102, −0.795165]
p_5^2 : [−0.007877, −0.050187, 0.010641, 0.016577, −0.009459, 0.516660, 0.261980, −0.648628]

Table 5 Generated mechanisms' p-vectors for input motion (*b*) where the initial pose is (−7.26, −0.51, −3.13)

p-Vectors
p_1^1 : [−0.035206, −0.347010, −0.000023, 0.035210, 0.035205, −0.348816, 0.174637, −0.848766]
p_1^2 : [−0.006938, −0.006874, −0.007817, −0.041669, −0.004975, −0.124753, −0.446701, 0.853344]
p_2^1 : [−0.052593, −0.410557, 0.000006, 0.052655, 0.052640, −0.412639, 0.205628, −0.774102]
p_2^2 : [−0.008053, −0.011037, −0.009197, −0.017850, 0.020630, −0.582789, −0.190842, 0.007662]
p_3^1 : [−0.121613, −0.589851, 0.000117, 0.121636, 0.121621, −0.590534, 0.295860, −0.038052]
p_3^2 : [−0.004519, −0.061770, −0.002624, −0.018701, −0.001267, 0.538357, −0.248519, −1.268157]
p_4^1 : [−0.068449, −0.490513, −0.000041, 0.068363, 0.068400, −0.492417, 0.246828, −0.649643]
p_4^2 : [−0.003932, −0.004324, −0.011685, −0.086369, −0.008513, −0.146005, −0.754997, 1.074805]
p_5^1 : [−0.075576, −0.462813, 0.000048, 0.075585, 0.075601, −0.464558, 0.232376, −0.704819]
p_5^2 : [−0.011338, −0.009916, 0.000746, 0.015756, 0.014263, −0.123773, 0.261150, 0.009074]

analysis (ICA), which is done using the fixed-point algorithm (FastICA) for its simplicity and computational efficiency [60].

Given two coupler curves X and $\bar{X}$, that are related by an affine transformation, the PCA whitening of the two sets of points, X and $\bar{X}$, makes them orthogonally equivalent, such that $\hat{\bar{X}} = Q\hat{\bar{X}}$, where $\hat{X}$ and $\hat{\bar{X}}$ are the normalized data sets obtained from X and $\bar{X}$, and Q is an orthogonal matrix. As mentioned before, this makes the coupler curves invariant to scale and translation by reducing the affine relationship to an orthogonal one.

At this step, the normalized curve, $\hat{\hat{X}}$, is invariant to scale and translation. The FastICA is a fixed-point iteration algorithm that finds a maximum of the non-Gaussianity of $Q^T\hat{\hat{X}}$. So, using FastICA, $\hat{\hat{X}}$ can be normalized with respect to rotation by $\hat{\hat{X}}_R = Q^T\hat{\hat{X}}$, where $\hat{\hat{X}}_R$ is the normalized curve with respect to rotation. Finally, the horizontal and vertical reflections are normalized by

$$\hat{\hat{X}}_{\mathrm{RR}} = \begin{bmatrix} \mathrm{sgn}(m_{1,2}) & 0 \\ 0 & \mathrm{sgn}(m_{2,1}) \end{bmatrix} \hat{\hat{X}}_{\mathrm{R}}$$

where sgn is the signum function, and $m_{1,2}$ and $m_{2,1}$ are the third-order moments of $\hat{\hat{X}}_{\mathrm{R}}$ given by

$$m_{p,q} = \frac{1}{N}\sum_{i=0}^{N-1} x_i^p y_i^q$$

After a coupler curve is normalized following the above procedure, we compute the covariance matrix of the original and the normalized curve. The angle of rotation, in radians, is calculated from the covariance matrix and added to each original pose ϕ_i to account for the rotation. Finally, the orientations are normalized by 2π. Figure 8 shows a coupler motion before (bigger motion) and after normalization (small motion). The transformation from the original motion to the normalized motion is stored, so that its inverse can be used on the output mechanism's normalized motion to produce a new motion that aligns with the input motion in position, scale, and orientation of the central axis.

5.3 Preparing the Data for Training. For training the motion VAE, normalized poses are packed into a tensor of shape 1080×1 as follows, $[x_i, y_i, \theta_i]$, $i = 1 \ldots 360$. The trained VAE model is then used to find the latent representation 10×1 of each normalized motion for use as a condition when training the C-VAE. Two **p** vectors, representing the mechanism parameters, are concatenated into a tensor of shape 16×1, and are used as the input to the C-VAE during training. The dataset is split into training, testing, and validation datasets by the ratio 70 : 15 : 15, respectively. For each of these datasets, one row contains a tensor of size 1099. The first 16 numbers are the pair of **p** vectors. The next 1080 numbers are the normalized poses. And the last three numbers are the coupler's pose at $t = 0$ for the original motion. This information was added because it is needed for the inverse computation of dyad parameters from a **p** vector, which is helpful for plotting the sampled mechanism's motion against the original motion from the testing dataset.

6 Identifying Motion Similarity and Mechanism Diversity

Sampling n times in the latent space of the C-VAE and passing the resulting latent vectors through the decoder yield n pairs of **p** vectors. From a pair of **p** vectors, a mechanism's parameters are obtained and the motion of its coupler is generated. FDs [61] of the path and coupler orientations are used to quantify the similarity of the output motion with the input motion. Joint locations are used to define the criteria for preventing the generation of duplicate mechanisms with the highest motion similarity.

Table 6 Generated mechanisms' p-vectors for input motion (c) where the initial pose is (−7.30, 0.55, 3.08)

p-Vectors
p_1^1 : [−0.110715, 0.013552, 0.000132, 0.110727, 0.110759, 0.012744, −0.005054, 2.668711]
p_1^2 : [−0.008358, 0.010360, −0.018644, −0.051382, −0.025961, 0.190136, −0.361031, −0.862383]
p_2^1 : [−0.043590, −0.421301, 0.000072, 0.043661, 0.043638, −0.423726, 0.211951, −0.179177]
p_2^2 : [−0.003596, −0.052872, −0.001292, −0.024551, −0.009619, 0.932015, −0.276301, −1.198711]
p_3^1 : [−0.109801, −0.472813, 0.000033, 0.109605, 0.109723, −0.473253, 0.236458, 0.764304]
p_3^2 : [−0.004496, 0.028780, 0.002154, 0.036105, −0.020649, −0.311649, 0.376057, 0.885589]
p_4^1 : [−0.048322, −0.401829, −0.000003, 0.048334, 0.048305, −0.404893, 0.201975, −0.784266]
p_4^2 : [−0.005191, −0.013786, −0.012834, −0.057370, 0.026788, −0.839701, −0.588956, 0.229593]
p_5^1 : [−0.036448, −0.362137, 0.000016, 0.036435, 0.036459, −0.365051, 0.183109, −0.835515]
p_5^2 : [−0.003449, −0.018654, 0.001799, −0.016448, −0.036097, 0.951884, −0.268276, 0.079570]

Table 7 Four-bar mechanism joint locations in the form [J_0, J_1, J_3, J_2, J_4] for RRRR and RRPR mechanisms and in the form [J_0, J_1, J_3, J_2, J_5, J_4] for RRRP mechanisms following the joint labeling convention in Fig. 7

Original mechanism joint locations
RRRR: [1, 1], [5.6927−4.4898], [−9.7863−5.0354], [−5.1789.358][7.51630.3498]
Mechanism joint locations for five poses
RRPR (1): [0.5231, 2.4743], [6.8476, −0.9213], [−7.2263, −5.4216], [−42.5456, −11.808], [7.6415, 2.856]
RRPR (2): [0.2349, 3.5451], [7.5939, 1.2404], [−4.9972, −6.8719], [31.4715, 5.6462], [8.1832, 3.8616]
RRRP (1): [−1.4062, 2.4159], [7.7027, 1.5054], [15.82, 34.9928], [6.0677, 16.1875], [7.3129, 5.645], [−3.6846, −2.6178]
RRRP (2): [−2.971, 1.094], [4.8248, −3.6142], [9.2695, 30.4112], [2.0956, 13.5515], [5.4095, 2.4946], [−5.0783, −3.3081]
RRRR (1): [0.9519, 1.7942], [6.954, −2.3549], [−7.2518, −6.7058], [−4.2665, 6.1435], [7.8634, 2.2531]
RRRR (2): [0.6691, 2.0367], [6.7265, −1.9986], [−7.1719, −6.9462], [−6.1273, 15.2609], [7.7418, 2.2124]
Mechanism joint locations for five poses (±**0.1**, ±**1** deg)
RRPR (1): [1.70260.4303], [8.8917−2.9532], [−6.3942−34.9644], [32.4773−26.7604], [8.05152.9103]
RRPR (2): [0.46253.113], [7.29730.3425], [−5.7122−5.7628], [30.10423.0098], [7.85493.5364]
RRRR (1): [0.7792.223], [6.3732−2.4462], [−6.8716−5.296], [−3.66484.9096], [7.70671.6215]
RRRR (2): [1.24181.9362], [8.7145−0.275], [−9.6854−6.6281], [−5.35581.1647], [8.33834.2356]
RRRP (1): [−2.14881.8717], [6.6886−0.7091], [10.950543.156], [3.351819.6879], [6.70463.9139], [−4.2469−3.7803]
RRRP (2): [−1.54462.7675], [7.4143.2303], [9.964633.92], [3.482815.9006], [6.48756.9122], [−2.999−2.1188]

6.1 Fourier Descriptors. FDs can be used to represent a closed curve of a mechanism [62–64], discretized by the points (x_i, y_i), and the orientations (θ_i) of the poses [61]. The Cartesian coordinates of points on the coupler curve, as a function of time, can be denoted by $x(t)$ and $y(t)$. They can then be cast into complex form as $z(t) = x(t) + iy(t)$. Similarly, the orientation as a function of time $\theta(t)$ is given in complex form by $e^{i\theta(t)} = \cos\theta(t) + i\sin\theta(t)$.

For a closed motion, given as a function of t, it can be rewritten as the summation of a series

$$z(t) = \sum_{m=-\infty}^{\infty} a_m e^{2\pi i m t}$$

$$\theta(t) = \sum_{m=-\infty}^{\infty} b_m e^{2\pi i m t}$$

where a_m denotes the FDs for the path and b_m denotes the FDs for the orientation. The Fourier descriptors can be calculated using the following approximation:

$$a_m = \frac{1}{N}\sum_{k=0}^{N-1} z_k e^{2\pi i m \frac{k}{N}}$$

$$b_m = \frac{1}{N}\sum_{k=0}^{N-1} \theta_k e^{2\pi i m \frac{k}{N}}$$

where z_k are the given points in complex form, θ_k are the given orientations in complex form, and N is the number of points.

Descriptor a_0 is called the fundamental harmonic. Descriptors a_{-1} and a_1 are called the first harmonics, a_{-2} and a_2 are called the second harmonics, and so on. It has been shown that $m = 5$ is sufficient for capturing the coupler path of a four-bar

Table 8 Pose coordinates and angle in radians in the form [x, y, θ]

Pose	Pose data for five poses	Pose data for five poses (±0.1, ±1 deg)	Pose data for five arbitrary poses
1	[7.5163, 0.3498, 1.2104]	[7.6163, 0.2498, 1.2279]	[4.9511, −2.5979, −1.1519]
2	[4.9893, 10.642, 1.8652]	[4.8893, 10.542, 1.8477]	[3.6536, 0.8216, 0.6255]
3	[−3.7786, 12.2772, 2.1534]	[−3.6786, 12.3772, 2.136]	[−0.1545, −0.499, −0.9499]
4	[−8.9945, 5.2998, 2.1366]	[−9.0945, 5.3998, 2.1192]	[−1.4778, 3.4081, 2.3731]
5	[−0.8476, −0.2924, 1.1227]	[−0.7476, −0.3924, 1.1402]	[2.3752, 4.0117, −0.6898]

accurately [61], so we use $m = 5$ to calculate the FD of the path and orientation. The L_2 norm for the original and sampled mechanism's $(a_{-5}, a_{-4}, \ldots, a_4, a_5)$ and $(b_{-5}, b_{-4}, \ldots, b_4, b_5)$ are calculated for path and orientation, respectively, to calculate the similarity between two motions.

6.2 Harmonic Coupling of Rotational and Translational Components of a Planar Four-Bar Motion. For a planar four-bar motion, the FDs of its rotational and translational components must be linearly proportional [61]:

$$\cdots \frac{a_{-3}}{b_{-3}} = \frac{a_{-2}}{b_{-2}} = \frac{a_{-1}}{b_{-1}} = \frac{a_2}{b_2} = \frac{a_3}{b_3} \cdots \tag{21}$$

There is a four-bar mechanism that can generate a given motion, if the FDs of its path and orientation satisfy the relation in Eq. (21), approximately. The approximation error for how well a motion satisfies Eq. (21), given the translational components as T_k and rotational components as Q_k, is given by

$$I_0 = \sum_{k \neq 0,1} (|T_k| - r|Q_k|)^2$$
$$r = \frac{\sum_{k \neq 0,1} |T_k|}{\sum_{k \neq 0,1} |Q_k|} \tag{22}$$

The linear proportionality of the translational and rotational components in Eq. (21) shows that when comparing two motions, a high similarity in path will also yield a high similarity in orientation and vice versa. Path and orientation are not independent for the motion of a four-bar linkage, as their FD values are linearly proportional.

6.3 Criteria for Eliminating Similar Mechanisms. Two or more mechanisms, of the same type, are considered similar if the L_2 norm of the input normalized joint locations and generated normalized joint locations is less than 0.5. Following the normalization of the coupler's path using the procedure discussed in Sec. 5.2, the homogeneous transformation matrix that transforms the original curve to the normalized curve is applied onto the original joints to obtain the normalized joints. This ensures that no two mechanisms of the same type have very similar joint configurations, thus promoting diversity in the generated mechanisms.

For every mechanism that has a coupler motion satisfying the chosen similarity threshold, it is compared on the basis of joint locations to the other mechanisms of the same type. On the basis of joint locations, we normalize the joints and compute the L2 norm difference in joint coordinates of two mechanisms. If the norm is less than 0.5, the mechanisms with a better motion match are chosen to be kept.

7 Results and Discussion

To examine the model's ability to produce mechanisms with different parameters that closely approximate an input motion, we choose three distinct motions generated by four-bar mechanisms not included in the training dataset. We first normalize the input motions and pass them through the motion VAE to obtain their latent space representations. These latent vectors serve as conditional inputs to the C-VAE. During inference, we concatenate each conditional latent vector with a random vector sampled from a standard normal distribution and feed this combined vector into the C-VAE decoder. This process generates 1000 distinct pairs of **p** vectors. The five joint locations, used to construct a mechanism, are computed from a pair of **p** vectors using Eq. (8) and the initial coupler pose. The four-bar mechanism is simulated to obtain the output coupler motion.

Sampled mechanisms that produce open motions are discarded. For mechanisms that produce closed motions, the L_2 norm of the difference of FDs for the input and output path and orientation is calculated to be used as a metric for comparing motion similarity. Given one set of FDs as x and another as y, the L_2 norm can be calculated using the following equation:

$$\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_i (x_i - y_i)^2} \tag{23}$$

Figure 9 shows three input motions that are distinct from one another. For each input motion, five mechanisms of different types approximating the input motion are selected and their L_2 norm of the difference of the FDs of the input path and orientation and those of the generated mechanisms are given.

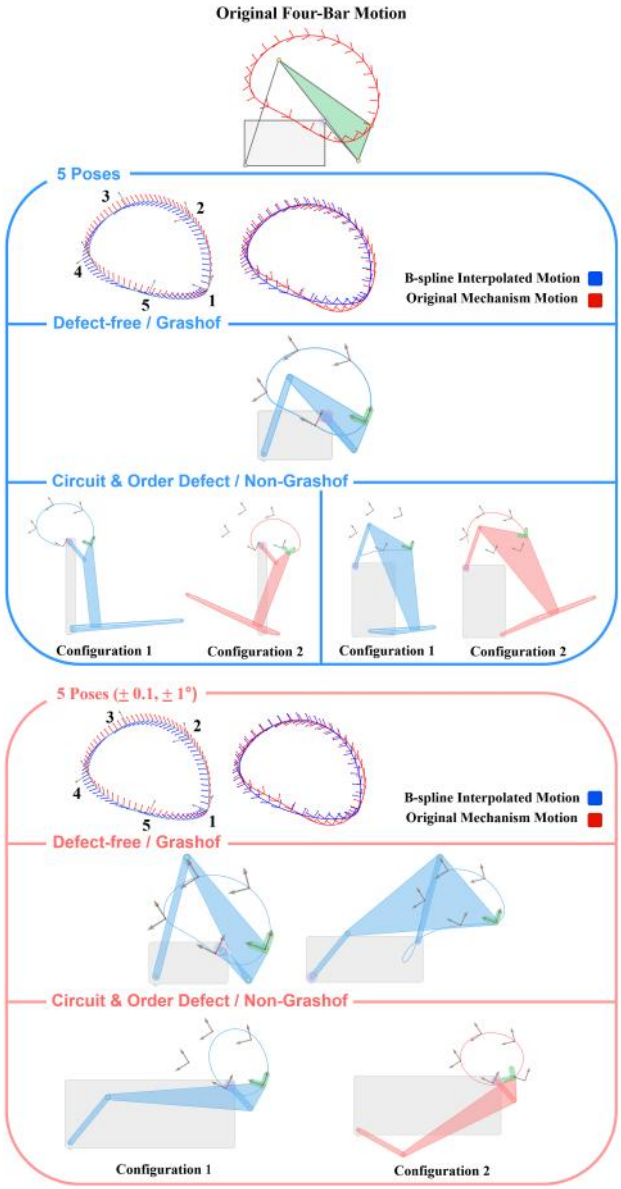


Fig. 10 Five poses are sampled from a RRRR mechanism and used as input into the algebraic fitting algorithm in MotionGen. Similarly, five poses ±0.1 for *x* and *y* coordinates and ±0.1 deg in orientation are used as input and the resulting mechanisms are shown for each input. Mechanisms displayed in red indicate a change in configuration of the blue mechanism to its left, achieved through reassembly of the linkage. This highlights the circuit defect that many solutions produced by this algorithm suffer from.

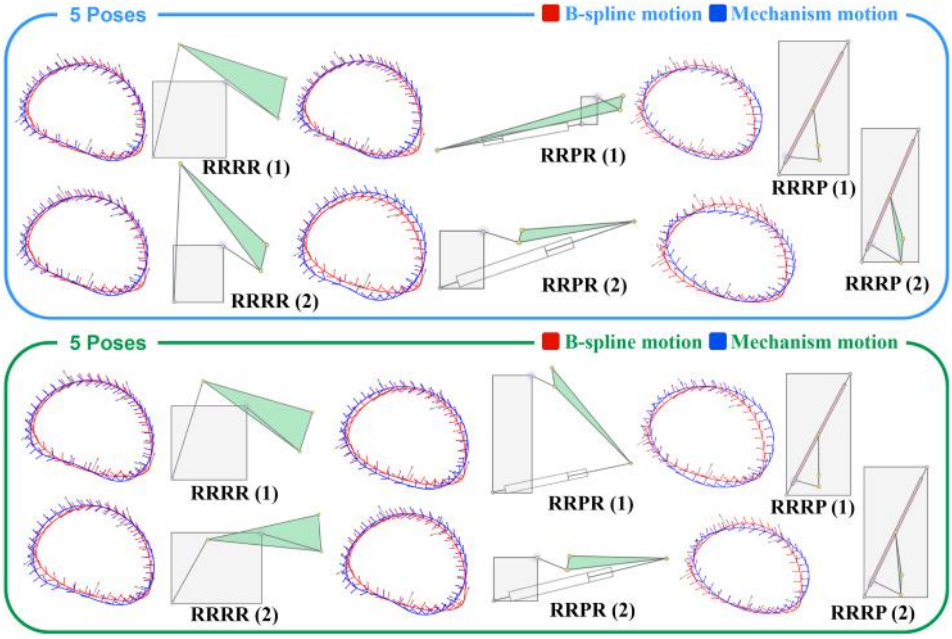


Fig. 11 Four-bar mechanism results from our ML-pipeline for two different cubic B-spline interpolated motions using five poses, and five poses ±0.1 for *x* and *y* coordinates and ±0.1 deg in orientation. For each input, two RRPR, RRRP, and RRRR mechanisms are provided on the basis of motion similarity with the input motion. The motion similarity scores are provided in Table 9.

Based on the perceptual similarity of two motions, we determined that a value less than 0.5 for the L_2 norm difference in FDs for the path provides an acceptable match in motion profile, whereby there is perceptual similarity in the path and orientation of the input and generated motions. A lower value means that the FDs match well and provide better motion generation. In Fig. 9, both L_2 norms of FD for path and orientation are provided and color-coded for comparison among generated mechanisms for each input motion. The values fall below the threshold denoting an acceptable motion match. We also show mechanisms with poorer motion matches, compared to other generated mechanisms, that are of a different mechanism type to highlight the model's ability to generate mechanisms of different types approximating a given motion. For example, in Fig. 9, it can be seen that the RRRP mechanism has the poorest path and orientation match, but optimization can possibly get it closer to the desired motion.

For the 15 mechanisms shown in Fig. 9, the **p** vectors for each dyad are given in Tables 4–6. The model generates pairs of **p** vectors, for which the first one is always an RR, since it was trained on mechanisms that are actuated using RR dyads, and the second **p** vector is interpreted three different times, once as an RR, then as an RP, and finally as a PR. So, from a pair of **p** vectors, we can generate a RRRR mechanism, a RRRP mechanism, and a RRPR mechanism, and the ones with the acceptable motion matches are kept.

The generated mechanisms not only produce coupler motions that closely approximate the input motion, but it can be seen that our method produces mechanisms of different types and produces variation in joint configurations of mechanisms of the same type. These results show that **p** vectors are sufficient for the representation of a mechanism's type and parameters in ML methods for path or motion generation.

8 Comparison of the Algebraic Fitting Algorithm and Our Method for Motion Generation

The algebraic fitting algorithm [37] solves the motion synthesis problem by fitting a set of planar displacements, represented as points in a three-dimensional projective space, to algebraic surfaces called G-manifolds. By using SVD, the least-squares fitting process identifies the best-fit manifolds satisfying the relations in Eq. (6), leading to the design parameters of the linkage. For exact synthesis, 4 or 5 poses are required to obtain independent parameters of a dyad. When the number of poses exceeds five, the algorithm performs approximate synthesis by finding a solution that best fits the given poses in the least-squares sense.

This algorithm, however, often produces defective mechanisms that primarily suffer from order, circuit, and branch defects. Furthermore, the resulting solutions may not include any Grashof mechanisms. Our method presented in this article solves these shortcomings of the algebraic fitting algorithm by generating defect-free Grashof four-bar mechanisms of different types.

To demonstrate the efficacy in addressing these issues, we sample five poses from a four-bar mechanism to be used as input

Table 9 Path and orientation similarity values for mechanisms in Fig. 11 calculated using the L_2 norm of the FDs of path and orientation

		RRPR		RRRP		RRRR	
		(1)	(2)	(1)	(2)	(1)	(2)
Five poses	L_2 norm FD path	0.147	0.252	0.397	0.280	0.099	0.089
	L_2 norm FD orientation	0.172	0.213	0.492	0.421	0.170	0.133
Five poses (± 0.1, ± 1 deg)	L_2 norm FD path	0.177	0.221	0.294	0.272	0.069	0.172
	L_2 norm FD orientation	0.275	0.240	0.442	0.401	0.190	0.295

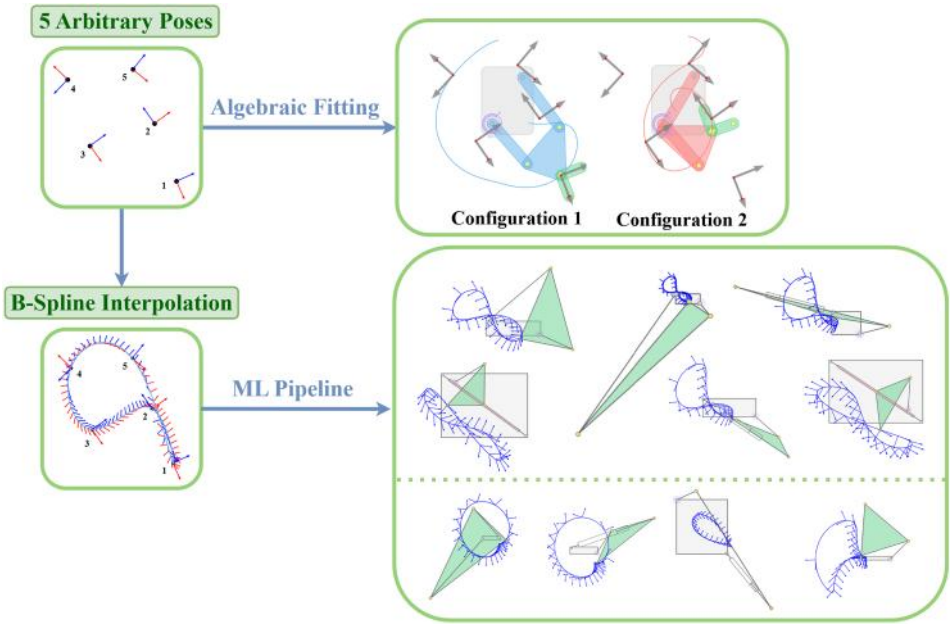


Fig. 12 Five arbitrary poses are used as input to the algebraic fitting algorithm which yields a single defective mechanism with two configurations that can satisfy certain poses in the wrong order. The same poses are used to generate a continuous motion, using B-spline interpolation, for input to our ML-pipeline. The resulting mechanisms include the three types: RRRR, RRRP, and RRPR. The four-bar motions above the dotted line capture the twist in the path between poses 1 and 2, while the ones below it capture path features resembling the path formed from poses 2 to 5.

to the motion synthesis algorithm in MotionGen. In order to use these five poses in our method, we perform a cubic B-spline interpolation using the *scipy* python library [65] to obtain a continuous motion of 360 poses to be used as input. These poses are strategically chosen such that the interpolation produces a very similar motion. The joint locations of the original mechanism are provided in Table 7, and the five sampled poses are provided in Table 8. Using these five poses as input to the motion synthesis algorithm in MotionGen, we obtain only one defect-free mechanism and two non-Grashof mechanisms that suffer from circuit and order defects, shown in Fig. 10.

We introduce slight variations to these 5 poses to observe the effect of small changes on the quality of solutions. The x and y coordinates of each pose are randomly perturbed by ± 0.1. Similarly, a variation of ± 0.1 deg is introduced to the orientation of every pose. With these variations, the algebraic fitting algorithm manages to find two Grashof mechanisms and one non-Grashof mechanism, all of which are RRRR. We can see that small variations in the position and orientation of input poses affect the type of generated mechanisms, as well as their quality. Perturbing the position and orientation of input poses, slightly, resulted in no RRRP mechanisms, and one more Grashof RRRR mechanism, different from the one previously generated.

Next, a cubic B-spline interpolation is performed for the same sets of poses used for algebraic fitting in Table 8 to produce a continuous motion of 360 poses. As shown in Fig. 11, our ML-pipeline can produce at least two mechanisms for each of the following types: RRPR, RRRP, and RRRR, approximating the input motion. Although the results do not satisfy the five input poses as well as the algebraic fitting algorithm does, it is consistent in providing defect-free Grashof mechanisms with perceptually similar motions, with the exception of RRRP mechanisms that had the poorest match of the three shown types. The L_2 norm of the FD of path and orientations is calculated separately for the B-spline motion and mechanism motion to quantify the motion similarity. This information is provided in Table 9, and it can be seen that the resulting mechanisms have similar paths and orientations to the input motion. This is attributed to the VAE's ability to map an input motion to a latent representation of the four-bar motions that are most similar to it.

We have shown that the algebraic algorithm is sensitive to small changes to the input, while our ML pipeline is not. For arbitrary poses that a planar four-bar mechanism cannot satisfy in the prescribed order, the motion synthesis algorithm in MotionGen may produce a mechanism that satisfies a few of the poses ignoring their order. The resulting mechanism is often defective and non-Grashof. To demonstrate this point, we randomly generate five poses, found in Table 8. These poses are used as input to the motion synthesis algorithm in MotionGen which yields a single non-Grashof four-bar mechanism with the first configuration satisfying poses 1 and 4, in that order. The second configuration of the mechanism satisfies 2, 3, and 5, in that order. Cubic B-spline interpolation is used to generate a closed motion of 360 poses from the five poses and used as input to our ML pipeline. To determine whether this arbitrary motion is achievable by a four-bar mechanism, we compute I_0 which comes out to 900306.43. This approximation error is very large which proves that this motion is not achievable by a four-bar mechanism. Therefore, we filter resulting mechanisms by the L_2 norm of FDs of path, using a generous cutoff value of 1.5. The input motion's path has a crossover, similar to that in an infinity shape, so some four-bar mechanisms with infinity-shaped motions were chosen to display. Other motions that have a similar curvature to the path generated by interpolating poses 2 to 5 are also shown in Fig. 12 below the dotted line.

9 Conclusions and Future Work

In this article, we presented a generative model that utilizes a VAE and a C-VAE for the generation of planar four-bar mechanisms of different types approximating an input motion. A normalization procedure for making the input motion invariant to certain affine transformations was reviewed. We used **p** vectors to represent the mechanism parameters and type, and the C-VAE is

trained to generate new **p** vectors using a latent representation of the normalized motion as the condition for supervised learning. We use the Fourier descriptors to find similarity between the path and orientation of the input motion and generated the mechanism's motion. We have shown that this method can generate planar four-bar mechanisms of three different types (RRRR, RRRP, RRPR) approximating an input four-bar motion. Furthermore, we have shown that our model is not susceptible to small variations in an input motion, as compared to the algebraic fitting algorithm, for producing diverse and defect-free mechanisms. Finally, we have shown that this method can handle arbitrary input motions that cannot be produced by four-bar mechanisms.

We have also shown that neural networks can learn the quadratic conditions, shown in Eq. (6), required for the **p** coefficients of the G-manifold equation to represent the geometric parameters of a planar dyad. Therefore, this work can be expanded to planar mechanisms with link configurations greater than four-bar since the G-manifold is easier to derive than the quadratic conditions, which are inferred through observation. Also, for path generation and motion generation, **p** vectors are sufficient as a single representation of mechanism parameters that also encode the type. Our ML pipeline produces good starting approximations, and using an optimization algorithm, the resulting mechanisms can be improved to get the exact input motion or a motion with greater similarity.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Kerle, H., Corves, B., Mauersberger, K., and Modler, K.-H., 2011, *The Role of Mechanism Models for Motion Generation in Mechanical Engineering*, Springer, Dordrecht, pp. 107–120.

[2] McCarthy, J. M., and Soh, G. S., 2010, *Geometric Design of Linkages*, Vol. 11, Springer, New York.

[3] Erdman, A. G., and Sandor, G. N., 1991, *Advanced Mechanism Design: Analysis and Synthesis*, 2nd ed., Vol. 2, Prentice-Hall, Englewood Cliffs, NJ.

[4] Hunt, K. H., 1978, *Kinematic Geometry of Mechanisms*, Clarendon Press, Oxford.

[5] Hartenberg, R. S., and Denavit, J., 1964, *Kinematic Synthesis of Linkages*, McGraw-Hill, New York.

[6] Suh, C. H., and Radcliffe, C. W., 1978, *Kinematics and Mechanism Design*, John Wiley and Sons, New York.

[7] Lohse, P., 2013, *Getriebesynthese: Bewegungsabläufe ebener Koppelmechanismen*, Springer-Verlag, Heidelberg, Germany.

[8] Burmester, L., 1886, *Lehrbuch der Kinematik*, Verlag Von Arthur Felix, Leipzig.

[9] Deshpande, S., and Purwar, A., 2017, "A Task-Driven Approach to Optimal Synthesis of Planar Four-Bar Linkages for Extended Burmester Problem," ASME J. Mech. Rob., **9**(6), p. 061005.

[10] Simionescu, P. A., 1999, "Contributions to the Optimum Synthesis of Linkage-Mechanisms With Applications," Ph.D. dissertation, Politehnica University of Bucharest, Bucharest, Hungary.

[11] Chase, T. R., and Mirth, J. A., 1993, "Circuits and Branches of Single-Degree-of-Freedom Planar Linkages," ASME J. Mech. Des., **115**(2), pp. 223–230.

[12] Ge, Q., Purwar, A., Zhao, P., and Deshpande, S., 2017, "A Task-Driven Approach to Unified Synthesis of Planar Four-Bar Linkages Using Algebraic Fitting of a Pencil of G-Manifolds," ASME J. Comput. Inf. Sci. Eng., **17**(3), p. 031011.

[13] Zhao, P., Li, X., Purwar, A., and Ge, Q. J., 2016, "A Task-Driven Unified Synthesis of Planar Four-Bar and Six-Bar Linkages With R- and P-Joints for Five-Position Realization," ASME J. Mech. Rob., **8**(6), p. 061003.

[14] Li, X., Zhao, P., Purwar, A., and Ge, Q. J., 2017, "A Unified Approach to Exact and Approximate Motion Synthesis of Spherical Four-Bar Linkages Via Kinematic Mapping," ASME J. Mech. Rob., **10**(1), p. 011003.

[15] Ge, X., Purwar, A., and Ge, Q. J., 2016, "From 5-SS Platform Linkage to Four-Revolute Jointed Planar, Spherical and Bennett Mechanisms," Volume 5B: 40th Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Charlotte, NC, Aug. 21–24.

[16] Hayes, M., and Zsombor-Murrary, P., 2004, "Towards Integrated Type and Dimensional Synthesis of Mechanisms for Rigid Body Guidance," Proceedings of the CSME Forum, London, ON, June 1–4, pp. 53–61.

[17] Ravani, B., and Roth, B., 1983, "Motion Synthesis Using Kinematic Mappings," J. Mech. Transm. Autom. Des., **105**(3), pp. 460–467.

[18] Bottema, O., and Roth, B., 1979, *Theoretical Kinematics*, Dover Publication Inc., New York.

[19] McCarthy, J. M., 1990, *Introduction to Theoretical Kinematics*, The MIT Press, Cambridge, MA.

[20] Larochelle, P., 2000, *Approximate Motion Synthesis Via Parametric Constraint Manifold Fitting*, Kluwer Academic Publishers, Dordrecht, pp. 103–110.

[21] Levitskii, N., 1950, "Design of Plane Mechanisms With Lower Pairs," Proceedings of the AH CCCP Izdatelstvo Akademii Nauk, Leningrad, Moscow, pp. 215–217.

[22] Modler, K.-H., and Lin, S., 1997, "General Solution of Three-Position Problem With Unsharpened Position," Proceedings of the International Conference on Mechanical Transmissions and Mechanisms (MTM'97), MTM'97, Tianjin, July 1–4, China Machine Press, Beijing, pp. 369–373.

[23] Modler, K.-H., and Lin, S., 1999, "Mechanism Synthesis Using Geometrical Similarity Transformation," *Advances in Multibody Systems and Mechatronics*, Institute of Mechanics and Gear Systems, Graz, pp. 309–318.

[24] Zimmerman, R., 2018, "Planar Linkage Synthesis for Mixed Motion, Path and Function Generation Using Poles and Rotation Angles," ASME J. Mech. Rob., **10**(2), p. 025004.

[25] Yao, J., and Angeles, J., 2000, "Computation of All Optimum Dyads in the Approximate Synthesis of Planar Linkages for Rigid-Body Guidance," Mech. Mach. Theory, **35**(8), pp. 1065–1078.

[26] Chen, C., Bai, S., and Angeles, J., 2008, "A Comprehensive Solution of the Classical Burmester Problem," Trans. CSME, **32**(2), pp. 137–154.

[27] Al-Widyan, K., Cervantes-Sanchez, J., and Angeles, J., 2002, "A Numerically Robust Algorithm to Solve the Five-Pose Burmester Problem," ASME Paper No. DETC2002/MECH-34270.

[28] Simionescu, P. A., 2022, "A Revisit of the Planar Four-Bar Linkage Synthesis Problem for Two and Three Input–Output Positions," 2nd USCToMM Symposium on Mechanical Systems and Robotics, Rapid City, SD, May 19–21, pp. 188–197.

[29] Cervantes-Sánchez, J. J., Rico, J., Garcia-Murillo, M., and Núñez Altamirano, D., 2023, "On the Exact Motion Synthesis of Planar Four-Bar-Linkages With Prismatic and Revolute Joints," Mech. Mach. Theory, **190**, p. 105432.

[30] Xu, T., Myszka, D. H., and Murray, A. P., 2024, "A Bi-invariant Approach to Approximate Motion Synthesis of Planar Four-Bar Linkage," Robotics, **13**(1), p. 13.

[31] Purwar, A., Deshpande, S., and Ge, Q. J., 2017, "MotionGen: Interactive Design and Editing of Planar Four-Bar Motions Via a Unified Framework for Generating Pose- and Geometric-Constraints," ASME J. Mech. Rob., **9**(2), p. 024504.

[32] Mechanismic Inc., 2022, "MotionGen," http://www.motiongen.io

[33] Chang, W.-T., Lin, C.-C., and Wu, L.-I., 2005, "A Note on Grashof Theorem," J. Mar. Sci. Technol., **13**(4), p. 1.

[34] Deshpande, S., and Purwar, A., 2019, "A Machine Learning Approach to Kinematic Synthesis of Defect-Free Planar Four-Bar Linkages," ASME J. Comput. Inf. Sci. Eng., **19**(2), p. 021004.

[35] Deshpande, S., and Purwar, A., 2019, "Computational Creativity Via Assisted Variational Synthesis of Mechanisms Using Deep Generative Models," ASME J. Mech. Des., **141**(12), p. 121402.

[36] Purwar, A., and Ge, Q. J., 2025, "Advancements in Kinematic Synthesis: Data-Driven Methods for Mechanism Design," ASME J. Comput. Inf. Sci. Eng., **25**(12), p. 121002.

[37] Ge, Q. J., Zhao, P., Purwar, A., and Li, X., 2012, "A Novel Approach to Algebraic Fitting of a Pencil of Quadrics for Planar 4R Motion Synthesis," ASME J. Comput. Inf. Sci. Eng., **12**(4), p. 041003.

[38] Lyu, Z., Purwar, A., and Liao, W., 2024, "A Unified Real-Time Motion Generation Algorithm for Approximate Position Analysis of Planar N-Bar Mechanisms," ASME J. Mech. Des., **146**(6), p. 063302.

[39] Purwar, A., and Chakraborty, N., 2023, "Deep Learning-Driven Design of Robot Mechanisms," ASME J. Comput. Inf. Sci. Eng., **23**(6), p. 060811.

[40] Haykin, S., 1994, *Neural Networks*, Vol. 2, Prentice Hall, New York.

[41] Cybenko, G., 1989, "Approximation by Superpositions of a Sigmoidal Function," Math. Control Signal. Syst., **2**(4), pp. 303–314.

[42] Hornik, K., Stinchcombe, M. B., and White, H. L., 1989, "Multilayer Feedforward Networks Are Universal Approximators," Neural Netw., **2**(5), pp. 359–366.

[43] Pearce, M. T., Dooms, T., Rigg, A., Oramas, J. M., and Sharkey, L., 2025, "Bilinear MLPs Enable Weight-Based Mechanistic Interpretability," International Conference on Representation Learning, Singapore, Apr. 24–28.

[44] Zeng, C., Wang, J., Shen, H., and Wang, Q., 2024, "KAN Versus MLP on Irregular or Noisy Functions," preprint arXiv:2408.07906.

[45] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M., 2024, "KAN: Kolmogorov-Arnold Networks," https://arxiv.org/abs/2404.19756

[46] Kingma, D. P., and Welling, M., 2014, "Auto-encoding Variational Bayes," Computing Research Repository, abs/1312.6114.

[47] Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A., 2016, "beta-VAE: Learning Basic Visual Concepts With a Constrained Variational Framework," International Conference on Learning Representations, Toulon, France, Apr. 24–26, pp. 60–74.

[48] Bahdanau, D., Cho, K., and Bengio, Y., 2014, "Neural Machine Translation by Jointly Learning to Align and Translate," International Conference on Learning Representations, San Diego, CA, May 7–9.
[49] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E., 2016, "Hierarchical Attention Networks for Document Classification," Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, June 13–15.
[50] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L., 2018, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, June 18–23, pp. 6077–6086.
[51] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y., 2015, "Attention-Based Models for Speech Recognition," 29th International Conference on Neural Information Processing Systems, Montreal, Canada, Dec. 7–12.
[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I., 2017, "Attention Is All You Need," CoRR abs/1706.03762, http://arxiv.org/abs/1706.03762
[53] Ba, J. L., Kiros, J. R., and Hinton, G. E., 2016, "Layer Normalization," arXiv.
[54] Agarap, A. F., 2018, "Deep Learning Using Rectified Linear Units (ReLU)," CoRR.
[55] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting," J. Mach. Learn. Res., **15**(1), pp. 1929–1958.
[56] Pearce, T., Brintrup, A., and Zhu, J., 2021, "Understanding Softmax Confidence and Uncertainty," CoRR, abs/2106.04972, https://arxiv.org/abs/2106.04972
[57] Blealtan, 2024, "Efficient-KAN: Efficient Implementation of Kolmogorov-Arnold Networks," https://github.com/Blealtan/efficient-kan, Accessed: Aug. 29, 2024.
[58] Sohn, K., Lee, H., and Yan, X., 2015, "Learning Structured Output Representation Using Deep Conditional Generative Models," 29th International Conference on Neural Information Processing Systems, Montreal, Canada, Dec. 7–12, pp. 3483–3491.
[59] Sener, S., and Unel, M., 2006, "Geometric Invariant Curve and Surface Normalization," Third international conference on Image Analysis and Recognition, Póvoa de Varzim, Portugal, Sept. 18–20, pp. 445-456.
[60] Hyvärinen, A., 1999, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis," IEEE Trans. Neural Netw., **10**(3), pp. 626–634.
[61] Li, X., Wu, J., and Ge, Q. J., 2016, "A Fourier Descriptor-Based Approach to Design Space Decomposition for Planar Motion Approximation," ASME J. Mech. Rob., **8**(6), p. 064501.
[62] Mcgarva, J., and Mullineux, G., 1993, "Harmonic Representation of Closed Curves," Appl. Math. Model., **17**(4), pp. 213–218.
[63] Mcgarva, J. R., 1994, "Rapid Search and Selection of Path Generating Mechanisms From a Library," Mech. Mach. Theory, **29**(2), pp. 223–235.
[64] Ullah, I., and Kota, S., 1997, "Optimal Synthesis of Mechanisms for Path Generation Using Fourier Descriptors and Global Search Methods," ASME J. Mech. Des., **119**(4), pp. 504–510.
[65] Developers, S., 2024, "SciPy Interpolate Documentation," https://docs.scipy.org/doc/scipy/reference/interpolate.html