ASME Journal of Mechanical Design
Online journal at:
https://asmedigitalcollection.asme.org/mechanicaldesign

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

Anar Nurizada
Department of Mechanical Engineering,
Computer-Aided Design and Innovation Lab,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: anar.nurizada@stonybrook.edu

Rohit Dhaipule
Department of Computer Science,
Computer-Aided Design and Innovation Lab,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: rohit.dhaipule@stonybrook.edu

Zhijie Lyu
Department of Mechanical Engineering,
Computer-Aided Design and Innovation Lab,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: zhijie.lyu@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

In recent years, there has been a strong interest in applying machine learning techniques to path synthesis of linkage mechanisms. However, progress has been stymied due to a scarcity of high-quality datasets. In this article, we present a comprehensive dataset comprising nearly three million samples of 4-, 6-, and 8-bar linkage mechanisms with open and closed coupler curves. Current machine learning approaches to path synthesis also lack standardized metrics for evaluating outcomes. To address this gap, we propose six key metrics to quantify results, providing a foundational framework for researchers to compare new models with existing ones. We also present a variational autoencoder-based model in conjunction with a k-nearest neighbor search approach to demonstrate the utility of our dataset. In the end, we provide example mechanisms that generate various curves along with a numerical evaluation of the proposed metrics.
[DOI: 10.1115/1.4067014]

Keywords: planar 4- 6- and 8-bar linkage, path synthesis, variational autoencoder, neural networks, machine learning, deep learning, dataset generation, data-driven design, deep generative models, artificial intelligence, computational kinematics, computer-aided engineering, data-driven design, generative design, linkages, mechanism synthesis

1 Introduction

Path-generating linkage mechanisms are fundamental in mechanical design due to their ability to translate rotational or linear input into specific, desired paths. These mechanisms play a crucial role across various industries, offering precision, reliability, and efficiency in creating complex motions. In industrial automation and robotics, path-generating linkages are essential for tasks that require high precision and consistency [1]. Robotic arms and automated machinery often rely on these mechanisms to execute repetitive motions like welding, cutting, or assembly with unparalleled accuracy. In the medical field, particularly in the design of prosthetics and rehabilitation devices, path-generating linkages are used to replicate or assist natural human motion [2,3]. These mechanisms can be customized to follow the natural gait of a patient or facilitate specific therapeutic exercises, thereby significantly improving patient outcomes. Agricultural machinery also benefits from path-generating linkages. Machines like harvesters and planters rely on these mechanisms to precisely control implements, ensuring optimal planting patterns, efficient harvesting, and minimal soil disruption [4]. In the realm of education and research, path-generating linkages serve as vital tools for teaching kinematics and mechanical design [5]. Historically, path-generating linkages have had significant cultural and artistic value, particularly in the creation of automata and mechanical art [6].

Given the broad range of applications, it is critical to design mechanisms that can generate desired paths efficiently and effectively. Kempe's Universality Theorem [7] asserts that any algebraic curve can be exactly traced by a specifically designed linkage, regardless of the complexity or practicality of the resulting mechanism. Nolle [8,9] and Koetsier [10,11] have provided a detailed historical account of path-generating mechanisms. More recently, Liu and McCarthy [12,13] have presented analytical methods for synthesizing simpler linkage mechanisms, which draw algebraic curves that have finite Fourier series parameterizations. Traditional path synthesis methods for single-degree-of-freedom linkage 4-bar linkage mechanisms discretize coupler curves into finitely separated points to determine linkage parameters, achieving exact solutions for up to nine points [14]. For more than nine points, optimization methods become necessary. These methods, while common, have limitations such as slow performance, dependence on initial

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received July 8, 2024; final manuscript received October 13, 2024; published online November 18, 2024. Assoc. Editor: Faez Ahmed.

conditions, and lack of guaranteed outcomes [15]. These methods focus on passing through finite precision points, often missing the overall shape of the path. Structural error objectives mischaracterize the design problem by requiring simultaneous optimization of the coupler curve's shape, size, orientation, and position. Typically, these methods yield a single mechanism at a high computational cost, limiting design flexibility. Lee and Russell [16] provide an extensive list of references on the topic of path generation by 4-bar mechanisms in a recently published survey article.

An emerging approach to path synthesis of linkage mechanisms has been to use machine learning (ML) to model the relationship between mechanism parameters and coupler curves. The most popular ML models used thus far employ artificial neural networks (ANNs) to map coupler curves to mechanisms. Hoskins and Kramer [17] were the first to introduce ANNs for synthesizing mechanical linkages based on user-specified curves. They employed radial basis function ANNs for 4-bar linkages, optimizing designs numerically, which underscored the effectiveness of ANNs in complex system inverse modeling.

In parallel, researchers explored alternative representations of input coupler curves such as Fourier descriptors (FD) [18] and wavelets [19]. These representations facilitated the development of neural networks to map curve features to mechanism parameters. Vasiliu and Yannou [20] introduced FD-based representation and trained neural networks for mechanism parameter approximation. Galan-Marin et al. [21] utilized wavelet-based representation for optimal crank-rocker mechanism design.

Recent advancements have leveraged deep generative models to synthesize diverse mechanisms for given coupler curves. Deshpande and Purwar [22,23] introduced image-based variational autoencoders (VAEs) for path synthesis, capable of generating multiple design solutions. Sharma and Purwar [24] extended this to spatial platform mechanisms for defect-free synthesis. Nurizada and Purwar [25] investigated the synthesis of coupler curves in planar mechanisms through deep neural networks. They compared Fourier descriptors, wavelets, point coordinates, and image representations, aiming to unify coupler curve representations using VAEs. Their approach involved training a fully connected neural network to map latent representations of coupler curves to mechanism parameters, demonstrating the versatility and effectiveness of deep learning in mechanism design.

Fogelson et al. [26] utilized deep reinforcement learning (DRL) to design linkages with up to 16 bars using revolute joints, focusing on path synthesis. Their methodology employed a DRL algorithm to optimize linkage configurations for different path specifications, highlighting advancements in synthesizing complex mechanisms through modern machine learning techniques.

The potential for deep generative machine learning models to revolutionize design engineering is evident in recent studies. A review by Regenwetter et al. [27] highlights the impact of deep generative models, such as feedforward neural networks, generative adversarial networks (GANs) [28], VAEs [29], and deep reinforcement learning, on structural optimization, materials design, kinematic, and shape synthesis. A position article on deep learning design of robot mechanisms by Purwar and Chakraborty [30] highlights problems and challenges related to representation, mapping, and model architecture design. Additionally, a study on high-quality synthetic datasets emphasizes guidelines for generating, annotating, and validating these datasets to ensure diversity, utility, and realism [31]. This study illustrates practical implications through the creation of a turbo-compressors dataset, highlighting the necessity of comprehensive datasets for effective artificial intelligence (AI) applications in engineering design.

Nobari et al. [32] introduced the LINKS dataset, comprising 100 million coupler curves. This dataset excluded prismatic joints and processed only fully dyadic decomposable mechanisms, thereby omitting certain critical mechanism types to be discussed later in this article. Moreover, it only features closed curves. To address these gaps, this article introduces a dataset of 3 million mechanisms that includes four types of 4-bar mechanisms with both revolute and prismatic joints and all revolute-jointed 6-bar and 8-bar mechanisms generating both open and closed curves.

A significant hurdle in advancing machine learning-based approaches for path synthesis is the difficulty of comparing new models with existing state-of-the-art approaches due to the lack of standardized comparison techniques. To address this, we propose six metrics: accuracy, novelty, diversity, robustness, computational efficiency, and scalability. We expect that these metrics may enable a comprehensive evaluation and comparison of different approaches in the future.

To demonstrate the practical application of our dataset, we extend the methodology introduced by Deshpande and Purwar [23], where coupler curves are standardized and converted into image representations. These images serve as inputs to a VAE, which learns the probability distribution of latent features, allowing it to generate diverse samples of new curves. Unlike traditional methods that produce a single optimal solution, our approach generates a range of plausible solutions. By combining with a k-nearest neighbor (k-NN) search, we produce well-formed and valid mechanisms all the time. Nonetheless, the dataset is general enough to allow researchers to use it with any model of their choice.

Figure 1 provides an overview of our method. There are two main processes depicted in this figure. The first process involves the training phase, where raw coupler curves are normalized and digitized into 64×64-pixel black-and-white images. These images are then processed through a VAE model. Specifically, the encoder maps the images into a latent space, while the decoder reconstructs the images from their latent representations. During training, both the reconstruction loss and KL divergence loss are minimized, ensuring that the model accurately captures the underlying structure of the coupler curves. The latent space clusters similar-looking coupler curves together, which enables sampling during inference.

The second process, depicted as the inference phase, utilizes this latent space for query and generation. Given an input coupler curve, it is normalized, discretized, and embedded as an image before being processed by the encoder to obtain its latent representation. We then perform a k-NN search within the latent space to identify k mechanisms from the dataset, which includes 4-bar, 6-bar, and 8-bar linkages, that best approximate the input coupler curve. Mechanisms whose resulting coupler curves closely resemble the input are retained, while those that either do not approximate the input well or are repetitive are filtered out. This ensures that the presented mechanisms are both relevant and diverse.

We trained six VAE models with varying latent dimensions—1, 2, 5, 10, 25, and 50—to determine the minimal dimension required for satisfactory results. Experimental analysis revealed that a latent dimension of 10 yields satisfactory performance, as determined by the Chamfer distance (CD) [33] metric. Specifically, if CD values were less than 0.2, the results were deemed successful. This threshold CD value was chosen heuristically from the scale of the curve embedded inside the image space. Finally, we conclude our study by presenting results obtained from predicting mechanisms for various coupler curves. While our approach is based on using VAEs, the compiled dataset can be leveraged to train other generative models, such as GANs [28] and contrastive language-image pretraining [34].

In summary, the key contributions of this work are: (1) introducing a comprehensive dataset that includes 4-bar, 6-bar, and 8-bar mechanisms generating both open and closed curves, while incorporating prismatic joints for 4-bar mechanisms, filling gaps left by previous datasets, (2) proposing six standardized metrics—accuracy, novelty, diversity, robustness, computational efficiency, and scalability—to evaluate machine learning-based approaches for path synthesis, (3) utilizing VAEs to represent a probabilistic distribution of coupler curves, enabling the generation of diverse and plausible solutions rather than focusing solely on optimal outcomes, and finally (4) identifying the optimal latent dimension for VAE models, ensuring robust performance in path synthesis applications.

The remainder of this article is organized as follows: Sec. 2 details the process and considerations involved in generating our dataset. Section 3 introduces essential concepts of the variational

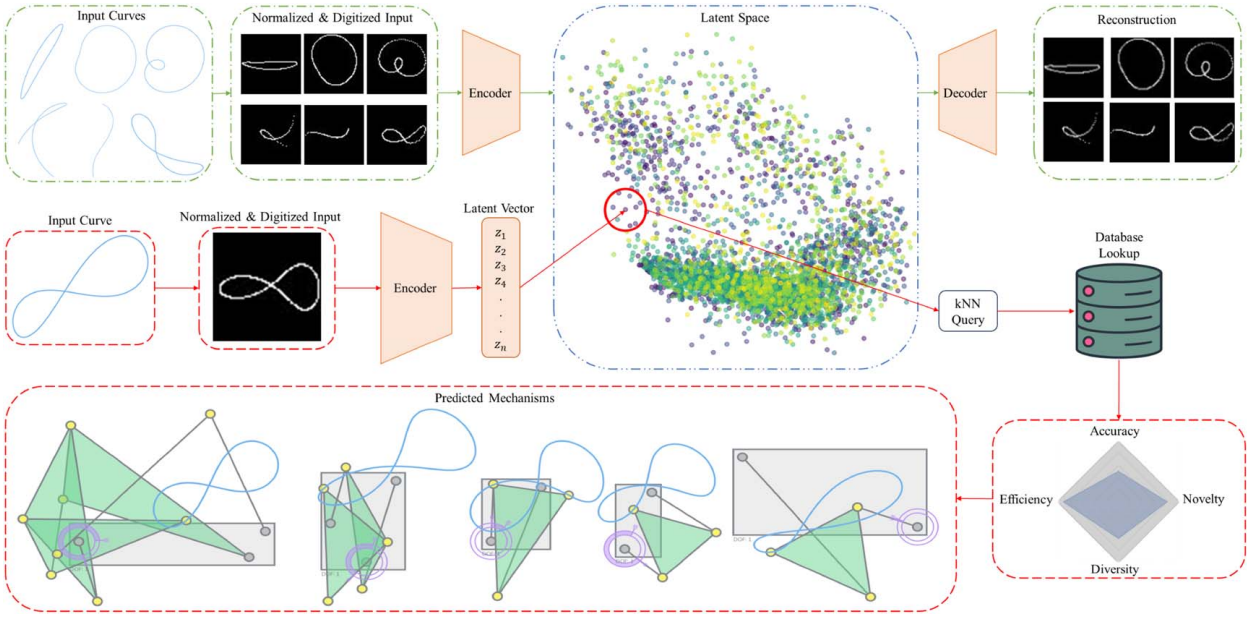

Fig. 1 Overview: the training phase and inference phase steps are bounded by green dash-dot lines and red dashed lines, respectively

autoencoder relevant to our methodology and describes our model's architecture. Section 4 outlines the evaluation metrics employed to assess our results and presents the findings from comparing different VAE models. Section 5 demonstrates the application of our model to different input coupler curves, illustrating its performance and results.

2 Dataset Generation

Our dataset is developed using a simulator capable of producing coupler curves for various one-degree-of-freedom mechanisms. For this purpose, we leveraged our unified simulation algorithm designed for planar n-bar mechanisms, which can handle both revolute and prismatic joints [35]. In this section, we provide a broad overview of the algorithm, while inviting readers to read the original paper for an in-depth explanation.

This algorithm is designed to streamline the real-time kinematic simulation of planar n-bar mechanisms, accommodating both revolute and prismatic joints. The algorithm introduces a novel approach to handle the complexities of different joint types by transforming all prismatic joints into equivalent revolute joints. This transformation is based on the idea that linear motion, typical of prismatic joints, can be closely approximated by a large-radius arc produced by a revolute joint. By adjusting the distance of this equivalent revolute joint, the algorithm ensures that the approximation is accurate, with minimal error.

To represent the mechanism, the algorithm uses a series of matrices that encode the relationships between joints and links. These include matrices that define which links are connected by which joints, the lengths of the links and the constraints imposed by slots and actuators. The various constraints, such as maintaining constant distances between connected joints or ensuring specific rotational relationships between links, are all formulated as generalized equations.

For the actual kinematic analysis, the algorithm employs two types of solvers. The first is a dyadic decomposition method, which is highly efficient and suitable for simpler mechanisms such as the ones shown in Fig. 2. This method breaks down the problem into smaller, easily solvable pairs of links and joints. However, for more complex mechanisms that cannot be decomposed into simple pairs, the algorithm turns to an optimization-based solver; see Fig. 3 for a few examples of complex mechanisms. This solver, using iterative methods like Newton–Raphson or Levenberg–Marquardt, is more computationally intensive but capable of handling mechanisms of any complexity.

The algorithm also incorporates mobility analysis, checking the degrees of freedom to ensure that the mechanism is properly constrained. It further simplifies the system by identifying and merging rigid subassemblies of links, reducing the number of variables, and making the problem easier to solve. The mechanism is then decomposed into smaller subproblems, which are solved sequentially to produce the overall solution.

As the mechanism is simulated across multiple states—each representing a different configuration—the algorithm compiles the results into a position tensor that tracks the positions of all joints over time. This iterative position analysis allows for a comprehensive understanding of the mechanism's motion.

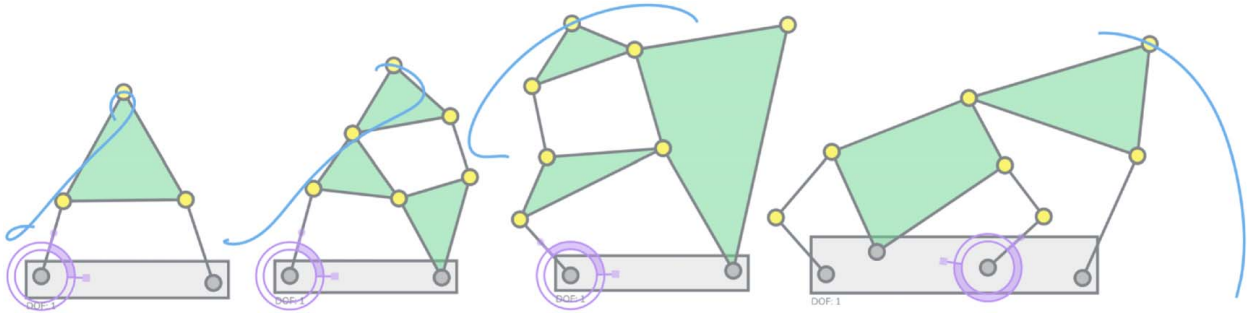

Fig. 2 Examples of mechanisms that can be simulated using dyadic decomposition (or, arc intersection) only

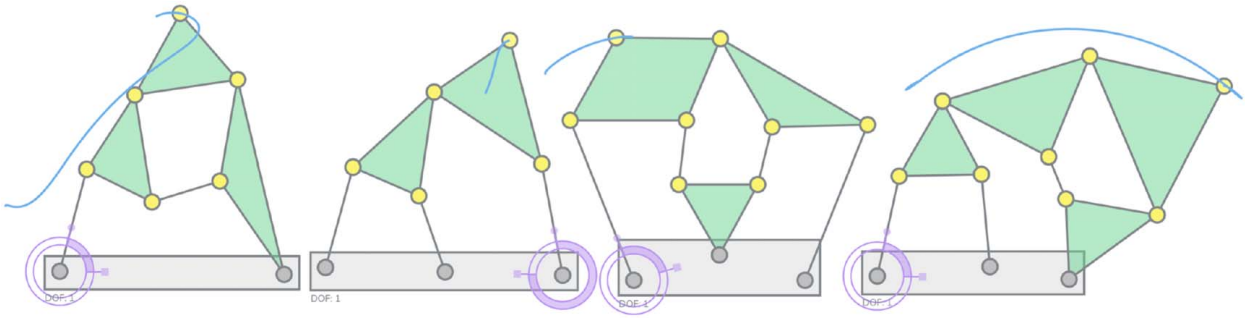

Fig. 3 Examples of mechanisms that require both arc intersection and optimization methods for simulation

Finally, given a mechanism's kinematic structure, we use randomly initialized joint locations (coordinates ranging from −10 to 10) to create the initial state of mechanisms. In this article, a set of joint locations includes the location of a coupler (or, tracer) point as we consider them to be a potential joint location irrespective of the fact that nothing may be joined at that point. To ensure the generation of useful mechanisms, we apply several filters: the ratio of maximum-to-minimum link length must be less than 5 for 4- and 6-bar mechanisms (10 for 8-bar mechanisms); joints must not overlap; and the rotary actuator must have at least a 20-deg range of actuation. In our approach, we decided to filter out mechanisms with a large maximum-to-minimum link size ratio, as such configurations are typically impractical. Limiting this ratio in a linkage mechanism is a well-known design practice that balances the trade-offs between kinematic feasibility, structural integrity, dynamic performance, and practical considerations like manufacturing and assembly. Since the coupler point is not allowed to be co-located with a moving joint of a dyad, it prohibits the generation of simple circular curves, which can easily bias the dataset.

The initialization logic employs a randomized approach rather than a fixed, discrete one because this method provides greater flexibility in exploring the design space and avoids the limitations imposed by a predefined grid. While a fixed grid, such as a 5×5 configuration with 25 potential joint positions, would result in a large number of configurations—2,422,728,000 for a 6-bar mechanism with six joints and a coupler point—this number escalates exponentially for more complex mechanisms. A randomized approach allows for a broader range of possible configurations, accommodating more complex topologies without the constraints of a fixed grid which does not need to be exhaustively sampled. Moreover, it enables the generation process to be efficiently terminated once the desired number of mechanisms are simulated.

In this study, we have successfully generated a comprehensive dataset comprising a variety of mechanisms. Specifically, our dataset comprises 4 distinct types of 4-bar mechanisms, 21 unique types of 6-bar mechanisms, and 153 different types of 8-bar mechanisms. These variations arise from altering the positions of the ground links and actuators. This extensive compilation results in a total of approximately 3 million individual mechanisms, each accompanied by their corresponding paths. This rich dataset serves as a valuable resource for analyzing and understanding the complex behaviors and trajectories of these mechanical systems, providing a robust foundation for further research and development in the field of mechanism design and analysis. Figure 4 shows the topologies of mechanisms used in the dataset. Two of the 4-bar mechanism topologies (not shown) resulted in simple arc-shaped coupler curves and were therefore excluded from the dataset. Using our approach, generating paths for 10,000 nonfully dyadic-decomposable mechanisms can take 150–300 s, where the exact time is determined by the specific kinematic structure. For 10,000 fully dyadic-decomposable ones, the simulation time is 10–20 s. Other statistical details for the kinematic structures in the dataset are presented in Table 1.

Previous related work by Nobari et al. [32] has demonstrated similar efforts. In their research, they developed a dataset featuring fully dyadic-decomposable linkages for mechanisms with up to 14 bars and 20 joints. Although our dataset includes mechanisms up to only 8 bars, it also encompasses one-degree mechanisms that are nondyadic decomposable, such as the double butterfly mechanism. Additionally, our dataset includes 4-bar mechanisms with prismatic joints as well as ones that generate open paths.

2.1 Normalization of Coupler Curves. It is well-known that linkage mechanisms with the same link ratio produce geometrically similar coupler curves. Thus, before they are embedded inside an image, we normalize them and store the normalization (or, transformation) matrix for post-processing. Specifically, the curve normalization includes translation, rotation, scaling, and reflection. Given a curve by a set of (x, y) points, we follow the normalization method presented by Deshpande and Purwar [23] for translation, rotation, and scaling. We improved their normalization method by implementing the reflection correction from Ref. [36] for 2D curves.

For translation, the center of the point cloud is set to (0,0). Next, we find the two principal axes for this point cloud and align the first axis to the X-axis, which results in the rotation invariance. Then, we isotropically scale the point cloud so that it takes up the rectangular capturing window as much as possible. Finally, we use the third-order central moment of these points to determine the reflection matrix according to Ref. [36]. This allows us to always get the same normalization result for the same point cloud in different positions, sizes, and orientations.

2.2 Using the Dataset. Our dataset is organized into two main folders. The first folder contains images of the coupler curves, providing a visual representation of each path. The second folder holds the Cartesian coordinates that correspond to these paths, stored in *.npy* files. Each *.npy* file details the exact coordinates traced by the tracer joint, complementing the visual data with precise spatial information.

The filenames in this dataset encode key details about the mechanism. Specifically, they include the joint coordinates, mechanism type, and the normalization matrix, all separated by underscores. The normalization matrix is used to scale the point locations so that the coupler curve is properly aligned within the normalized image window. A script is provided to facilitate the loading and processing of this data. This dataset can be employed for various applications, such as training generative models to create mechanisms based on a specified path or simulating a given mechanism. The dataset is available online.[2]

For example, consider a file named "−0.001_1.332_6.782_−3.289_−1.912_0.471_9.951_3.802_7.147_1.303_RRRR_0.03_−0.35_0.53_−0.35_−0.03_−0.3_.jpg." This file name encodes

[2]https://www.kaggle.com/datasets/purwarlab/four-six-and-eight-bar-mechanisms-with-curves

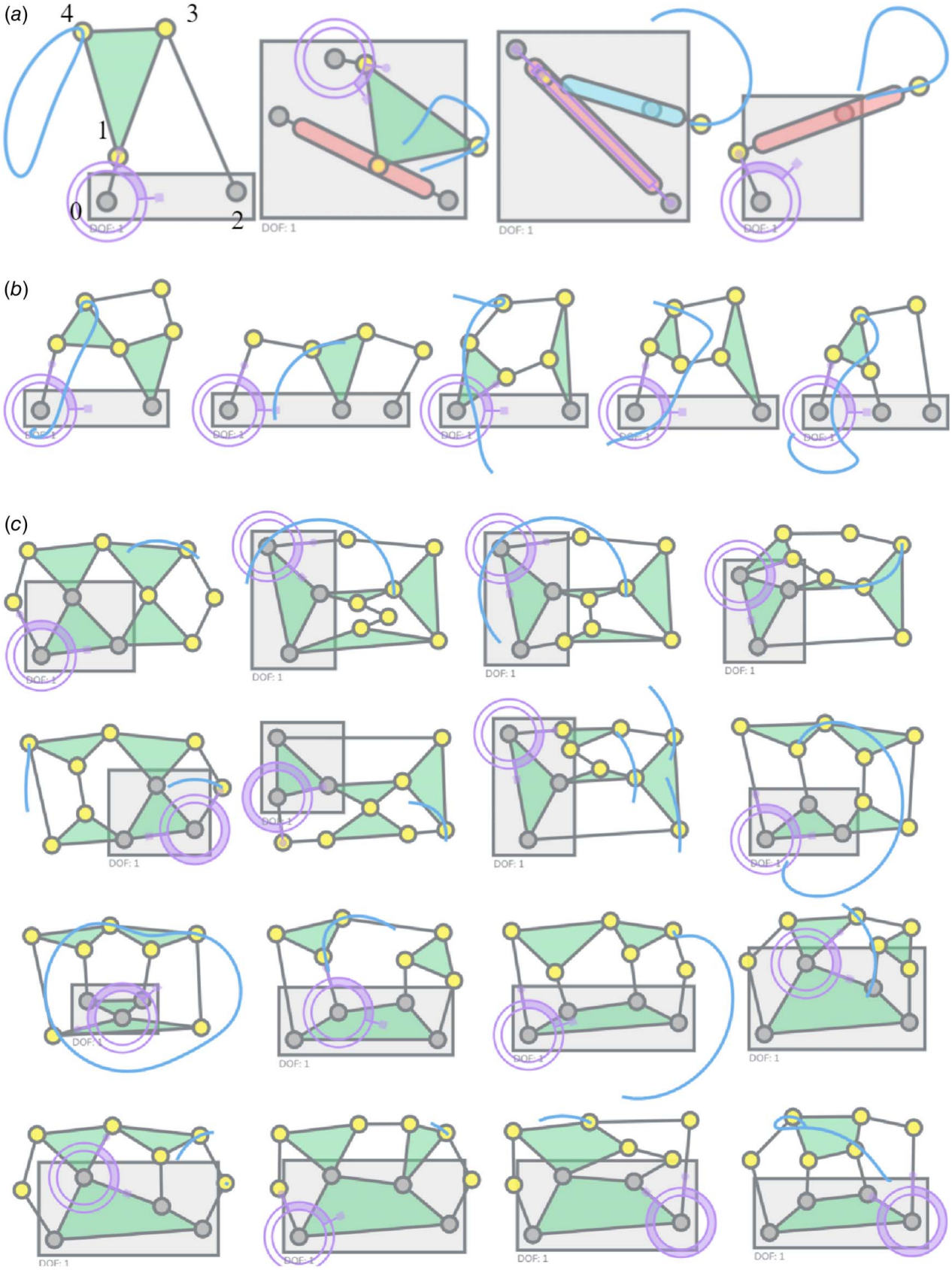

Fig. 4 Single-DOF 4-, 6-, and 8-bar mechanism types compiled in the dataset with their coupler curves. Altering the positions of the ground links and actuators, we get 21 different types of 6-bar linkages and 153 types of eight-bar linkages in our dataset. Moving joints are shown in yellow, whereas fixed joints are shown in gray. (*a*) Four types of 4-bar mechanisms: RRRR, RRRP, PRPR, and RRPR, where R and P stand for revolute and prismatic joints, respectively. (*b*) Five types of 6-bar mechanisms (from left to right): Watt I, Watt II, Stephenson I, Stephenson II, and Stephenson III, (*c*) Sixteen types of 8-bar mechanisms.

Table 1 Statistics of the dataset: complex mechanisms require optimization, while simple ones can be simulated using arc intersection

Mechanism type	# of types of mechanisms	# of complex mechanisms	Success rate of simple mechanisms (%)	Success rate of complex mechanisms (%)	# of samples
Four bars	4	0	75.84%	N/A	606,731
Six bars	21	6	24.06	0.81	397,589
Eight bars	153	84	20.44	0.66	1,967,187

Note: Success rate refers to the percentage of randomly chosen samples that generated valid mechanisms after applying the filters.

array of joint coordinates, type of mechanism, and the normalization matrix, in that order:

(1) Mechanism type: RRRR
(2) Joint coordinates:

$$\begin{bmatrix} -0.001 & 1.332 \\ 6.782 & -3.289 \\ -1.912 & 0.471 \\ 9.951 & 3.802 \end{bmatrix}$$

(3) Normalization matrix:

$$\begin{bmatrix} 0.03 & -0.35 & 0.53 \\ -0.35 & -0.3 & 0.3 \\ 0 & 0 & 1 \end{bmatrix}$$

Furthermore, we created a topology dictionary file called "BSIdict," to store each type of mechanism in the dataset with their kinematic structures, where a name string like "RRRR" corresponds to a certain kinematic structure. This structure is represented with three dimensionless matrices—**B**, **S**, and **I**—which are the incidence matrix, the slot matrix, and the actuator matrix, respectively. For example, for the RRRR linkage mechanism shown in Fig. 4(*a*), its **B** and **I** matrices are given below. Its slot matrix **S** is empty because this mechanism does not have a prismatic joint.

$$B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$I = \begin{bmatrix} 2 & 0 & 1 & 0 \end{bmatrix}$$

The **B** matrix specifies which point of interest (either a joint or the coupler point) belongs to which links and the indices of these links and points. For example, link 0 is the first row in the matrix and it contains joint 0 and point 2, and coupler link 3 contains joints 1, 3, and 4. Since a 4-bar mechanism has four links and five points of interest, the **B** matrix has four rows and five columns with their indices always starting from zero. Furthermore, the first link is always the ground link, so joints 0 and 2 are the fixed joints.

The actuator matrix **I** specifies the indices of joints defining each actuator and its type. For the RRRR mechanism shown in Fig. 4(*a*), the actuator is rotary, which controls the angle defined by three points: points 0, 1, and 2, but the order is [2,0,1] in the row because it specifies the angle between vector $\mathbf{v}_{0,2}$ and vector $\mathbf{v}_{0,1}$. The actuator's property is marked at the end of the row, where 0 means rotary and 1 means linear. See Fig. 5 for the two types of actuators.

The definitions of these three matrices follow from Ref. [35], but we have expanded the **S** matrix by one additional column for this implementation. Specifically, a slot constraint means that three points stay on the same line, so the matrix is *m*-by3, where *m* is the number of slot constraints, and the three entries in a row correspond to the three joints' indices in the *B* matrix. However, we also want to specify which link the two end joints belong to, so we added the link index in the additional column. For example, if a straight slot's two end joints are *p* and *r*, and joint *q* is in the slot, then such a straight slot constraint is represented with [*p*, *q*, *r*, *l*] in the slot matrix, where *l* is the link index. See Fig. 6 for example.

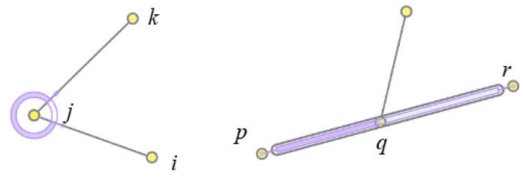

Fig. 5 An actuator is defined by four attributes: the first three are the joint indices and the last one is the actuator type. The left one is the rotary type of actuator, which determines the angle between vector $\mathbf{v}_{j,i}$ and vector $\mathbf{v}_{j,k}$. In the I matrix, this actuator is represented with [*i*, *j*, *k*, 0]. Then, the right one is the linear type of actuator, where vector $\mathbf{v}_{p,r}$ determines the actuator's direction and the length of the vector $\mathbf{v}_{p,q}$ is the actuator displacement. Correspondingly, this actuator is represented with [*p*, *q*, *r*, 1].

An additional **c** vector is used to describe which point is the coupler point for this mechanism and link structure. For example, the **c** vector for the RRRR 4-bar linkage mechanism in discussion is [0,0,0,0,1], which means that such a linkage has five joints, and only the last joint is the tracer point. We design c as a vector instead of a point index because we want to leave room for potentially adding more tracer points. Finally, given a kinematic structure, **p** is the position matrix for the initial state. If a mechanism has five points of interest (four joints and one coupler point), the size of **p** is 5×2, where the first and second rows store the *x*- and *y*-coordinates of the joints, respectively.

2.3 FAIR Principles. This dataset is designed in accordance with the FAIR Principles [37], ensuring that it is **findable**, accessible, interoperable, and reusable. Findability is facilitated by a well-organized structure and clear metadata, making it straightforward to locate specific mechanisms within the dataset. We have included a script compatible with multiple operating systems, enhancing **accessibility** by simplifying the process of loading and using the data across different operating systems. The dataset's dual-format availability (*.npy* and image) supports **interoperability** by allowing integration into diverse workflows, whether for analysis, storage, or processing. Finally, the comprehensive metadata and versatile data

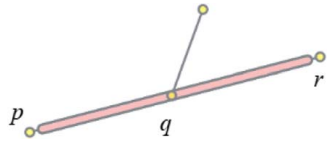

Fig. 6 A row on a slot matrix is generally [*p*, *q*, *r*, *l*], where *p*, *q*, *r* are the joint indices for the first end joint, the slot joint, and the second end joint. Then, *l* is the corresponding link index in the B matrix. In other words, $\mathbf{B}_{l,p}$ and $\mathbf{B}_{l,q}$ are 1, respectively, for the linkage system.

formats ensure that the dataset is **reusable** across various use cases, such as dataset simulation, path synthesis, and more. By adhering to these principles, our dataset promotes broad usability and long-term value for the research community.

3 Variational Autoencoder and Exploration of Its Latent Space

The VAE [29] is a generative model composed of an encoder that learns a probabilistic mapping from the input space to a latent space and a decoder, which maps an element of the latent space, also called a latent vector, to the output space. Samples from the latent space, which has a Gaussian structure, when passed through the decoder generate new data.

The loss function for a VAE is composed of two main terms: the reconstruction term and the regularization term. The reconstruction term measures the model's ability to reconstruct the input data from sampled latent vectors. It is defined as the negative log-likelihood of the data given the latent representation. Mathematically, this term is represented as $-\mathbb{E}_{q(z|x)}[\log p(x|z)]$, where $q(z|x)$ is the probability distribution of latent vectors given the input data x, and $p(x|z)$ is the probability distribution of reconstructing x from a sampled latent vector z. In this work, we use mean square error (MSE) between the input and its reconstruction as the reconstruction error.

The regularization term ensures that the latent space follows a specific distribution, typically a standard normal distribution. The Kullback–Leibler (KL) [38] divergence given by $\mathrm{KL}(q(z|x)\|p(z))$ measures the difference between the distribution $q(z|x)$ and the chosen prior distribution $p(z)$. The regularization term penalizes deviations from the chosen prior distribution, guiding the model to learn a more structured and interpretable latent space.

The overall loss function for a regular VAE is the sum of the reconstruction term and the regularization term given by

$$L_{\mathrm{VAE}} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + \mathrm{KL}(q(z|x)||p(z)) \tag{1}$$

The reconstruction error is crucial in guiding the learning process of the VAE. Minimizing this error helps the model learn a meaningful representation in the latent space and generate outputs that resemble the original input data. The regularization term in the VAE objective also ensures that the latent space is continuous and smooth, promoting better generalization. The optimization process involves adjusting the model parameters to minimize this combined loss, resulting in an effective generative model with a structured latent space.

In the context of mechanism design, the input can be the coupler curves represented as images, which are then embedded in a latent space of a much smaller dimension, thus serving as a feature or key. The outputs are also curve images similar to the input obtained through decoding. However, this model alone is insufficient to generate new mechanisms as it merely reproduces the coupler curves. Trained on 64×64-pixel black-and-white images of coupler curves, the VAE maps similar-looking curves close to each other in its latent space. It is important to note that we treat two curves—one being a closed curve and the other an open segment of the same curve—as distinct entities. Given a desired coupler curve, it is possible to map it to the latent space of the trained VAE and then explore the neighboring space to get latent representations by performing a k-NN search. Using these latent representations, we can perform a lookup in the compiled dataset and produce a large set of solutions comprising 4-bar, 6-bar, and 8-bar mechanisms. For more detailed information regarding this approach, refer to Ref. [23].

3.1 k-Nearest Neighbor Algorithm. The k-NN [39] algorithm is a straightforward and nonparametric learning algorithm. The principle behind k-NN is to make predictions for a new data point based on the labels or values of the k most similar data points in the feature space. The similarity between data points is usually determined using distance metrics. The Euclidean distance between two points $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ in n-dimensional space is used in this article and is defined as

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik}, x_{jk})^2}$$

We use the k-NN algorithm to find neighboring latent representations. k-NN is known for its simplicity and effectiveness, particularly in low-dimensional spaces. However, it can become computationally expensive as the size of the training dataset grows, and its performance can degrade with high-dimensional data due to the curse of dimensionality [40].

3.2 Neural Network Architecture of Variational Autoencoder. The architecture of the VAE used in this article is shown in Table 2. The encoder transforms input data into a latent representation by sequentially applying convolutional layers, ReLU activation [41], and max pooling operations. It consists of convolutional layers with specified filter counts and sizes, followed by max pooling to down-sample feature maps. After flattening the end feature maps into a 1D vector, fully connected layers generate a latent representation. The Decoder reconstructs data from the latent space generated by the encoder. It includes fully connected layers to process the latent vector, followed by reshaping into feature maps. Transpose convolutional layers with ReLU activation are used to up-sample and reconstruct data. A final convolutional layer with a Sigmoid activation generates the output data.

We examined VAEs with latent dimensions of 1, 2, 5, 10, 25, and 50 to understand how the latent space dimension affects the mean

Table 2 Variational autoencoder architecture

Network	Layer	Filter count	Filter size	Stride	Output shape	Activation
Encoder	Convolution 1	32	(11, 11)	(1, 1)	Batch size, 32, H, W	ReLU
	Max Pooling 1	–	(2, 2)	(2, 2)	Batch size, 32, H/2, W/2	–
	Convolution 2	64	(5, 5)	(1, 1)	Batch size, 64, H/2, W/2	ReLU
	Max pooling 2	–	(2, 2)	(2, 2)	Batch size, 64, H/4, W/4	–
	Convolution 3	128	(3, 3)	(1, 1)	Batch size, 128, H/4, W/4	ReLU
	Max pooling 3	–	(2, 2)	(2, 2)	Batch size, 128, H/8, W/8	–
	Flatten	–	–	–	Batch size, 8192	–
	Fully connected	–	–	–	Batch size, latent dim × 2	–
Decoder	Fully connected	–	–	–	Batch size, 1024	ReLU
	Reshape	–	–	–	Batch size, 16, 8, 8	–
	Transpose conv 1	128	(3, 3)	(2, 2)	Batch size, 128, 16, 16	ReLU
	Transpose conv 2	64	(5, 5)	(2, 2)	Batch size, 64, 32, 32	ReLU
	Transpose conv 3	32	(11, 11)	(2, 2)	Batch size, 32, 64, 64	ReLU
	Transpose conv 4	1	(2, 2)	(1, 1)	Batch size, 1, 64, 64	Sigmoid

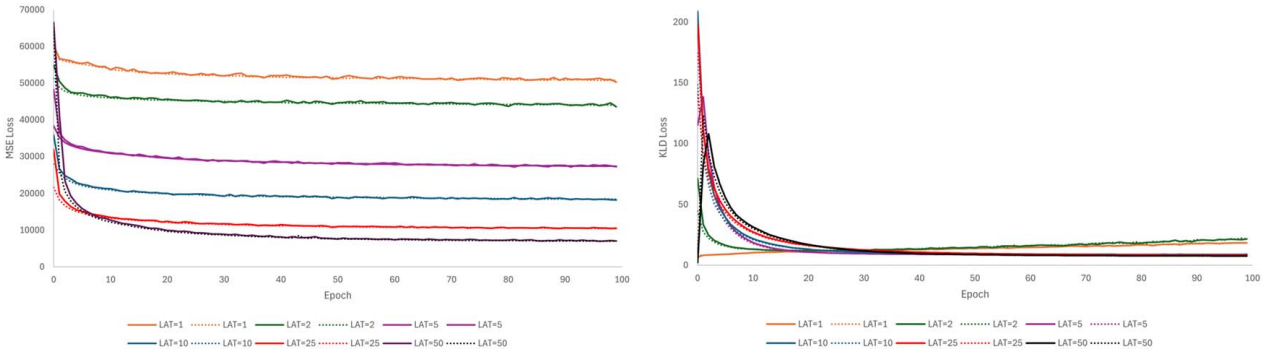

Fig. 7 Training and validation losses for VAEs with different latent dimensions. Solid lines represent training losses, while dashed lines represent validation losses.

squared error (MSE) and Kullback–Leibler Divergence (KLD) losses, as well as the overall pipeline performance. To measure the success of the trained VAE, we quantified our results systematically.

Figure 7 shows the training and validation losses for VAEs with different latent dimensions. The solid lines represent training losses, while the dashed lines represent validation losses. The left side shows MSE loss, and the right side shows KLD loss. No overfitting is observed throughout the 100 epochs of training. Both losses exhibit a clear trend: as the latent dimension increases, both final loss values decrease. During training, the MSE loss was calculated by summing batch losses, whereas the KLD loss was averaged, resulting in a significant difference in the scale of the losses. The code for the model architecture, along with the training and testing scripts, is available.[3]

4 Six-Dimensional Evaluation Metrics and Results

Although there is significant interest in applying machine learning methods to the path synthesis of mechanisms, no universal metrics exist for comparing the resulting models. In this section, we propose six distinct metrics to evaluate a model's accuracy and other characteristics and present the calculated values for each metric. While some metrics, such as accuracy, novelty, diversity, and computational efficiency, can be readily computed and quantified, others, like robustness and scalability, are more qualitative in nature and cannot be easily quantified. We note that our use of the word metric is colloquial only, not to be confused with the mathematical concept of metric.

4.1 Accuracy. The first and most important criterion is the quality of the reconstructed curves, i.e., how close they are to the input coupler curve. We consider the bidirectional CD [32,33] for evaluating the accuracy, which is a metric used to quantify the dissimilarity between two sets of points $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ in a two-dimensional space. It evaluates Euclidean distances in both directions: from each point in P to its nearest neighbor in Q, and vice versa. For each point $p_i \in P$

$$d_P(p_i) = \min_{q_j \in Q} \| p_i - q_j \|_2 \tag{2}$$

For each point $q_j \in Q$

$$d_Q(q_j) = \min_{p_i \in P} \| q_j - p_i \|_2 \tag{3}$$

The bidirectional Chamfer distance CD(P, Q) is then defined as

$$\mathrm{CD}(P, Q) = \frac{1}{m}\sum_{i=1}^{m} d_P(p_i) + \frac{1}{n}\sum_{j=1}^{n} d_Q(q_j) \tag{4}$$

where m and n are the number of points in sets P and Q, respectively. The CD measures the dissimilarity between sets P and Q based on their closest point matches in both directions, providing a comprehensive measure of overall dissimilarity in 2D space.

To calculate the accuracy of the models trained, we undertake the following steps: 1000 randomly selected coupler curves are extracted from the testing dataset. For each curve, we perform a k-NN search and predict 25 mechanisms. Subsequently, we simulate the generated coupler curves for each mechanism and compute the CD values, averaging them across all predictions. By repeating these procedures across all our trained VAEs, we ascertain the minimum latent dimension required to achieve satisfactory results. Table 3 illustrates a clear trend in our findings: as the latent dimension increases, the accuracy increases as evidenced by lower CD values. However, the decline in accuracy is most pronounced in the initial latent dimensions, after which it stabilizes. Notably, there is a modest increase in accuracy when transitioning from a latent dimension of 10–25, despite the latter having 50% more parameters. Conversely, the VAE with a latent dimension of 5 with a slight difference in parameter count results in a considerable drop in performance. This suggests that a latent dimension of 10 adequately captures the nuances of both open and closed curves in 4-, 6-, and 8-bar mechanisms. VAEs with latent dimensions of 1 and 2 produce unsatisfactory results.

Li et al. [42] demonstrated that the first five harmonics of Fourier decomposition effectively encapsulate all pertinent information about 4-bar coupler curves. Given the increased complexity of higher-order linkages, expanding the number of Fourier descriptors becomes essential to comprehensively capture curve characteristics. Our findings imply a parallel utility in VAEs, where a 10-dimensional latent space suffices as a standalone representation for 4-, 6-, and 8-bar coupler curves.

Given the superior performance of the 10-dimensional VAE, we proceed to compute the remaining metrics exclusively using this model of interest. Initially, we investigate the impact of the number of sampled points in the vicinity of the latent representation of the input curve on the results. To do this, we input a random testing coupler curve and sample 100, 1000, and 5000 points in

Table 3 Reconstruction accuracy of different VAEs trained, as well as their training and inference times

Latent dimension	Chamfer distance	Number of parameters	Training time per epoch (s)	Inference time per batch of 1024 (ms)
1	0.574	618,851	141	205 ± 2.72
2	0.586	636,261	138	204 ± 3.42
5	0.183	688,491	140	205 ± 4.49
10	0.135	775,541	139	206 ± 2.69
25	0.139	1,036,691	141	205 ± 3.19
50	0.142	1,471,941	141	215 ± 5.11

[3]https://github.com/purwarlab/vae_dataset_project

the latent space. We then observe how the variety of the predicted mechanisms changes with an increasing number of sampled points.

It is crucial to note that as the number of sampled points in the latent space increases, the points tend to deviate further from the latent representation of the target coupler curve. Consequently, while a greater number of mechanisms are generated, their resulting coupler curves may diverge from the intended design. To ensure the exclusion of poor predictions, we filter out results that exceed a threshold value of 0.2 for the accuracy metric (CD). Experimental analyses indicated that beyond this threshold, the results visually deviate from the intended coupler curve. The calculation of the CD is performed using normalized versions of the curves, ensuring that this metric is invariant to scale, rotation, and other transformations. The normalized coupler curves in our study have a scale ranging from −3.5 to 3.5 units on both axes.

For sampled points of 100, 1000, and 5000, we obtained 60, 155, and 172 types of mechanisms capable of approximating the input coupler curve, respectively, and a total of 80, 569, and 2346 mechanisms. The trend is clear: increasing the number of samples results in a greater variety of mechanisms. Specifically, sampling 5000 points provides 172 types of mechanisms (out of the 178 total types in our dataset) for the given coupler curve. However, the more samples you take, the further you get from the latent representation, leading to a decrease in the ratio of the total number of good approximations to the total number of predictions.

4.2 Novelty. After establishing the initial alignment metric between the generated coupler curve and the desired one, our focus shifts to evaluating the novelty of the generated mechanisms. Novelty in this context refers to how distinct mechanisms are from each other within specific types.

Unlike determining accuracy, defining and calculating novelty poses challenges due to the absence of a universally agreed-upon criterion for what constitutes differences between mechanisms. To assess novelty, we generate a set number of random mechanisms based on a target coupler curve, filter them out based on accuracy, and categorize them into groups by their type. The novelty metric is calculated as the average L_2 norm of the differences between the joint coordinates of mechanisms within each type. This metric enables a straightforward comparison of dissimilarities: larger values indicate greater divergence, suggesting the creation of genuinely novel mechanisms rather than variations of similar ones. Conversely, smaller values signal redundancy, prompting the exclusion of duplicated mechanisms.

Throughout this analysis, we normalize the mechanisms to mitigate biases stemming from size discrepancies, ensuring a fair comparison across all generated examples. In calculating novelty scores, we employ a threshold of 10 to gauge the resulting L_2 norms. A higher threshold indicates a more stringent criterion for differentiating between two mechanisms based on their dissimilarities.

We evaluate novelty by calculating the average results for a set of 100 test coupler curves, which were previously utilized for accuracy assessment. For each coupler curve, we perform 500 sampling iterations to generate new mechanisms. This comprehensive sampling process results in a total of 18,653 novel mechanisms, encompassing all 178 different mechanism types. By analyzing these extensive samples, we can thoroughly assess the novelty of the generated mechanisms and ensure that the variety and uniqueness of the outputs are adequately captured.

4.3 Diversity. Diversity is measured by calculating the proportion of novel mechanisms generated for each type relative to the total. This metric helps determine whether the model excels in producing certain types of mechanisms while potentially overlooking others. A balanced distribution of novel mechanisms across all types indicates diversity within the generated solutions.

Having calculated novelty, we now proceed to assess the diversity of the generated mechanism types. Figure 8 depicts the frequency of each mechanism type in the predicted solutions. The three most frequently appearing mechanisms are RRRP, RRPR, and Stephenson III, with occurrences of 251, 256, and 295 times, respectively. In contrast, the least frequent mechanisms are predominantly 8-bar types, appearing only a few times each for the 100 curves tested. Despite the significant disparity, it is noteworthy that all trained mechanism types were represented. Excluding these extreme cases, the distribution is relatively balanced, indicating a broad representation of all types.

It is also important to note that we subdivide mechanisms related to the same connectivity type depending on the selected activation link. Disregarding this subdivision and combining these cases, we find that each mechanism type was present more than 25 times in our results, suggesting significant diversity in our generated mechanisms.

This holistic evaluation, encompassing approximation accuracy, novelty, and diversity, offers a thorough assessment of the model's capabilities and limitations in generating a diverse range of innovative mechanisms.

4.4 Robustness. Robustness is the model's capacity to maintain high performance across diverse conditions, including scenarios with noisy or imperfect data. This metric is crucial because real-world datasets often exhibit variability and inconsistencies, demanding a model that can reliably handle such challenges in practical applications. To assess the robustness of our model, we subject it to perturbed input data and evaluate how consistently it generates accurate mechanisms under these altered conditions. This involves introducing noise or randomly removing segments from the input coupler curves and analyzing the fidelity of the reconstructed curves compared to the original target curves.

Beyond perturbing the coupler curves, evaluating the model's robustness also includes testing its performance on previously unseen data. While it is anticipated that the model may struggle to produce meaningful results under these circumstances, examining its behavior when faced with unfamiliar inputs provides valuable insights. Observing instances where the model fails can offer critical feedback for further refinement and improvement.

To calculate the robustness of the model against noise in the input data, we employ the following methodology. This procedure mirrors the accuracy assessment described previously, with an additional step of introducing standard Gaussian noise to the input coupler curve points before processing them through the VAE's encoder. The noise has a mean of 0 and varying standard deviations. We observe the impact of increasing the standard deviation on the model's performance.

Figure 9 illustrates the effect of progressively increasing noise on image quality. As shown, a small amount of noise does not significantly degrade the image, but as the standard deviation exceeds 0.2, the images become unrecognizable.

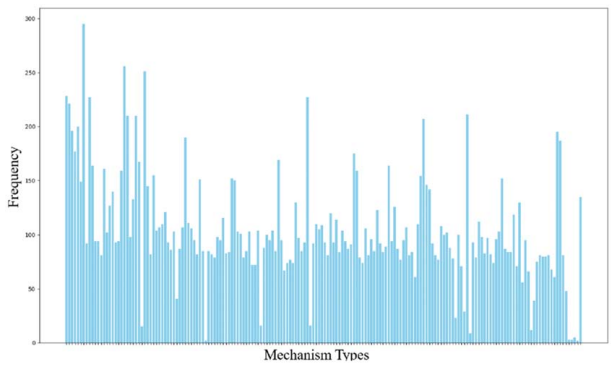

Fig. 8 Diversity analysis of sampling 500 times for 500 test curves. The *X*-axis is the mechanism type and the *Y*-axis is the number of mechanisms generated. Since our dataset has a lot of mechanism types, we omit the mechanism's names in the *X*-axis.

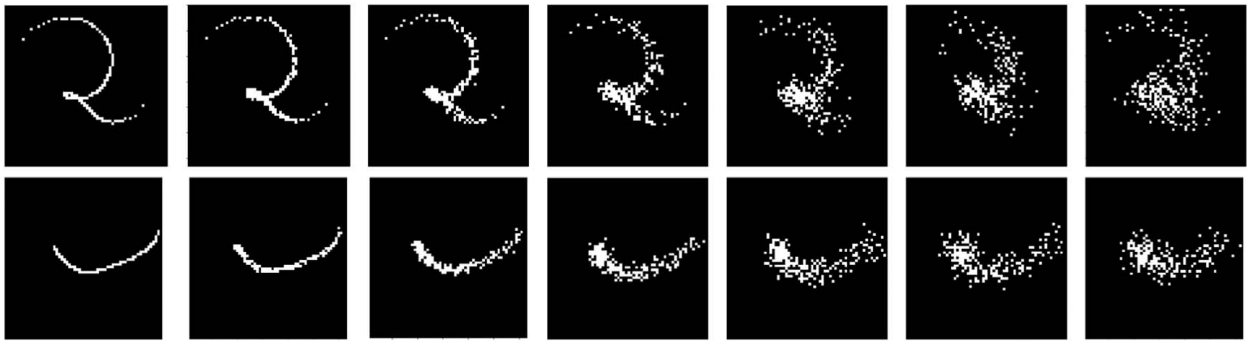

Fig. 9 Gradual degradation of image quality (from left to right) with increasing Gaussian noise standard deviation. At lower levels, images remain recognizable, while higher noise levels lead to significant distortion and loss of fidelity.

Table 4 Average Chamfer distance values for test images as noise standard deviation increases

Noise standard deviation	0.0	0.05	0.1	0.2	0.3	0.4	0.5
Chamfer distance	0.135	0.143	0.159	0.197	0.243	0.281	0.313

Table 4 presents the average CD values obtained from the same test images as the standard deviation of the introduced noise increases. The results indicate that the average CD value rises with increasing noise levels. However, for small noise levels (standard deviations of 0.05 and 0.1), the performance remains acceptable.

We also evaluated our model using previously unseen data. Figure 10 showcases the spiral coupler curve depicted as blue dots, juxtaposed with the generated mechanisms. While the model does not precisely replicate the curve, the generation of similar structures demonstrates its capacity to adapt to novel data and strive for close approximations. This resilience suggests robustness against unseen data inputs, with efforts directed toward accurately approximating any given input. These findings underscore the model's ability to handle variations in input data, including minor noise levels.

These assessments, encompassing both perturbed data experiments and evaluations of unseen data, provide a comprehensive understanding of the model's robustness in practical applications. They highlight the model's ability to maintain stability and reliability amidst data imperfections, essential for its effective deployment in real-world scenarios.

4.5 Computational Efficiency. Computational efficiency is the time and computational resources required to generate mechanisms. This metric is crucial because efficient models can save significant computational costs and time, making them more practical for real-world applications. To evaluate computational efficiency, we measure the runtime for generating mechanisms and compare these metrics with baseline methods. By assessing the computational overhead, we can determine the practicality and scalability of our model in various scenarios.

Despite the differences in the size of our VAE models, the time difference in training and inference between them is very small as shown in Table 3. Training all our VAEs on a dataset of 2,971,507 images, with an 80%, 10%, and 10% split for training, validation, and testing sets, using batches of 1024 on 2 × NVIDIA RTX A6000 GPUs, results in approximately 140 s per epoch. Although our largest model, with 50 latent dimensions, is significantly larger than our smallest model, with only 1 dimension, the relative total number of parameters for all of them is very small. Consequently, there are practically no differences in the training times between any of them. Similarly, the inference times for the models remain consistent across different latent dimensions, averaging 205 ms to process a batch of 1024 coupler curves and provide the results. This is a significant improvement compared to the 45 min reported by Ref. [26]. This efficiency ensures that our models perform reliably without significant time penalties, regardless of their complexity. The dataset generation time, as already detailed in Sec. 2, further complements this efficiency.

4.6 Scalability. Scalability is the capability of the model to handle increasing amounts of data or more complex mechanisms.

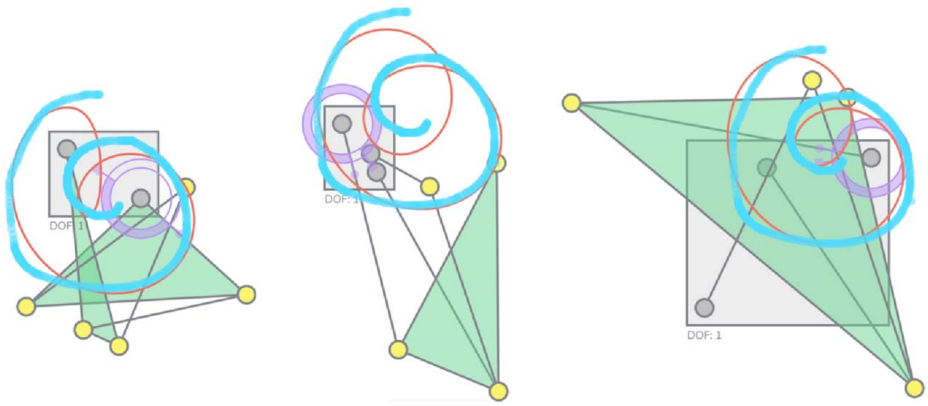

Fig. 10 Performance of our pipeline with an input (thicker line) significantly different from the training images in the database, illustrated with predictions (less thick line) for a spiral coupler curve. Although the model does not accurately approximate the curve, the similar predicted curves suggest robustness to previously unseen data and an effort to approximate as closely as possible.

This metric is crucial for understanding the model's potential to be applied to larger-scale problems and more intricate designs. To measure scalability, we evaluate the model's performance with larger datasets and more complex mechanism configurations. By testing the model under these expanded conditions, we can determine its ability to maintain efficiency and accuracy as the complexity and volume of input data increase.

The scalability of this method can be compared to the work by Deshpande and Purwar [23], who trained a 50dimensional VAE on a dataset of 6818 linkages, including 4-bar, Stephenson I, and Stephenson III 6-bar linkages. In our work, we extended their approach to include all 4-bar, 6-bar, and 8-bar mechanisms with all revolute joints, as well as 4-bar mechanisms with prismatic joints, resulting in a dataset of 2,971,507 linkages. This demonstrates that the approach is easily scalable to higher-order mechanisms. The only requirement for further scaling is the addition of newly generated mechanisms to the database.

5 Examples

To conclude our work, we apply our method to coupler curves of two particular shapes—straight-line and infinity-shaped curves. A perfectly straight line cannot be generated by a 4-bar linkage, but

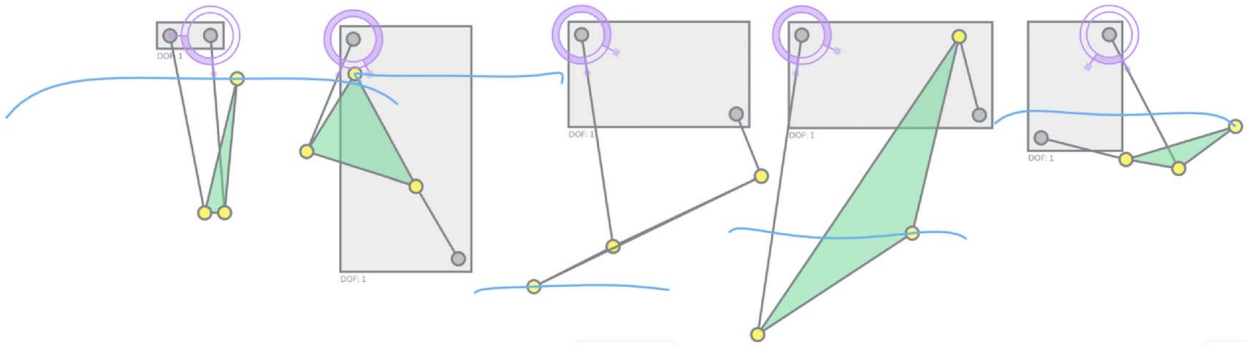

Fig. 11 Generated 4-bar mechanisms for a straight-line coupler curve. Several predicted 4-bar mechanisms exhibit striking similarities to well-known designs such as the Chebyshev lambda, Watt's, and Robert's linkages.

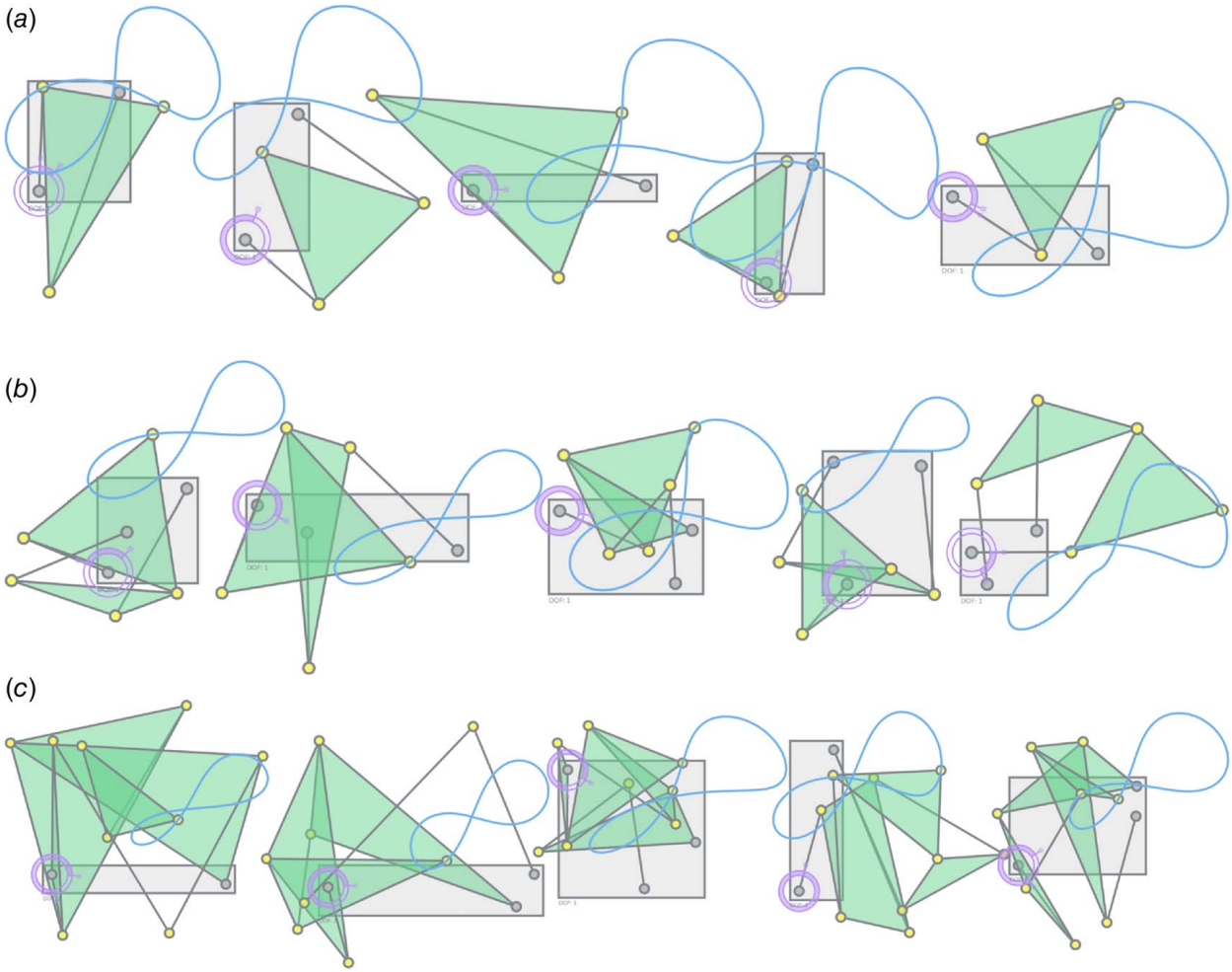

Fig. 12 The 4-, 6-, and 8-bar mechanisms predicted for an infinity-shaped input coupler curve. (*a*) Predicted 4-bar mechanisms, (*b*) predicted 6-bar mechanisms, and (c) predicted 8-bar mechanisms.

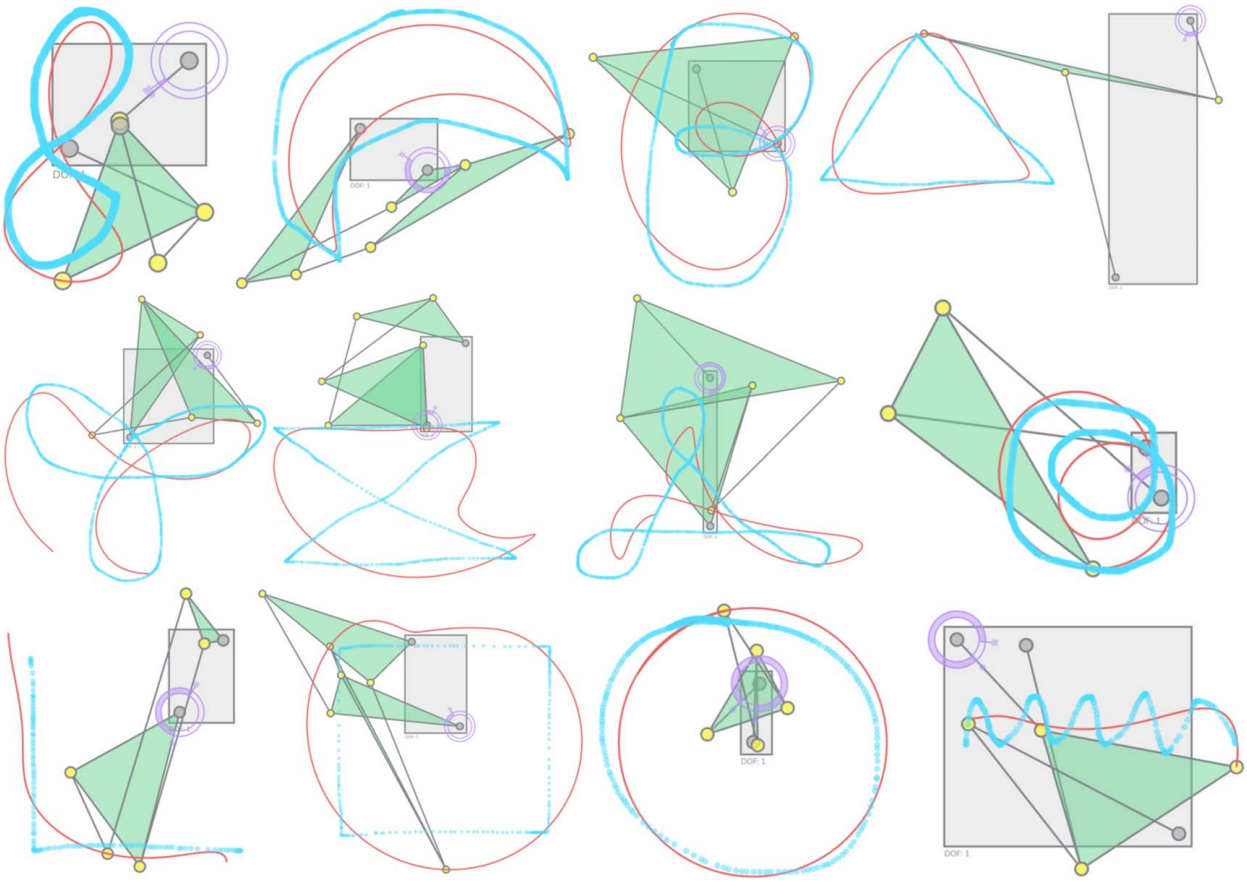

Fig. 13 Twelve input coupler curves drawn roughly using the mouse in a thicker line alongside their corresponding top predicted mechanisms, with the generated coupler curves drawn in a less thick line. The predictions were generated using MotionGen [44]. The results demonstrate varying degrees of accuracy, with some predictions closely approximating the input curves, while others show less precise alignment.

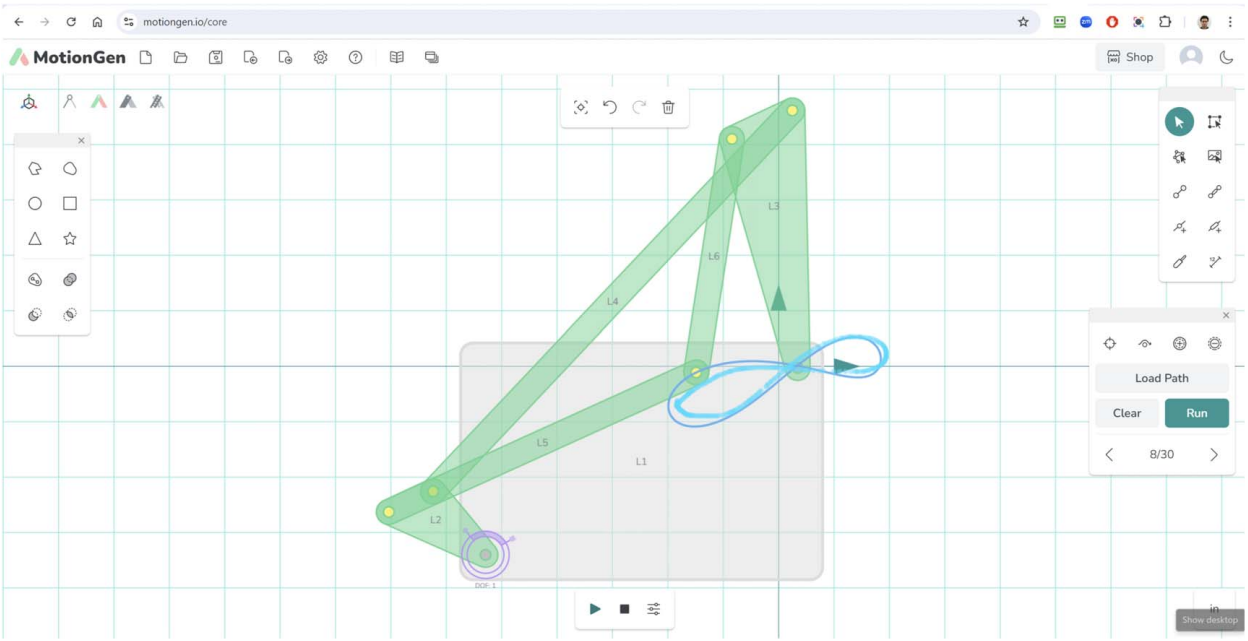

Fig. 14 Steps for testing path synthesis function in MotionGen [44]: (1) click on the *Panels*, which is the right-most icon in the top menu, (2) select the "Path Synthesis" option, (3) click the target-shaped button to draw a curve, (4) click on the *Run* button to execute the program and obtain the predicted mechanisms, and (5) explore the predicted 30 mechanisms using left and right arrow in the panel

there exist mechanisms capable of approximating the straight-line curve, such as Watt's linkage, Robert's linkage, and Chebyshev's linkage. Higher-order linkages like the Peaucellier–Lipkin inversor and Hart's first inversor can trace a straight line perfectly [43].

Figure 11 displays several of our predicted mechanisms for a straight-line path input, showcasing a wide variety of mechanisms. It is noteworthy that all results approximate the straight-line path to some extent. Additionally, some of the predicted mechanisms, if optimized, resemble well-known linkages such as the Chebyshev lambda, Watt's, and Robert's linkages. Figure 12 presents the results for an infinity-shaped coupler curve, demonstrating the model's ability to predict a variety of mechanisms for different coupler curves. Due to space limitations, we show only five linkages per mechanism type, though there are many more predicted mechanisms approximating the infinity-shaped curve. The diversity in the predicted mechanisms highlights the model's versatility and effectiveness.

Figure 13 presents 12 input coupler curves alongside their corresponding predicted mechanisms. This figure, in contrast to earlier ones, highlights only the top prediction for each input curve to illustrate the pipeline's full capabilities. The accuracy of the approximations varies, with some predictions aligning closely with the input curves, while others deviate more significantly. Notably, certain input curves were not included in the original dataset, which contributed to less accurate predictions. The coupler curves in the top two rows are taken from Pan et al. [15], and the results are comparable to those reported by Fogelson et al. [26].

For further exploration, other input coupler curves can be tested using our web-based application MotionGen [44], see Fig. 14. At the moment, MotionGen only generates mechanisms of the following types—RRRR, RRRP, Watt I, Watt II, Stephenson I, Stephenson II, and Stephenson III—as it was trained on 62,946 mechanisms of the mentioned types. It also provides only 30 predicted mechanisms for a given coupler curve to explore through. However, the proposed dataset is being integrated into MotionGen and will be available in the near future along with a filtering option.

Overall, our findings highlight the scalability of the VAE-based method for capturing and generating diverse mechanisms that can approximate various input coupler curves, including challenging cases like straight-line curves.

The efficiency and consistency in training and inference times across different latent dimensions underscore the robustness and practical applicability of our approach. This adaptability, combined with the ability to extend to higher-order mechanisms, demonstrates the method's potential for broader applications in mechanical design and synthesis.

6 Conclusions and Future Work

In this work, we successfully generated a dataset comprising 4-, 6-, and 8-bar mechanisms with revolute joints, as well as 4-bar linkages with prismatic joints. We have then showcased its usage by training VAEs with varying latent dimensions to find the lowest number of dimensions needed to handle the great variety of curves present in the database. We showed that a VAE with 10 latent dimensions would suffice. We then proposed different metrics to quantify the results and showed a sample calculation. In the end, we presented a few examples of curves generated by a variety of mechanisms.

For future work, we plan to expand our dataset to include higher-order mechanisms, such as spherical and spatial mechanisms. Additionally, we will investigate using conditional VAEs, where mechanism parameters can serve as inputs with coupler curves as conditions, allowing for a more targeted and accurate generation of linkages. Another promising avenue of exploration is the extension of our approach to motion generation problems, thereby broadening the applicability of our techniques to more complex mechanical systems and problems.

Acknowledgment

This publication has research support from a National Science Foundation STTR phase II award (# 2126882) to CoPI Purwar who also holds stocks in Mechanismic Inc. The research findings included 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 in accordance with its policy on objectivity in research.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available.[4]

References

[1] Chettibi, T., 2006, "Synthesis of Dynamic Motions for Robotic Manipulators With Geometric Path Constraints," Mechatronics, **16**(9), pp. 547–563.
[2] Lyu, Z., and Purwar, A., 2024, "Deep Learning Conceptual Design of Sit-to-Stand Parallel Motion Six-Bar Mechanisms," ASME J. Mech. Des., **147**(1), p. 013306.
[3] Kapsalyamov, A., Hussain, S., Brown, N. A. T., Goecke, R., Hayat, M., and Jamwal, P. K., 2023, "Synthesis of a Six-Bar Mechanism for Generating Knee and Ankle Motion Trajectories Using Deep Generative Neural Network," Eng. Appl. Artif. Intell., **117**(1), p. 105500.
[4] Xu, X., Zhou, M., Chen, X., and Yang, J., 2022, "Processing Method of Gearbox With NonCircular Gear Train and Its Application in Rice Potted Seedling Transplanting Mechanism," Agriculture, **10**(10), p. 1676.
[5] Norton, R., 2011, *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*, 5th ed., McGraw Hill, Boston, MA.
[6] Bernhard, T., Stelian, C., Damien, G., Vittorio, M., Eitan, G., and Markus, G., 2014, "Computational Design of Linkage-Based Characters," ACM Trans. Graph., **33**(4), pp. 1–9.
[7] Kempe, A. B., "On a General Method of Describing Plane Curves of the nth Degree by Linkwork," Proc. Lond. Math. Soc., VII, **s1–7**(1), pp. 213–216.
[8] Nolle, H., 1974, "Linkage Coupler Curve Synthesis: A Historical Review – I. Developments up to 1875," Mech. Mach. Theory, **9**(2), pp. 147–168.
[9] Nolle, H., 1974, "Linkage Coupler Curve Synthesis: A Historical Review – II. Developments After 1875," Mech. Mach. Theory, **9**(3–4), pp. 325–348.
[10] Koetsier, T., 1983, "A Contribution to the History of Kinematics – I," Mech. Mach. Theory, **18**(1), pp. 37–42.
[11] Koetsier, T., 1983, "A Contribution to the History of Kinematics – II," Mech. Mach. Theory, **18**(1), pp. 43–48.
[12] Liu, Y., and McCarthy, J. M., 2017, "Synthesis of a Linkage to Draw a Plane Algebraic Curve," Mech. Mach. Theory, **111**(1), pp. 10–20.
[13] Liu, Y., and McCarthy, J. M., 2017, "Design of Mechanisms to Draw Trigonometric Plane Curves," ASME J. Mech. Rob., **9**(2), p. 024503.
[14] Wampler, C. W., Morgan, A. P., and Sommese, A. J., 1992, "Complete Solution of the Nine-Point Path Synthesis Problem for Four-Bar Linkages," ASME J. Mech. Des., **114**(1), pp. 153–159.
[15] Pan, Z., Liu, M., Gao, X., and Manocha, D., 2022, "Joint Search of Optimal Topology and Trajectory for Planar Linkages," Int. J. Rob. Res., **42**(4–5), pp. 176–195.
[16] Lee, W.-T., and Russell, K., 2018, "Developments in Quantitative Dimensional Synthesis (1970–Present): Four-Bar Path and Function Generation," Inverse Prob. Sci. Eng., **26**(9), pp. 1280–1304.
[17] Hoskins, J., and Kramer, G., 1993, "Synthesis of Mechanical Linkages Using Artificial Neural Networks and Optimization," IEEE International Conference on Neural Networks, San Francisco, CA, Mar. 28–Apr. 1.
[18] Zahn, C. T., and Roskies, R. Z., 1972, "Fourier Descriptors for Plane Closed Curves," IEEE Trans. Comput., **C-21**(3), pp. 269–281.
[19] Chuang, G. C.-H., and Kuo, C.-C. J., 1996, "Wavelet Descriptor of Planar Curves: Theory and Applications," IEEE Trans. Image Process., **5**(1), pp. 56–70.
[20] Vasiliu, A., and Yannou, B., 2001, "Dimensional Synthesis of Planar Mechanisms Using Neural Networks: Application to Path Generator Linkages," Mech. Mach. Theory, **36**(2), p. 299–310.
[21] Galan-Marin, G., Alonso, F. J., and Del Castillo, J. M., 2009, "Shape Optimization for Path Synthesis of Crankrocker Mechanisms Using a Wavelet-Based Neural Network," Mech. Mach. Theory, **44**(6), p. 1132–1143.
[22] 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.

[4]See Note 2.

[23] Deshpande, S., and Purwar, A., 2020, "An Image-Based Approach to Variational Path Synthesis of Linkages," ASME J. Comput. Inf. Sci. Eng., **21**(2), p. 021005.
[24] Sharma, S., and Purwar, A., 2022, "A Machine Learning Approach to Solve the Alt-Burmester Problem for Synthesis of Defect-Free Spatial Mechanisms," ASME J. Comput. Inf. Sci. Eng., **22**(2), p. 021012.
[25] Nurizada, A., and Purwar, A., 2023, "An Invariant Representation of Coupler Curves Using a Variational AutoEncoder: Application to Path Synthesis of Four-Bar Mechanisms," ASME J. Comput. Inf. Sci. Eng., **24**(1), p. 011003.
[26] Fogelson, M. B., Tucker, C., and Cagan, J., 2023, "GCP-HOLO: Generating High-Order Linkage Graphs for Path Synthesis," ASME J. Mech. Des., **145**(7), p. 071701.
[27] Regenwetter, L., Nobari, A. H., and Ahmed, F., 2022, "Deep Generative Models in Engineering Design: A Review," ASME J. Mech. Des., **144**(7), p. 071701.
[28] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, Vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., Red Hook, NY.
[29] Kingma, D. P., and Welling, M., 2014, "Auto-Encoding Variational Bayes," Computing Research Repository. abs/1312.6114
[30] Purwar, A., and Chakraborty, N., 2023, "Deep Learning-Driven Design of Robot Mechanisms," ASME J. Comput. Inf. Sci. Eng., **23**(6), p. 060811.
[31] Picard, C., Schiffmann, J., and Ahmed, F., 2023, "Dated: Guidelines for Creating Synthetic Datasets for Engineering Design Applications," Volume 3A: 49th Design Automation Conference (DAC), Boston, MA, Aug. 20–23, p. V03AT03A048.
[32] Nobari, A. H., Srivastava, A., Gutfreund, D., and Ahmed, F., 2022, "LINKS: A Dataset of a Hundred Million Planar Linkage Mechanisms for Data-Driven Kinematic Design," Volume 3A: 48th Design Automation Conference (DAC), St. Louis, MO, Aug. 14–17, p. V03AT03A040.
[33] Hajdu, A., Hajdu, L., and Tijdeman, R., 2012, "Approximations of the Euclidean Distance by Chamfer Distances," arXiv.
[34] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., et al.et al., 2021, "Learning Transferable Visual Models From Natural Language Supervision," arXiv.
[35] 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.
[36] Sener, S., and Unel, M., 2006, "Geometric Invariant Curve and Surface Normalization," International Conference on Image Analysis and Recognition, Póvoa de Varzim, Portugal, Sept. 18–20, pp. 539–548.
[37] Wilkinson, M., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., et al., 2016, "The FAIR Guiding Principles for Scientific Data Management and Stewardship," Sci. Data, **3**(1), p. 160018.
[38] Kullback, S., and Leibler, R. A., 1951, "On Information and Sufficiency," Ann. Math. Stat., **22**(1), pp. 79–86.
[39] Sun, J., Wang, P., Liu, W., and Chu, J., 2018, "Noninteger-Period Motion Generation of a Planar Four-Bar Mechanism Using Wavelet Series," Mech. Mach. Theory, **121**(3), pp. 28–41.
[40] Kouiroukidis, N., and Evangelidis, G., 2011, "The Effects of Dimensionality Curse in High Dimensional kNN Search," 2011 15th Panhellenic Conference on Informatics, Kastoria, Greece, Sept. 30–Oct. 2, pp. 41–45.
[41] Agarap, A. F., 2018, "Deep Learning Using Rectified Linear Units (ReLU)," CoRR. abs/1803.08375.
[42] 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.
[43] Kempe, A. B., 1877, *How to Draw a Straight Line: A Lecture on Linkages*, Macmillan and Company, London. https://books.google.az/books?id=QsF6mxz0Fl4C&redir esc=y
[44] Mechanismic Inc., 2022, "MotionGen," http://www.motiongen.io.