A Machine Learning Approach to Solve the Alt–Burmester Problem for Synthesis of Defect-Free Spatial Mechanisms

Shashank Sharma
Computer-Aided Design and Innovation Lab,
Department of Mechanical Engineering,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: shashank.sharma.1991@outlook.com

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

In this paper, we present a machine learning algorithm to synthesize defect-free single-degree-of-freedom spatial mechanisms for the Alt–Burmester problem. The Alt–Burmester problem is a generalization of a pure motion synthesis problem to include via path-points with missing orientations. While much work has been done towards the synthesis of planar and, to some extent, spherical mechanisms, the generation of mechanisms that are free of circuit, branch, and order defects has proven to be a difficult task. This is even more challenging for spatial mechanisms, which can consist of a large number of circuits and branches. Moreover, the Alt–Burmester problem makes solving such problems using an analytical approach further demanding. In this paper, we present a novel machine learning algorithm for solving the Alt–Burmester problem for spatial 5-SS platform mechanism using a variational autoencoder (VAE) architecture. The VAE helps capture the relationship between path and orientation properties of the motion of the 5-SS mechanisms, which enables reformulating the Alt–Burmester problem into a pure motion synthesis problem. The end goal is to produce defect-free spatial mechanism design solutions. While our focus in this paper is on the 5-SS mechanisms, this approach can be scaled to any single-degree-of-freedom spatial mechanisms. [DOI: 10.1115/1.4051913]

Keywords: artificial intelligence, mechanisms and robotics, computational synthesis, computer-aided design, machine learning for engineering applications

1 Introduction

This paper is concerned with the Alt–Burmester problem, which provides a set of pose and path-point (or position-) input data for the floating link of a spatial mechanism. Here, a pose is defined as a combination of position and orientation of the floating link and a path-point is defined by position only. Traditionally, kinematic synthesis of a mechanism has been classified and studied as the path-, motion-, and function generation problems [1,2]. Path synthesis problems consider only path-point coordinates (x_i, y_i, z_i) as inputs while motion synthesis problems specify poses $(x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$, where (x_i, y_i, z_i) are the coordinates of the origin of the moving frame attached to the floating link and the angles $(\alpha_i, \beta_i, \gamma_i)$ represent the orientation of the moving frame with respect to a fixed frame. Thus, an Alt–Burmester problem can be considered to be a special motion synthesis problem, where the orientation information for a few poses is missing. The goal is to find the dimensions of mechanisms for a given input of this type.

There exists a vast amount of literature on the path generation problem for planar, spherical, and spatial mechanisms. Algebraic methods for synthesis use techniques like complex number analysis and displacement matrix methods to find analytical closed-form solutions [1,3–8]. Optimization techniques attempt to minimize an objective function and find mechanisms whose coupler traces the best approximating motion curve [9–13]. Atlas approaches explore the use of curve invariants like Fourier descriptors to create and search a database of coupler curves [14,15]. Neural network approaches have also been proposed for planar path synthesis [16,17]. Recently, Deshpande and Purwar have proposed machine learning models using autoencoders to synthesize planar mechanisms [18–20]. Chiang presented an exhaustive review of the kinematics of spherical mechanisms [21]. Path generation of spherical mechanisms has been carried out using numerical atlas methods [22,23]. Premkumar et al. have proposed an optimization solution for the synthesis of the RRSC and RRSS spatial mechanisms [24,25]. Ananthasuresh and Kramer proposed the generalized reduced gradient method of optimization for the synthesis of the RSCR spatial mechanisms [26]. Jiménez et al. outlined a generalized constraint optimization technique [27]. Sun et al. created a database using the Fourier series to compare curves and synthesized RCCC spatial mechanisms [28]. Our previous work used machine learning (ML) and database search for path synthesis of spatial 5-SS mechanisms [29].

Similarly, the motion synthesis of mechanisms is also a well-researched topic. Kinematic mapping and algebraic fitting algorithms have been used for motion synthesis of planar mechanisms [30–34]. Generalizations of these approaches using analytical, homotopy, or optimization algorithms have been proposed for both spherical mechanisms [35–40] and spatial mechanisms [41–45].

However, the theoretical distinction between path and motion synthesis often presented in the literature does not play well with real-world problems. Many practical problems require a designer to satisfy a combination of both path constraints and pose constraints. This happens when designers have information about a few key poses but lack any knowledge of the orientation of some path-points. In the absence of precise constraints on poses, this is a more natural specification of the problem. Tong et al. [46] solved the Alt–Burmester problems, which they named after Alt's [3] and Burmester's [30] work on planar four-bar mechanisms synthesis. Brake et al. [47] proposed a homotopy approach to synthesize planar four-bar mechanisms for this problem. Zimmerman [48] presented a geometric approach to find four-bar solutions for the mixed synthesis problem. It is well-known that the orientations of the coupler of a planar four-bar mechanism are

[1]Corresponding author.

Contributed by the Computers and Information Division of ASME for publication in the Journal of Computing and Information Science in Engineering. Manuscript received October 5, 2020; final manuscript received July 22, 2021; published online October 13, 2021. Assoc. Editor: Matthew I. Campbell.

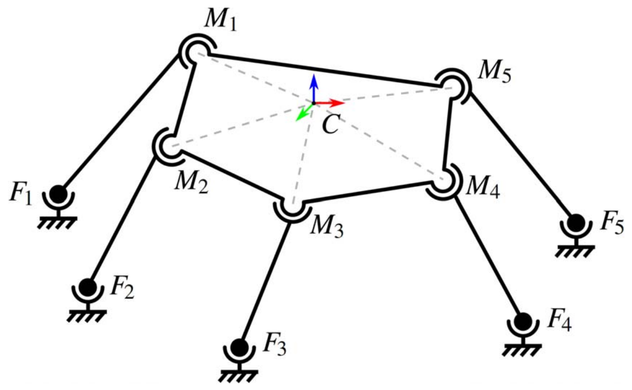


Fig. 1 Kinematic diagram for a 5-SS mechanism

inextricably tied to its path, i.e., one cannot specify arbitrary orientations for path-points and expect to find a suitable four-bar mechanism. This is true of all single-degree-of-freedom mechanisms. In our previous work [49], we have found this closed-form relationship between path and orientation data using Fourier descriptors for planar four-bar mechanisms and proposed a motion synthesis approach to solve the Alt–Burmester problems. However, this approach requires that we manually discover the connection between the properties of the path and the motion for each planar mechanism independently. Also, since Fourier analysis is used to break down 2D coupler curves as a sum of trigonometric functions using a complex Fourier transform, this approach cannot scale to spatial 3D mechanisms. In this paper, we propose a machine learning approach to solve the Alt–Burmester problems for the purpose of generating defect-free spatial mechanisms. Although the paper focuses on the synthesis of 5-SS linkage mechanisms, the approach is general enough to handle any single-degree-of-freedom spatial mechanism. A 5-SS platform linkage consists of a rigid movable platform (also called, coupler) supported by five legs, each with a spherical joint on both ends (see Fig. 1). The motion of a point on the moving platform of an example 5-SS mechanism is shown in Fig. 2. Incidentally, the figure shows universal (T) joints at the fixed base because a 5-TS mechanism was easier to simulate in a CAD package like Autodesk Inventor since it does not have redundant degrees-of-freedom present in a binary SS link. The kinematics of a 5-SS mechanism is exactly the same as a 5-TS mechanism. The coordinates of fixed pivots (F_i) and moving pivots (M_i) in the figure are taken from Innocenti's paper on 5-SS mechanism synthesis and represent the five smallest SS dyads in the paper [41].

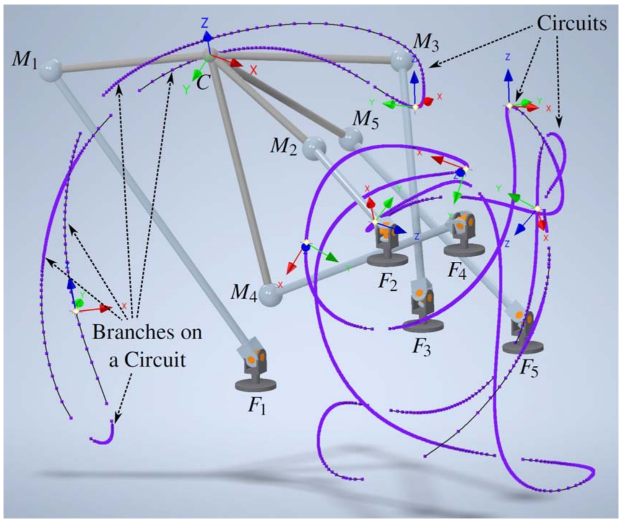


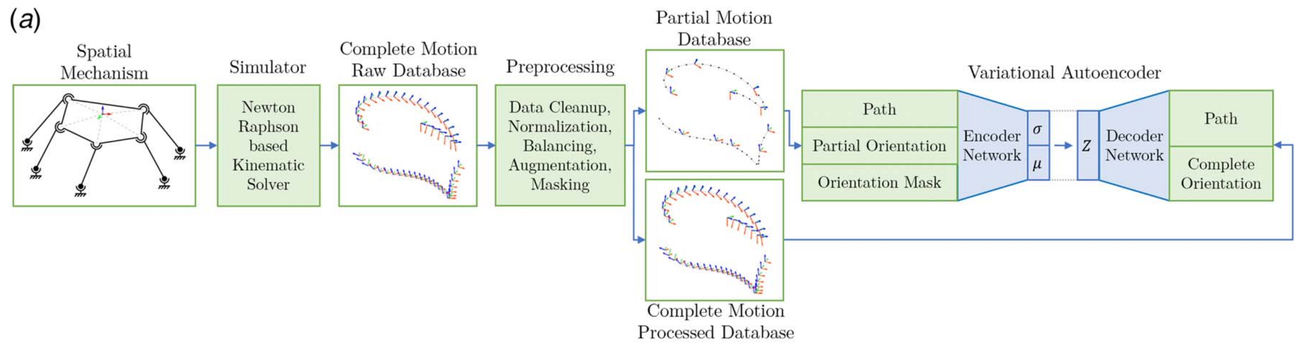
Fig. 2 Motion of coupler point of an example 5-SS mechanism

As can be seen in Fig. 2, coupler curves of spatial 5-SS mechanisms tend to have multiple branches and circuits. Here, we see two circuits characterized by two closed-loop curves and several branches, which break each curve into multiple segments separated by singularity points. A mechanism can carry a floating link continuously only in a given circuit and even in each circuit; limits of the actuation can break the circuit into pieces called branches. Thus, if several poses are given as input and they belong to different circuits, then the mechanism can carry the floating link through the poses within one circuit only. The mechanism would have to be disassembled and assembled again in a different configuration to make it pass through poses on another circuit. This is known as a circuit defect. Chase and Mirth discuss in great detail the challenges of synthesizing single-degree-of-freedom mechanisms due to circuit and branch defects [50]. Roth and Freudenstein have discussed the occurrence of defects in mechanism synthesis for path generation using numerical methods [51]. Wampler et al. show that there exist many defective mechanisms for the nine-point path synthesis problem [52]. These defects tend to be more prominent and more difficult to resolve in spatial mechanisms compared to planar mechanisms. For example, a planar four-bar can have up to two circuits and four branches. However, in Fig. 2, we can observe the existence of at least three circuits and 15 branches.

The field of computer vision has spearheaded the use of machine learning technologies to solve challenging real-world problems [53,54]. One of these problems has been Image Inpainting which aims to restore damaged paintings and photographs [55]. The use of generative models like variational autoencoders (VAEs) [56] and generative adversarial networks [57] has proven to be extremely successful in adding missing information to corrupted images [58,59]. The VAE's ability to do data augmentation, variation synthesis, and dimensionality reduction could be advantageous in the kinematic synthesis of mechanisms as well.

In the context of the Alt–Burmester problem, the VAE takes a mixture of path-points and poses as input, and outputs multiple pure motions, i.e., it augments the "missing" orientations to the path-points. To add *compatible* orientations to the Alt–Burmester problem, the VAE has to learn path-orientation relationships for defect-free motions. Since orientation augmentation can result in multiple motions, we are interested in generating a variation of plausible defect-free motions. The VAE does this by also learning the underlying distribution of the family of possible trajectories. This underlying multivariate Gaussian distribution is used to sample a latent vector that represents a low dimensional signature of motions. Thus, we use a VAE to reformulate the Alt–Burmester problem into a pure motion synthesis problem (also known as rigid body guidance problem) which enables us to leverage existing motion synthesis algorithms. In this paper, we use an atlas-based approach and search a clustered database to find solution mechanisms because during the training of the VAE, we already built a database. Although this paper focuses on 5-SS spatial mechanisms, the basic approach presented in this paper is agnostic of the type of mechanisms. Moreover, since the output of VAE is a conditioned motion with compatible orientation information, the last step of synthesis of mechanisms can also utilize other algebraic methods instead of a database-driven approach. We note that in the absence of compatible orientations for the given path-points, most algebraic methods, such as the ones proposed in Refs. [20,33,34], would produce defective planar mechanisms.

To train the ML model, we first create a database consisting of possible 5-SS linkages' coupler motion. These motions are calculated using an iterative Newton–Raphson algorithm solver. We use a quaternion representation for orientations and then data are normalized, pruned, and augmented using curvature and torsion of path-curves. After that, we use a VAE to learn the underlying distribution of coupler motions and their path-orientation relationship. Once the training phase is completed, the VAE is utilized to reformulate the Alt–Burmester problem into a motion synthesis problem by augmenting *compatible* orientation data. The basic idea is that once the VAE has learned a relationship between the orientations



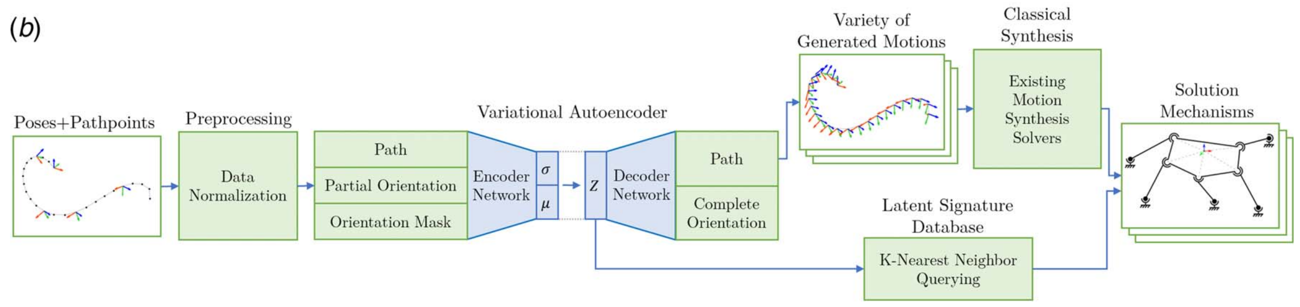


Fig. 3 Schematic of the algorithm: (*a*) training pipeline and (*b*) inference pipeline

and path-points of a mechanism, during the testing phase, it can be queried to provide missing orientations for the Alt–Burmester problems. An advantage of this approach is that since the training is performed only on defect-free motions, the likely output is a highly conditioned one, which would result in a defect-free mechanism. By sampling in the latent space of the VAE, a variety of plausible motion trajectory signatures, that fall in the family of defect-free 5-SS coupler motions, can be generated. These latent signatures are finally looked up in a hierarchical database, created using the K-means clustering algorithm, to find solution mechanisms. Alternatively, classical motion synthesis algorithms can also be used at this stage to generate more accurate mechanisms [41,42,44]. The outlined approach can be scaled to any single-degree-of-freedom spatial mechanisms. The algorithm proposed in this paper is visualized in Fig. 3.

The rest of the paper is organized as follows. Section 2 presents the numerical approach to generate 5-SS coupler motions; Sec. 3 discusses the methodology devised to make data more conducive to learning, while Sec. 4 uses semi-supervised machine learning tools to calculate multiple solution mechanisms. In Sec. 5, we present our approach to the mechanism synthesis for the spatial Alt–Burmester problem. Finally, in Sec. 6, two examples are presented, which showcase the effectiveness of the proposed approach.

2 Data Generation

The first step is to generate a sufficiently large data set of 5-SS mechanisms with their coupler motion trajectory. This is achieved by creating a Newton–Raphson solver, which uses the general constraint equations proposed in our previous work [60].

A 5-SS spatial mechanism in motion is subjected to a set of constraints imposed by the rigidity of its links. The general constraint enforces the rigidity of a binary link with two spherical joints represented by the homogeneous point coordinates of the fixed point (a_1, a_2, a_3, a_4) and that of the floating point (c_1, c_2, c_3, c_4), where a_4 and c_4 are the homogenizing factors. The constraint equation is given as

$$C_{\mathrm{SS}}: 2a_1c_1 + 2a_2c_2 + 2a_3c_3 + a_0c_4 = a_4\left(\frac{c_1^2 + c_2^2 + c_3^2}{c_4}\right) \quad (1)$$

where a_0 is given as

$$a_0 = a_4r^2 - \frac{a_1^2 + a_2^2 + a_3^2}{a_4} \quad (2)$$

Here, r is the radius of the sphere formed by the SS link with the center given by (a_1, a_2, a_3, a_4). Equation (1) represents a spherical constraint arising from the fact that the floating point is constrained to move on a sphere of radius r with its center located at the fixed joint.

In Figs. 1 and 2, the fixed pivots are labeled F_i, while the moving pivots on the coupler are labeled M_i, and the moving frame attached to the platform is located at the point C. The six-node moving platform $M_1M_2M_3M_4M_5C$ is kinematically equivalent to a set of 12 binary links M_1M_2, M_1M_3, M_1M_4, M_1M_5, M_2M_3, M_3M_4, M_4M_5, M_1C, M_2C, M_3C, M_4C, and M_5C, which represent 12 constraints. For a ternary link with three nodes, there would be three binary constraints and addition of a new node on the link would add three new constraints. Therefore, an L-node coupler has $l\times 3+3$ such constraints, where $l=L-3;\ l\geq 3$. An additional five constraints exist for the binary links $F_iM_i;\ i=1\ldots 5$. Thus, a spatial 5-SS mechanism is subjected to 17 independent rigidity constraints. During simulation, the Cartesian coordinates of the five fixed pivots $F_1, \ldots, F_5$ are the known parameters while the Cartesian coordinates of the five moving pivots $M_1, \ldots, M_5$ and the coupler point C are the unknowns. Since there exist 17 constraints and 18 unknowns, this results in a one degree-of-freedom coupler motion.

Since it is practically difficult to actuate a base spherical joint directly, we actuate the mechanism by placing a linear actuator between the fixed pivot of one dyad and the moving pivot of another dyad. Liao and McCarthy use the same actuation scheme [42]. The practicality of this scheme is demonstrated by Plecnik and McCarthy via a spatial steering linkage design [43]. The length of the actuator imposes an additional constraint on the motion and can be defined using Eq. (1) as a spherical constraint with a changing radius. To simulate the mechanism, the input actuator is iteratively perturbed by a finite displacement and the new position of the mechanism is calculated until the algorithm fails to converge. Newton–Raphson algorithm fails to converge at singular configurations and these configurations occur at the extreme points of each defect-free trajectory.

For a 5-SS mechanism, a system of 18 unknowns and 18 constraint equations can be formed and is represented as

$$\mathbf{\Phi}(\mathbf{q}) = 0 \tag{3}$$

where $\mathbf{q}$ is the state vector that consists of the unknown coordinates. Since the linear actuator is perturbed by a small finite displacement, the previous state of the mechanism serves as a good initial approximation.

The iterative simulation algorithm followed can be defined as

$$\mathbf{q}_{i+1} = \mathbf{q}_i - [\boldsymbol{J}^{-1}(\mathbf{q}_i)]\mathbf{\Phi}(\mathbf{q}_i) \tag{4}$$

where $\mathbf{q}_i$ is the state vector at ith iteration, $\mathbf{\Phi}(\mathbf{q}_i)$ is the vector of residuals at $\mathbf{q} = \mathbf{q}_i$, and $[\boldsymbol{J}^{-1}(\mathbf{q}_i)]$ is the inverse of Jacobian matrix evaluated at $\mathbf{q} = \mathbf{q}_i$. The Jacobian matrix is of the following form:

$$[\boldsymbol{J}(\mathbf{q})] = \begin{bmatrix} \frac{\partial\phi_1}{\partial q_1} & \frac{\partial\phi_1}{\partial q_2} & \cdots & \frac{\partial\phi_1}{\partial q_{18}} \\ \frac{\partial\phi_2}{\partial q_1} & \frac{\partial\phi_2}{\partial q_2} & \cdots & \frac{\partial\phi_2}{\partial q_{18}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial\phi_{18}}{\partial q_1} & \frac{\partial\phi_{18}}{\partial q_2} & \cdots & \frac{\partial\phi_{18}}{\partial q_{18}} \end{bmatrix} \tag{5}$$

To calculate the Jacobian matrix, relations describing the first-order partial derivatives of constraint equations are required. For an SS dyad described in Eq. (1), the first-order partial derivatives can be given as follows:

$$\frac{\partial C_{SS}}{\partial a_1} = 2(a_1 - c_1) \tag{6}$$

$$\frac{\partial C_{SS}}{\partial a_2} = 2(a_2 - c_2) \tag{7}$$

$$\frac{\partial C_{SS}}{\partial a_3} = 2(a_3 - c_3) \tag{8}$$

Here, the homogeneous point coordinates a_4 and c_4 have been assumed to be unity without any loss in generality.

Thus, by iteratively perturbing the input actuator and solving the constraints for moving pivot coordinates, we can simulate a 5-SS mechanism and extract the path traced by the six unknown points $M_1, M_2, M_3, M_4, M_5, C$ on the coupler represented by their Cartesian coordinates $P = (P_x, P_y, P_z, 1)$. There exists an accuracy-storage trade-off for the simulation process. The accuracy of the path increases with decreased perturbation magnitude. However, this results in sampling more points on the path and thus needs more storage. The hyperparameters were set so that each defect-free segment of the motion consisted of at least 20 precision points.

To find the orientation of the moving platform, we first calculate the 4×4 homogeneous matrix $[\mathbf{D_f}]$ representing its spatial displacement using the six points on coupler as follows:

$$[\mathbf{D_f}] = [\mathbf{P_f}][\mathbf{P_i}]^{+} = [\mathbf{P_f}][\mathbf{P_i}]^{*}([\mathbf{P_i}][\mathbf{P_i}]^{*})^{-1} \tag{9}$$

where $[\mathbf{P_i}] = [P^T_{i,1}, P^T_{i,2}, \ldots, P^T_{i,6}]$ is the matrix representing initial coupler coordinates and $[\mathbf{P_f}] = [P^T_{f,1}, P^T_{f,2}, \ldots, P^T_{f,6}]$ represents the coupler coordinates after displacement. The + operator represents the pseudo-inverse while the * stands for conjugate transpose operation. The displacement matrix is of the form

$$[\mathbf{D_f}] = \begin{bmatrix} [\mathbf{R}] & \mathbf{d} \\ 0 & 1 \end{bmatrix} \tag{10}$$

where the 3×3 rotation matrix $[\mathbf{R}]$ is isolated from the 4×4 displacement matrix $[\mathbf{D_f}]$. This rotation matrix can then be converted into a unit quaternion $Q_f = (q_1, q_2, q_3, q_4)$ using the following equation (see Ref. [61] for details):

$$[\mathbf{R}] = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_4q_3) & 2(q_1q_3 + q_4q_2) \\ 2(q_1q_2 + q_4q_3) & q_4^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_4q_1) \\ 2(q_1q_3 - q_4q_2) & 2(q_2q_3 + q_4q_1) & q_4^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{11}$$

In this paper, we define a coupler motion as a time-sequenced series of coupler location and orientation, each denoted by three Cartesian coordinates (P_x, P_y, P_z) of the coupler point C and four quaternion coordinates (q_1, q_2, q_3, q_4). Thus, for our purposes, a spatial coupler motion is represented by a 7D curve. We generated a data set of 10,000 defect-free coupler motions using arbitrarily selected 5-SS mechanisms. This database represents a family of motions a general 5-SS mechanism can achieve. Figure 4 shows one of the simulated mechanisms and the motion generated by the moving frame attached to the coupler. Please note that the floating coupler is a rigid link, i.e., there is no relative motion between the light blue lines. In Fig. 3(*a*), we can see that the simulator takes in spatial mechanisms and outputs the complete motion raw database. The database took approximately 150 MB of storage and 1 h of computing time to generate using MATLAB on a PC with an Intel i5 processor. In the next section, we discuss the methodology used to refine this data set for machine learning purposes.

It should be noted that the branches of a mechanism are dependent on its actuation scheme. A different actuation scheme like using a motor at fixed pivots or a motor at moving pivot will result in different branches for the same mechanism. As a result, a separate data set will have to be created for each actuation scheme.

3 Data Preprocessing

Before the generated data can be used for machine learning, the data need to be cleaned, normalized, balanced, augmented, and masked.

3.1 Data Cleanup. Our data set stores the spatial motion of the moving platform as an array of n 7D data points. In the data set of 10,000 motion curves, we observe that n ranges from 1 to 1562 as can be seen in Fig. 5. Since curves with a very low number of data points do not capture their geometry well, we choose to remove them. Thus, curves made of less than 20 data points are removed resulting in a data set of 9472 curves.

When the solver is simulating a 5 SS mechanism, as outlined in Sec. 2, it may jump from one branch to another. Such motions are characterized by a C^1 discontinuity at the point where the branch jump occurs. Examples of two such curves are shown in Fig. 6.

Thus, to isolate such curves with discontinuity, the first-order differential is calculated and spikes in its magnitude are observed. The Z-score metric, also called the standard score, is used to characterize these spikes. A Z-score indicates how many standard deviations away an element is from the mean and is given as

$$z = \frac{X - \mu}{\sigma} \tag{12}$$

where μ is the mean, σ is the standard deviation, and X is the magnitude of the first-order differential of coupler motion. In our study, an outlier is defined as any curve having $Z_{\max} > 15$. Filtering out the outliers results in a cleaner database containing 8688 coupler curves.

3.2 Data Normalization. The remaining coupler motions are fitted with an interpolation curve. Fourth-order B-spline [62] interpolation is used for path data while spherical linear quaternion interpolation (Slerp) [63] is used for orientation data. Twenty-five data points are uniformly sampled on each curve leading to an arc-length parameterization. The benefit of using this arc-length

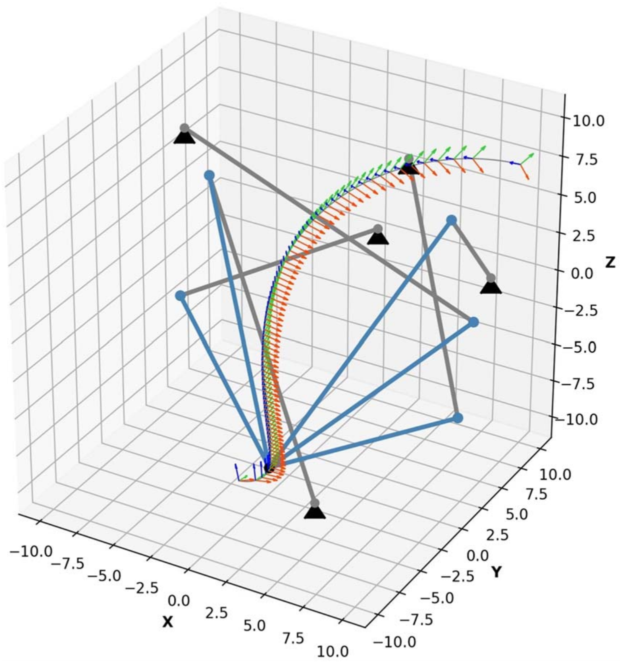


Fig. 4 A 5-SS mechanism simulation where black triangles are the fixed pivots, gray lines are the SS dyads, light blue lines are the floating coupler, and the red, green, and blue arrows represent the moving frame attached to the coupler

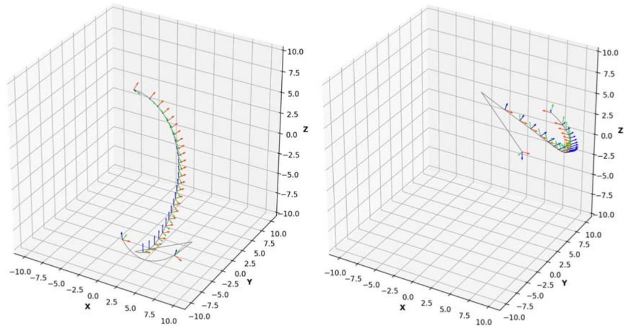


Fig. 6 Coupler curves with C_1 discontinuity caused due to a branch jump

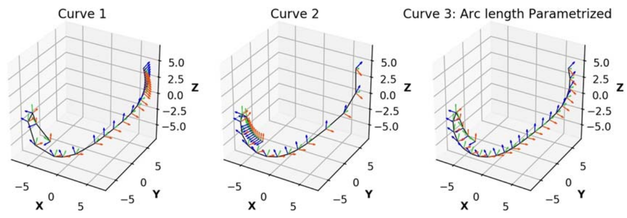


Fig. 7 Curve 1 and curve 2 represent the same geometric curve, but with different time parametrization. They share the same unique arc-length parametrization as shown in curve 3.

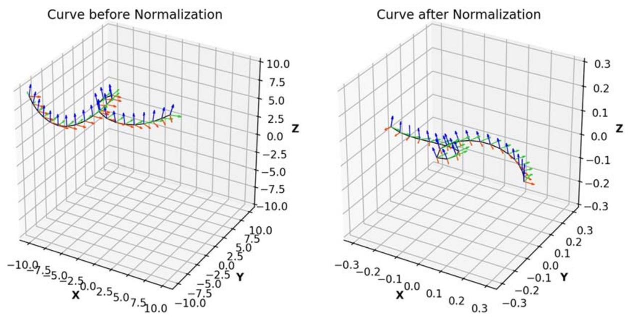


Fig. 8 Before and after normalizing a motion curve

parametrization is that it allows a unique coupler curve representation. This property is desirable since it makes comparing two curves with a similar trajectory but different time parametrization much easier as demonstrated in Fig. 7.

Creating a curve representation that is invariant to translation, rotation, and scaling is desirable when comparing curves. For the path data, first, the mean $(\bar{x}, \bar{y}, \bar{z})$ of the curve is calculated and it is translated to the origin of our coordinate system. Next, the principal axes of the path data are rotated to align with the Cartesian axes. The principal component axes are the eigenvectors of the covariance matrix of the point cloud that defines the curve. Also, the path data are scaled to unit arc-length. The orientation data are normalized by rotating the moving frame such that it is aligned with the fixed frame at the start of motion. The effect of normalization on a sample coupler motion can be seen in Fig. 8.

3.3 Data Balancing. The database in its present form is unbalanced, i.e., it has more samples of coupler curves which are more probable with a lesser number of samples of other more diverse examples. This leads to the algorithm not learning well since it comes across the more probable examples most of the time. To overcome this bias, a limited number of diverse motions are selected from the complete database by under-sampling similar curves. This balancing is also consistent with our goal to produce a variety of 5-SS mechanisms for a given problem to enhance the creativity of designers and provide them with a large set of solutions to choose from.

According to the fundamental theorem of space curves in differential geometry, every regular curve in three-dimensional space, with non-zero curvature, has its shape completely determined by its curvature and torsion [64]. Thus, a good metric to compare the similarity of two curves is curvature–torsion descriptor. For a spatial curve, curvature is a scalar measurement of the magnitude of the bending of the curve within the osculating plane. Torsion is a scalar measurement of the amount that the curve bends out of the osculating plane. The curvature and torsion can be calculated as follows:

$$\kappa = \frac{\|\mathbf{r}'(t) \times \mathbf{r}''(t)\|}{\|\mathbf{r}'(t)\|^3} \tag{13}$$

$$\tau = \frac{(\mathbf{r}'(t) \times \mathbf{r}''(t)) \cdot \mathbf{r}'''(t)}{\|\mathbf{r}'(t) \times \mathbf{r}''(t)\|^2} \tag{14}$$

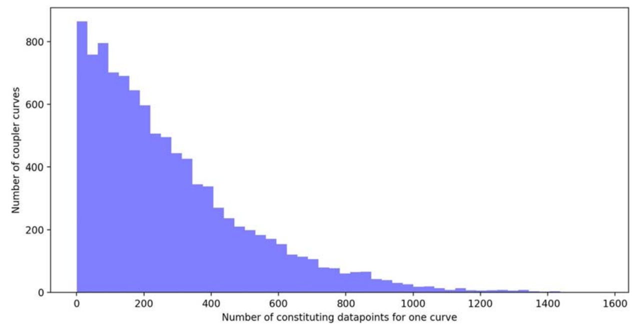


Fig. 5 Histogram showing number of data points in each motion curve included in database

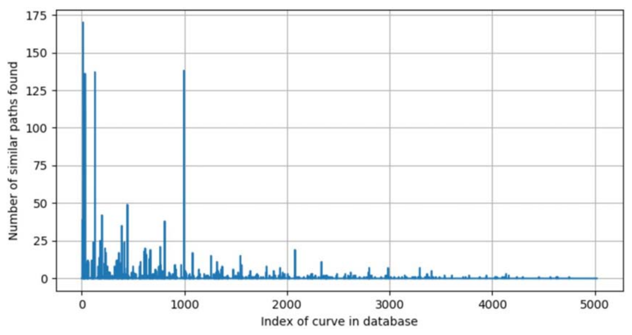


Fig. 9 Bar graph showing the number of similar curves found for each curve

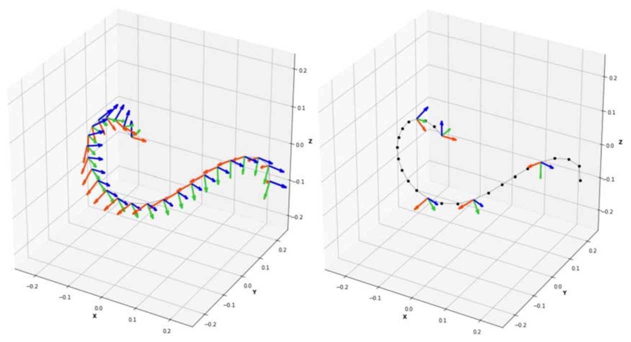


Fig. 11 An unmasked and masked 5-SS coupler motion curve

where $\mathbf{r}(t)$ is the curve and the n number of superscripts "′" define its nth order derivative. The curvature is always positive while the torsion can be negative.

For the path data, we calculate the curvature and torsion, while for the orientation data, we use the quaternion data directly to compare two curves. We define a similarity score (δ) as a weighted sum of l^2 norm of difference between path curvature (κ), path torsion (τ), and quaternion (Q) of two motions which is given as

$$\delta = \frac{w_1\|\kappa_2 - \kappa_1\| + w_2\|\tau_2 - \tau_1\| + w_3\|Q_2 - Q_1\|}{n} \qquad (15)$$

where w_1, w_2, and w_3 are weights associated with κ, τ, and Q, respectively, and n is the total number of constituent points in each curve. In our numerical experiments, we have found that weights $w_1 = 1$, $w_2 = 0.1$, and $w_3 = 2$ provide a fair balance between weighing of path and orientation data.

We select the similarity metric threshold such that if $\delta < 0.25$, the two curves are considered to be similar and one of them is dropped from the database. It can be observed in Fig. 9 that some curves occur up to 170 times in the database. On further exploring, we find that the common curves represent simple arcs and line trajectories and their reflections as seen in Fig. 10. Under-sampling similar curves lead to a balanced data set containing 5222 coupler paths.

3.4 Data Augmentation. In kinematics, it is known that if a curve is a valid coupler motion, its mirrored curve is also a valid motion. For the machine learning algorithm to gain this domain knowledge, coupler motions mirrored across XY, YZ, and XZ planes are added to the database. Thus, this step encourages the model to be invariant to mirror operations. Since reflecting the moving coordinate frame converts it from a right-handed system to a left-handed system, we attach a new right-handed frame to the coupler and recalculate the orientation data over the motion. After this step, our database contains 20,888 motion curves.

3.5 Data Masking. For the machine learning model to learn the underlying kinematic relationship between path-points and orientations, we synthetically mask the coupler motion database to create a path + pose constraint database. By providing the masked database as input and the unmasked database as output to the ML model, we can train it to learn the path-orientation relationship in a supervised manner. This step is essential to reformulate the Alt–Burmester problem into a motion synthesis problem. To create a masked motion from a n-point coupler motion, m-points ($0 \le m \le n$) are randomly selected and the orientations of these m points are set to zero. Also, a 1D binary mask is created representing the locations where the information is missing. An example of unmasked and masked motion curves is shown in Fig. 11. The mask value is 0 at m-points and 1 at the remaining points. This operation increases the database size to 417,760 masked motion curves.

Finally, some Gaussian noise is added to all the curves. This acts as a regularizer to the machine learning algorithm, encourages robust learning, and avoids overfitting. The magnitude of the added noise is up to 3% the maximum magnitude of motion curve coordinates.

This concludes the data preprocessing pipeline which generates two databases that are used to train the machine learning models. First is the database containing 7D pure coupler motion curves defined by three location coordinates and four orientation coordinates. Second is the database containing 8D masked coupler motion curves defined by three location coordinates, four orientation coordinates, and one mask coordinate. In Fig. 3(*a*), we can see that the preprocessing step generates the partial motion database and completes motion database. The data preprocessing pipeline was implemented using PYTHON and its libraries and took approximately 3 h of computing time on a PC with an Intel i5 processor.

4 Machine Learning Model Training

4.1 Training the Variational Autoencoder. Now that the database has been prepared; it can be used to train a machine

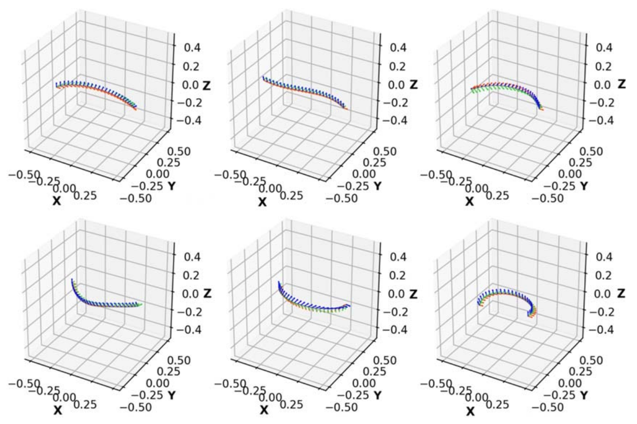


Fig. 10 Most common variety of motions in the database

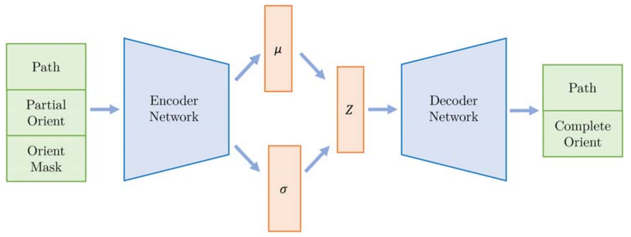


Fig. 12 Architecture of a VAE

Table 1 VAE model architectures that were tested and their performance: this table shows details of each architecture to indicate the number of hidden FC layers, the number of neurons in each layer, and the dimension of the latent space

Name	Encoder arch.	Latent (z) dim.	Decoder arch.	Training loss	Validation loss
VAE-FC-H1-Z15	(100)	15	(100)	32.2734	32.4453
VAE-FC-H1-Z30	(100)	30	(100)	32.2415	32.2879
VAE-FC-H1-Z60	(100)	60	(100)	32.0597	32.3151
VAE-FC-H2-Z15	(150,75)	15	(75,150)	28.1138	28.8488
VAE-FC-H2-Z30	(150,75)	30	(75,150)	28.6563	29.2358
VAE-FC-H2-Z60	(150,75)	60	(75,150)	28.1563	28.8108
VAE-FC-H3-Z15	**(200,100,60)**	**15**	**(60,100,200)**	**25.2833**	**25.8237**
VAE-FC-H3-Z30	(200,100,60)	30	(60,100,200)	25.5534	26.0324
VAE-FC-H3-Z60	(200,100,60)	60	(60,100,200)	26.2158	26.6811
VAE-FC-H4-Z15	(200,150,100,60)	15	(60,100,150,200)	25.5023	25.8424
VAE-FC-H4-Z30	(200,150,100,60)	30	(60,100,150,200)	26.3350	26.6101
VAE-FC-H4-Z60	(200,150,100,60)	60	(60,100,150,200)	28.2903	28.3822

Notes: For example, VAE-FC-H3-Z15 means that this is a VAE architecture with FC layers, three hidden (H3) layers, and 15-dimensional latent space (Z15). This is the seventh entry in this table and reading columns of this entry, and we see that the encoder network has 200, 100, and 60 neurons in its three layers, while the decoder network has 60, 100, and 200 neurons in its three layers.

learning model. The goal of our machine learning model is threefold: (1) to learn the path-orientation relationship and augment compatible missing orientations, (2) to learn the distribution of defect-free coupler motions and generate plausible motions similar to the user-inputted path and motion constraints, and (3) provide a low dimensional signature to the coupler motions which can easily be compared to other motions using a similarity metric.

To achieve this, we use a VAE, which is a type of generative neural network composed of an encoder and a decoder network. The encoder encodes input information, while the decoder decodes encoded information using probabilistic inference. We train a VAE network on a data set of complete and partial motion curves and allow it to predict complete motion curves by approximating the underlying distribution of observed data. Our general VAE architecture is shown in Fig. 12, where the encoder model finds the latent distributions defined as a multivariate Gaussian distribution defined by mean vector μ and standard deviation vector σ. A latent vector z can then be sampled from this distribution and used to generate unmasked motion trajectories using the decoder model. The samplings can represent data points not seen by the network during training and this is what makes a VAE rich and useful for generating new data. The encoder is represented as $q_\theta(z|X)$ where θ are the encoder weights and biases while a decoder is represented as $p_\phi(X|z)$ where ϕ denotes decoder weights and biases.

The input to our network is a concatenated vector $\mathbf{X} = (X_i, X_i + 1, \ldots, X_m)$, where m is the number of points on each motion curve and $X_i = (x_i, y_i, z_i, q_{1i}, q_{2i}, q_{3i}, q_{4i}, v_i)$. Here, v_i is the binary mask with value 1 or 0 to indicate if the orientation information is complete or not, respectively. For 20 points on a given motion curve, the $\mathbf{X}$ is a 160-dimensional vector, while the output of the network is defined by another vector $\hat{\mathbf{X}}_{\text{motion}}$, which consists of only the path- and orientation information.

The loss function used to train the VAE is defined as sum of reconstruction loss (RL) and Kullback–Leibler divergence (KL), which is given as

$$\text{Loss} = \text{RL} + \text{KL} \tag{16}$$

$$\text{RL} = \|\hat{\mathbf{X}}_{\text{motion}} - \mathbf{X}_{\text{motion}}\| \tag{17}$$

$$\text{KL} = \sum_{i=1}^{k} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1 \tag{18}$$

where the latent space is k dimensional and the reconstruction loss is the Euclidean norm, represented as $\|.\|$ operator, of the difference between generated motion ($\hat{\mathbf{X}}_{\text{motion}}$) and true (or labeled) motion ($\mathbf{X}_{\text{motion}}$).

This supervised training step for VAE is visualized in Fig. 3(*a*). It can be seen that during training, the partial motion database serves as input while the complete motion database serves as output to the VAE.

Multiple VAEs with different depths and bottlenecks were tested to find the best architecture. The capacity of a network increases with increasing depths and it can describe a much more complex function. However, due to the problem of vanishing gradient, deep networks tend to be harder to train. Thus, there exists an optimal depth that balances complexity and trainability. Similarly, the narrower the bottleneck layer, the better is the dimensionality reduction. However, reducing the width too much can lead to excessive loss of information. Networks with different depths (up to four) and bottleneck layer widths (15, 30, and 60) were tested and the results are given in Table 1. Each VAE was trained for 1000 epochs with a batch size of 256 using Adam (adaptive moment estimation) optimizer.

In our computational experiments, we found that the VAE-FC-H3-Z15 performed the best and had the lowest validation loss. It contains three hidden fully connected (FC) layers consisting of 200, 100, and 60 nodes each, rectified linear unit activation function after each layer, and the bottleneck layer z contains 15 nodes. The training curves of VAE-FC-H3-Z15 can be seen in Fig. 13.

A few example outputs generated using VAE-FC-H3-Z15 are shown in Fig. 14, where an input constraint (top-left) generates three motion trajectories sampled from the underlying distribution. We note that the VAE can generate variations in both path and orientation data using its understanding of family of 5-SS mechanism motions.

The model training pipeline was implemented using PYTHON and KERAS. We initially used Google Colaboratory in GPU mode where each model took about 5–6 h to train. Later, we used a workstation

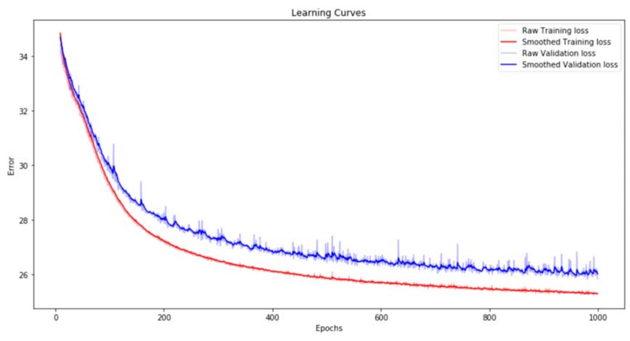


Fig. 13 Training losses for the FC VAE

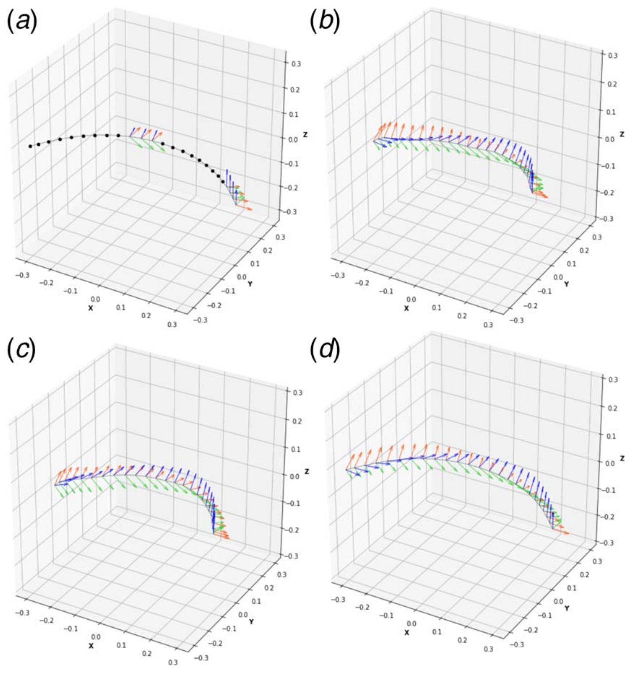


Fig. 14 Motion trajectories generated using VAE for a Alt–Burmester input constraint: (*a*) input masked motion, (*b*) generated motion 1, (*c*) generated motion 2, and (*d*) generated motion 3

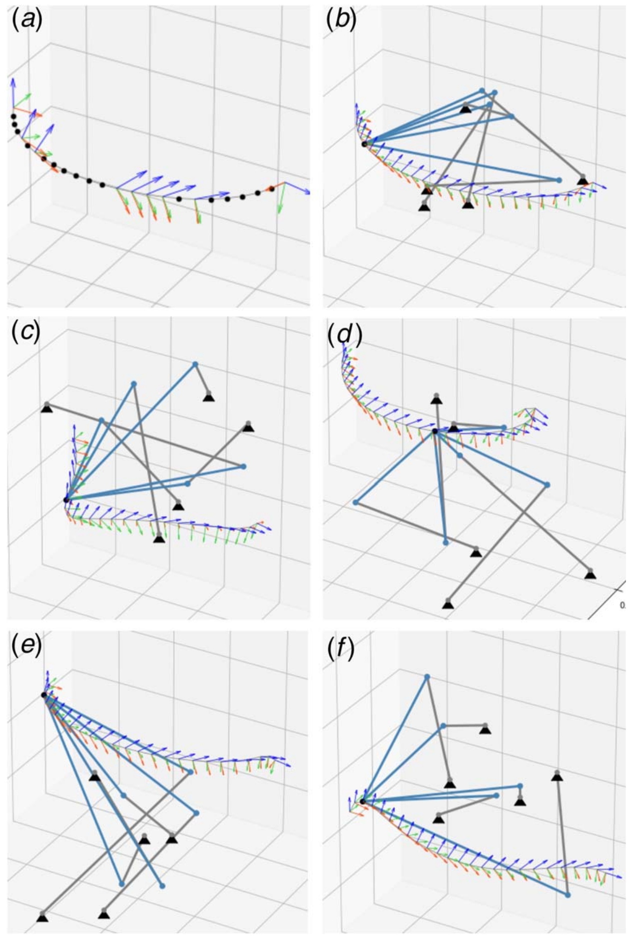


Fig. 15 Example 1: (*a*) input Alt–Burmester problem, (*b*) synthesized mechanism 1, (*c*) synthesized mechanism 2, (*d*) synthesized mechanism 3, (*e*) synthesized mechanism 4, and (*f*) synthesized mechanism 5

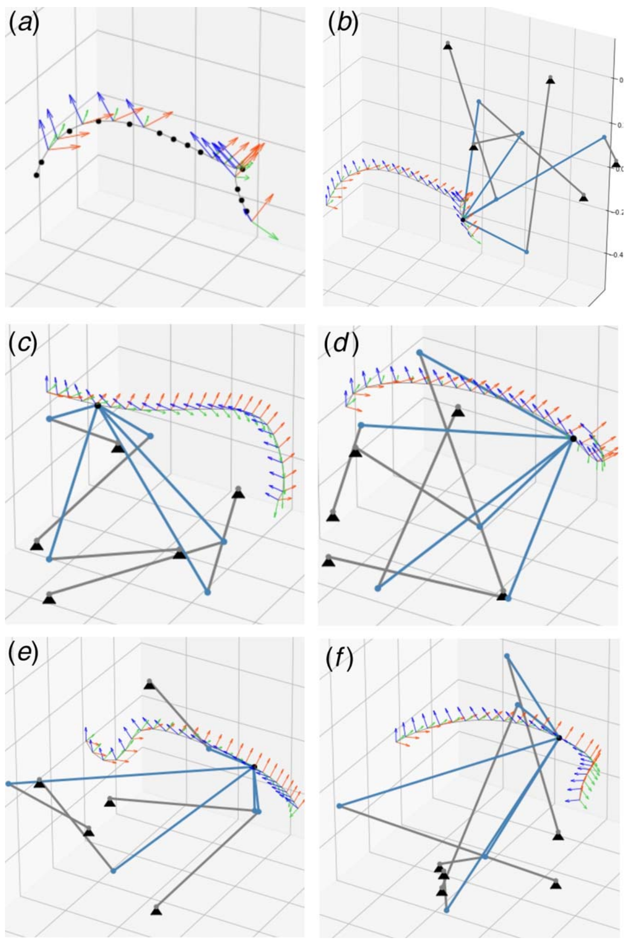


Fig. 16 Example 2: (*a*) input Alt–Burmester problem, (*b*) synthesized mechanism 1, (*c*) synthesized mechanism 2, (*d*) synthesized mechanism 3, (*e*) synthesized mechanism 4, and (*f*) synthesized mechanism 5

with an Intel i7 processor and an Nvidia Titan X GPU and got the training time down to almost an hour per model. The code is available on Github.[2]

4.2 Creating a Hierarchical Database. One of the most powerful features of the VAE is its ability to provide a compact representation of each coupler motion using its latent vector. We can use this compact signature for fast and efficient database search and clustering algorithms. Thus, once the training is completed, the encoder module is used to generate latent vector signatures of each masked motion in the database denoted by the μ vector. Then, these signatures are clustered into 500 groups using the K-means clustering algorithm. The distance metric used is the Euclidean distance. As a result, we get 500 cluster centers subdividing the original data set of 400k+ coupler constraint curves.

5 Mechanism Synthesis for the Spatial Alt–Burmester Problem

Once the VAE is trained and the hierarchical database created, we can use them to synthesize mechanisms. When the user inputs a curve consisting of path-points and poses, it is normalized and runs through the encoder network of VAE to find the μ and σ vectors. Multiple z vectors are then sampled from the latent distribution which denotes a family of feasible curve signatures. These

[2]https://github.com/ssharma1991/ml˙kinsys

curve signatures are then compared to each of the cluster centers using the L^2 norm difference metric. Once a center is selected, the best available mechanism within the cluster can be returned to the user as a feasible solution. Thus, the user can find multiple defect-free solution mechanisms.

Alternatively, instead of only using the encoder, we can use the complete VAE to transform any Alt–Burmester problem into a pure motion synthesis problem. With the conditioned output available from the VAE, existing synthesis algorithms can also be used to generate 5-SS mechanisms [41,42]. With compatible orientations available, it is also likely that the traditional synthesis algorithms would produce better results.

We visualize the inference pipeline explained above in Fig. 3(*b*), where one can see the flow of data in both cases: using the hierarchical database (latent signature database) or using the classical synthesis approach.

6 Examples

In this section, we provide two examples of our algorithm in action. In the examples, we input a partial spatial motion curve. The trajectory is then processed by the encoder of the VAE-FC-H3-Z15 resulting in a 15-dimensional Gaussian distribution specified by μ and σ. We sample five latent vectors z from this distribution and look up the closest cluster centers in our database. In the cluster, we find the best approximation of the coupler motion available and provide it as a solution.

The input motion curves are shown in the first plot in each of Figs. 15 and 16. The other plots show a prospective 5-SS solution that closely matches the target constraints. More mechanisms can be generated by sampling additional latent vectors from the VAE.

7 Conclusions

In this paper, we have presented a complete pipeline for the defect-free synthesis of spatial 5-SS mechanisms consisting of mechanism simulation, data preprocessing, and machine learning stages. To generate coupler motion data, we use a geometric constraint-based numerical approach which uses the Newton–Raphson method. Then, the data are pre-processed and masked for robust learning. Finally, semi-supervised machine learning techniques of VAE and K-mean clustering are used to efficiently find solution mechanisms. While our focus was on the spatial 5-SS mechanisms, the algorithm presented is general enough to solve the Alt–Burmester problem for any spatial mechanism and generate multiple solutions. While this paper is one of the first attempts at using machine learning to solve the spatial kinematic synthesis problem, it is a mystery as to what does a latent space truly represent. Our current implementation for the 5-SS mechanisms uses a specific mode of actuation. It is expected that other modes of actuation would require re-training the VAE model. There is also an issue of sparse-coverage of high dimensional design space for spatial mechanisms by a given data set. In the paper, we attempt to reduce this design space by focusing on "uniquely shaped" motions, but better ways of addressing this "curse of dimensionality" need to be explored. Future work could also focus on extending the presented approach to multidimensional spatial mechanisms.

Acknowledgment

This work has been financially supported by The National Science Foundation under a research grant #CMMI-1563413 to Stony Brook University. All findings and results presented in this paper are those of the authors and do not represent those of the funding agencies.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available. The authors attest that all data for this study are included in the paper.

References

[1] Erdman, A. G., and Sandor, G. N., 1991, *Advanced Mechanism Design: Analysis and Synthesis*, 2nd ed., Vol. 2, Prentice-Hall, Englewood Cliffs, NJ.
[2] McCarthy, J. M., and Soh, G. S., 2010, *Geometric Design of Linkages*, Vol. 11, Springer, New York.
[3] Alt, H., 1923, "Uber die Erzeugung gegebener ebener Kurven mit Hilfe des Gelenkvierecks," ZAMM, **3**(1), pp. 13–19.
[4] Freudenstein, F., 1954, "An Analytical Approach to the Design of Four-Link Mechanisms," Trans. ASME, **76**(3), pp. 483–492.
[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] Blechschmidt, J. L., and Uicker, J. J., 1986, "Linkage Synthesis Using Algebraic-Curves," J. Mech. Trans. Autom. Des. Trans. ASME, **108**(4), pp. 543–548.
[8] 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.
[9] Nolle, H., and Hunt, K. H., 1971, "Optimum Synthesis of Planar Linkages to Generate Coupler Curves," J. Mech., **6**(3), p. 267.
[10] Sancibrian, R., Viadero, F., Garcia, P., and Fernandez, A., 2004, "Gradient-Based Optimization of Path Synthesis Problems in Planar Mechanisms," Mech. Mach. Theory, **39**(8), pp. 839–856.
[11] 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.
[12] Wu, J., Ge, Q. J., and Gao, F., 2009, "An Efficient Method for Synthesizing Crank-Rocker Mechanisms for Generating Low Harmonic Curves," ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Paper No. DETC2009-87140.
[13] Sharma, S., Purwar, A., and Ge, Q. J., 2019, "An Optimal Parametrization Scheme for Path Generation Using Fourier Descriptors for Four-Bar Mechanism Synthesis," ASME J. Comput. Inform. Sci. Eng., **19**(1), p. 014501.
[14] Chu, J. K., and Sun, J. W., 2010, "A New Approach to Dimension Synthesis of Spatial Four-Bar Linkage Through Numerical Atlas Method," ASME J. Mech. Rob., **2**(4), p. 041004.
[15] Wandling Sr, G. R., 2000, "Synthesis of Mechanisms for Function, Path, and Motion Generation Using Invariant Characterization, Storage and Search Methods," Ph.D. thesis, Iowa State University, Ames, IA.
[16] Vasiliu, A., and Yannou, B., 2001, "Dimensional Synthesis of Planar Mechanisms Using Neural Networks: Application to Path Generator Linkages," Mech. Mach. Theory, **36**(2), pp. 299–310.
[17] Galan-Marin, G., Alonso, F. J., and Del Castillo, J. M., 2009, "Shape Optimization for Path Synthesis of Crank-Rocker Mechanisms Using a Wavelet-Based Neural Network," Mech. Mach. Theory, **44**(6), pp. 1132–1143.
[18] Deshpande, S., and Purwar, A., 2019, "A Machine Learning Approach to Kinematic Synthesis of Defect-Free Planar Four-Bar Linkages," ASME J. Comput. Inform. Sci. Eng., **19**(2), p. 021004.
[19] 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.
[20] Deshpande, S., and Purwar, A., 2020, "An Image-Based Approach to Variational Path Synthesis of Linkages," ASME J. Comput. Inform. Sci. Eng., **21**(2), p. 021005.
[21] Chiang, C. H., 2000, *Kinematics of Spherical Mechanisms*, Krieger Pub., Malabar, FL.
[22] Chu, J., and Sun, J., 2010, "Numerical Atlas Method for Path Generation of Spherical Four-Bar Mechanism," Mech. Mach. Theory, **45**(6), pp. 867–879.
[23] Mullineux, G., 2011, "Atlas of Spherical Four-Bar Mechanisms," Mech. Mach. Theory, **46**(11), pp. 1811–1823.
[24] Premkumar, P., Dhall, S. R., and Kramer, S. N., 1988, "Selective Precision Synthesis of the Spatial Slider Crank Mechanism for Path and Function Generation," ASME J. Mech. Trans. Autom. Des., **110**(3), pp. 295–302.
[25] Premkumar, P., and Kramer, S. N., 1989, "Position, Velocity, and Acceleration Synthesis of the RRSS Spatial Path-Generating Mechanism Using the Selective Precision Synthesis Method," ASME J. Mech. Trans. Autom. Des., **111**(1), pp. 54–58.
[26] Ananthasuresh, G. K., and Kramer, S. N., 1994, "Analysis and Optimal Synthesis of the RSCR Spatial Mechanisms," ASME J. Mech. Des., **116**(1), pp. 174–181.
[27] Jiménez, J., Álvarez, G., Cardenal, J., and Cuadrado, J., 1997, "A Simple and General Method for Kinematic Synthesis of Spatial Mechanisms," Mech. Mach. Theory, **32**(3), pp. 323–341.

[28] Sun, J. W., Mu, D. Q., and Chu, J. K., 2012, "Fourier Series Method for Path Generation of RCCC Mechanism," Proc. Inst. Mech. Eng. Part C; J. Mech. Eng. Sci., **226**(3), pp. 816–827.

[29] Sharma, S., and Purwar, A., 2020, "Path Synthesis of Defect-Free Spatial 5-SS Mechanisms Using Machine Learning," ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 83990, Virtual, Online, Aug. 17–19, American Society of Mechanical Engineers, p. V010T10A034.

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

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

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

[33] 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. Inform. Sci. Eng., **12**(4), p. 041003.

[34] Ge, Q. J., Purwar, A., Zhao, P., and Deshpande, S., 2016, "A Task Driven Approach to Unified Synthesis of Planar Four-Bar Linkages Using Algebraic Fitting of a Pencil of G-Manifolds," ASME J. Comput. Inform. Sci. Eng., **17**(3), p. 031011.

[35] Bodduluri, R. M. C., and McCarthy, J. M., 1992, "Finite Position Synthesis Using Image Curve of a Spherical Four-Bar Motion," ASME J. Mech. Des., **114**(1), pp. 55–60.

[36] Lin, C.-C., 1998, "Complete Solution of the Five-Position Synthesis for Spherical Four-Bar Mechanisms," J. Marine Sci. Technol., **6**(1), pp. 17–27.

[37] Ruth, D., and McCarthy, J., 1999, "The Design of Spherical 4R Linkages for Four Specified Orientations," Mech. Mach. Theory, **34**(5), pp. 677–692.

[38] Brunnthaler, K., Schrocker, H., and Husty, M., 2006, *Synthesis of Spherical Four-Bar Mechanisms Using Spherical Kinematic Mapping* (Advances in Robot Kinematics), Springer, Dordrecht.

[39] Zhuang, Y., Zhang, Y., and Duan, X., 2015, "Complete Real Solution of the Five-Orientation Motion Generation Problem for a Spherical Four-Bar Linkage," Chin. J. Mech. Eng., **28**(2), pp. 258–266.

[40] Li, X., Zhao, P., Purwar, A., and Ge, Q., 2018, "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.

[41] Innocenti, C., 1995, "Polynomial Solution of the Spatial Burmester Problem," ASME J. Mech. Des., **117**(1), pp. 64–68.

[42] Liao, Q., and McCarthy, J. M., 1997, "On the Seven Position Synthesis of a 5-SS Platform Linkage," ASME J. Mech. Des., **123**(1), pp. 74–79.

[43] Plecnik, M. M., and McCarthy, J. M., 2012, "Design of a 5-SS Spatial Steering Linkage," ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, IL, Aug. 12–15, American Society of Mechanical Engineers Digital Collection, pp. 725–735.

[44] Li, X., Ge, Q. J., and Gao, F., 2014, "A Unified Algorithm for Geometric Design of Platform Linkages With Spherical and Plane Constraints," 38th Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 5B, Buffalo, NY, Aug. 17–20, p. v05BT08A101.

[45] Sharma, S., and Purwar, A., 2020, "Unified Motion Synthesis of Spatial Seven-Bar Platform Mechanisms and Planar-Four Bar Mechanisms," ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 83990, Virtual, Online, Aug. 17–19, American Society of Mechanical Engineers, p. V010T10A033.

[46] Tong, Y., Myszka, D. H., and Murray, A. P., 2013, "Four-Bar Linkage Synthesis for a Combination of Motion and Path-Point Generation," Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 6A: 37th Mechanisms and Robotics Conference, Portland, OR, Aug. 4–7, ASME, p. V06AT07A054.

[47] Brake, D. A., Hauenstein, J. D., Murray, A. P., Myszka, D. H., and Wampler, C. W., 2016, "The Complete Solution of Alt–Burmester Synthesis Problems for Four-Bar Linkages," ASME J. Mech. Rob., **8**(4), p. 041018.

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

[49] Sharma, S., Purwar, A., and Ge, Q. J., 2019, "A Motion Synthesis Approach to Solving Alt–Burmester Problem by Exploiting Fourier Descriptor Relationship Between Path and Orientation Data," ASME J. Mech. Rob., **11**(1), p. 011016.

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

[51] Roth, B., and Freudenstein, F., 1963, "Synthesis of Path-Generating Mechanisms by Numerical Methods," ASME J. Eng. Ind., **85**(3), pp. 298–304.

[52] 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.

[53] Sebe, N., Cohen, I., Garg, A., and Huang, T., 2005, *Machine Learning in Computer Vision* (Computational Imaging and Vision), Springer, Dordrecht.

[54] Hemanth, D., and Estrela, V., 2017, *Deep Learning for Image Processing Applications* (Advances in Parallel Computing), IOS Press, Amsterdam, The Netherlands.

[55] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C., 2000, "Image Inpainting," Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'00, New Orleans, LA, July, ACM Press/Addison-Wesley Publishing Co., pp. 417–424.

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

[57] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Nets," *Advances in Neural Information Processing Systems 27*, Z., Ghahramani, M., Welling, C., Cortes, N. D., Lawrence, and K. Q., Weinberger, eds., Curran Associates, Inc., pp. 2672–2680.

[58] Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N., 2017, "Semantic Image Inpainting With Deep Generative Models," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, July 21–26, pp. 6882–6890.

[59] Ivanov, O., Figurnov, M., and Vetrov, D., 2018, "Variational Autoencoder With Arbitrary Conditioning," International Conference on Learning Representations, New Orleans, LA, May 6–9, preprint arXiv:1806.02382.

[60] Sharma, S., and Purwar, A., 2020, "Using a Point-Line-Plane Representation for Unified Simulation of Planar and Spherical Mechanisms," ASME J. Comput. Inform. Sci. Eng., **20**(6), p. 061002.

[61] Purwar, A., and Ge, Q. J., 2005, "The Effects of Weights in Rational Motion Design," ASME J. Mech. Des., **127**(5), pp. 967–972.

[62] Piegl, L., and Tiller, W., 1995, *The NURBS Book*, Springer, Berlin.

[63] Dam, E. B., Koch, M., and Lillholm, M., 1998, *Quaternions, Interpolation and Animation*, Vol. 2, Citeseer.

[64] Kühnel, W., 2015, *Differential Geometry* (Student Mathematical Library), American Mathematical Society, Providence, RI.