Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Canada

DRAFT: DETC2018-85578

A MACHINE LEARNING APPROACH TO KINEMATIC SYNTHESIS OF DEFECT-FREE PLANAR FOUR-BAR LINKAGES

Shrinath Deshpande, Anurag Purwar[*]
Computer-Aided Design and Innovation Lab
Department of Mechanical Engineering
Stony Brook University
Stony Brook, New York, 11794-2300

ABSTRACT

The past forty years of research in mechanism synthesis has witnessed an unprecedented volume of work in formulating and solving planar four-bar linkage synthesis problems. However, finding practical and useful mechanisms for the motion synthesis problem has proven to be elusive, as a large majority of mechanisms turn out to be defective with respect to their assembly modes. Most methods formulate the problem as a discrete precision position problem, which inherently ignores the continuity information in the input, resulting in linkages with branch-, circuit- and order-defects. In this paper, we bring together diverse fields of pattern recognition, machine learning, artificial neural network, and computational kinematics to present a novel approach that solves this problem both efficiently and effectively. At the heart of this approach lies an objective function that compares the motion as a whole thereby capturing designer's intent. In contrast to widely used structural error or loop-closure equation based error functions which convolute the optimization by considering shape, size, position, and orientation simultaneously, this objective function computes motion difference in a form, which is invariant to similarity transformations. We employ auto-encoder neural networks to create a compact and clustered database of invariant motions of known linkages. The query is raised in the database for nearest neighbors, which are either solutions or good initial conditions for fast local optimization techniques. In spite of highly non-linear parameters space, our approach discovers a wide pool of defect-free solutions very quickly. We show that by employing proven machine learning techniques, this work could have far-reaching consequences to creating a multitude of useful and creative conceptual design solutions for mechanism synthesis problems, which go beyond planar four-bar linkages.

keywords: *Machine Learning, Artificial Neural Network (ANN), Path Synthesis, Motion Synthesis, Planar Four-bar linkage, Circuit- Order-, and Branch-Defect*

1 Introduction

Classical mechanism synthesis problem deals with computing type and dimensions of linkage system for performing specific tasks, which are categorized as path, motion and function generation; for an exhaustive reference, see books from McCarthy and Soh [1], Sandor and Erdman [2], Hunt [3], Hartenberg and Denavit [4], Suh and Radcliffe [5], and Lohse [6]. Various approaches have been proposed in the literature with aim of finding acceptable solutions for practical situations. However, the majority of the approaches do not account for circuit and branch defect in the synthesis process, which can render the solution useless for practical applications. These defects in the single degree of freedom mechanisms are thoroughly discussed by Chase and Mirth [7].

In this paper, we 1) employ pattern recognition and computational shape analysis, to create a similarity transform-independent signature for coupler path and motion, 2) exploit the non-linear nature of relationship between the linkage param-

[*]Address all correspondence to this author at anurag.purwar@stonybrook.edu

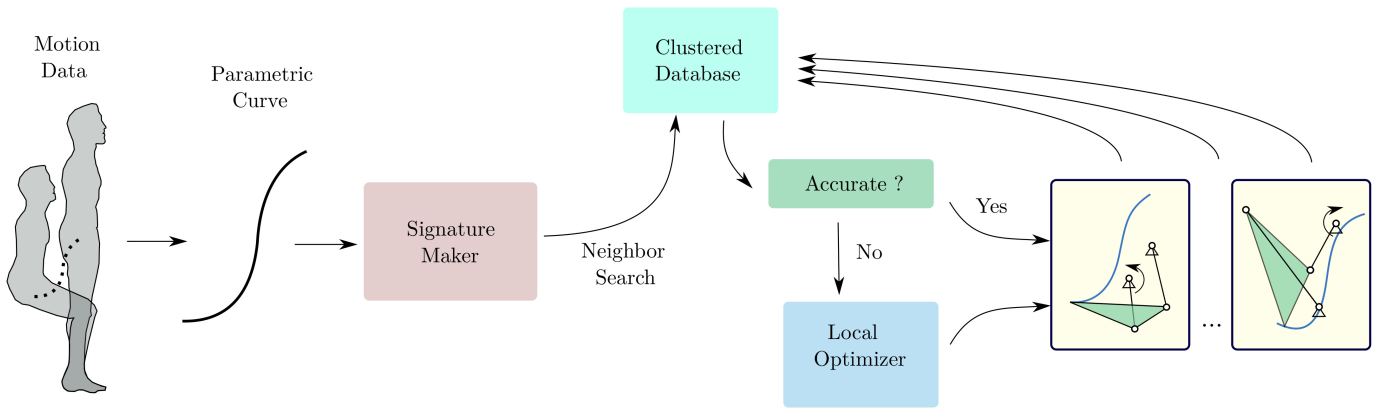

FIGURE 1: The Machine Learning approach begins by creating an invariant signature for the path and the motion data, which facilitates a compact and hierarchical clustered database and an auto-encoder Neural Network trained to elicit good, defect-free solutions or subjected to local, fast optimization. The results are astonishingly good, defect-free concept design solutions for input problems.

eters and coupler motions to create a sensitive, wide-ranging, compact, and efficient database with hierarchical clustering, and 3) enable dimensionality reduction using Auto-encoder Neural Networks. When a user inputs a motion or a path, a query representing invariant signature [1] of the input is raised for k nearest neighbors among cluster centers in the database. These k neighbors, if needed, are subjected to fine-tuning by local optimization to obtain a set of defect-free solutions. We define a novel objective function for defect-free motion generation problems and present an algorithm for partial matching of motions. The objective function that drives the synthesis process computes a distance measure of dissimilarity between the task motion and the coupler motion generated by current linkage parameters. This distance measure of dissimilarity inherently requires continuity of motion, thus ensuring that the output mechanism is defect-free throughout the task. The Fig. 1 illustrates an overview of our method, which is codified in Algorithm 1.

Path synthesis methods based on Fourier analysis do take continuity information of the coupler path into account, however, most of them are defined only for closed loop curves. Ullah and Kota [8] have presented an invariant approach towards representation and synthesis of closed loop paths through shape optimization. They use a combination of global and local search methods for optimizing Fourier Deviant function, to compute the dimensions of planar four-bar linkages without an initial guess. Wu et al. [9] presented a method for open and closed path generation based on finite Fourier series. In the case of motion generation, Li et al. [10] have developed a Fourier descriptor-based approach for approximate motion generation.

Due to highly nonlinear nature of the problem, most optimization based methods require a good initial condition to start. Thus, an atlas or a design library is used to provide good initial conditions. McGarva [11] took the earliest approach towards creating a library for coupler trajectories based on a harmonic analysis. Wandling [12] has presented an atlas-based approach, where coupler paths and motions are stored in terms of Fourier Transforms. Input motion is searched for neighbors based on Euclidean distances of Fourier Transforms. Yue et al. [13] presented a similar approach of path generation using P-Type Fourier Descriptor applicable for open curves. In their approach, a task curve is transformed into normalized Fourier coefficients and queried for the nearest neighbor search. The best match is returned as the solution to input. Chu et al. [14] presented an atlas-based method for synthesizing spatial four-bar linkages for function generation problems, where orientation data is stored in terms of Fourier descriptors.

The above methods generate data based on uniform sampling in the linkage parameter space. Given highly nonlinear mapping between linkage parameters and coupler trajectory, this way of sampling leads to a non-uniform sampling of trajectory space causing under-representation of possible motions. We address this issue by employing log-normal distribution in the linkage parameter space to generate data and perform compact clustering of the data using machine learning techniques. Hierarchy is created in the database by means of clustering, where the top level comprises of data points called cluster centers, which are representative of the cluster points in lower levels. Wandling [12] and Yu et al. [13] have built libraries with all possible coupler curves, where one curve is broken down into many segments for creating data for partial curves. In contrast to this, we need to store only one curve representing all the segments in

[1] path or motion signature

it, as our representation facilitates partial matching. None of the other previous methods facilitate this partial matching of open motion curve into another open or closed curve, which significantly contributes to providing a large number of solutions, and reduces the data requirement even further.

Rest of this paper is organized as follows. Section 2 presents the computation of motion and path signatures. Section 3 is comprised of evaluation criterion for signatures based on the shape similarity, which leads to the formulation of error function for optimization. Section 4 discusses the nature of objective function via sensitivity analysis at a singularity. Section 5 presents the database generation and clustering using auto-encoders for efficient sampling and query operations. Finally, two case studies are presented in section 6 to illustrate the efficiency and efficacy of the method.

Algorithm 1: Planar Linkage Synthesis

```
Input  : Task Motion $\{x_i, y_i, \theta_i\}_{i=1}^N$ or Path$\{x_i, y_i\}_{i=1}^N$
Output: Linkage Parameters l: $l_1$, $l_2$, ...
1  signature = calculateSignature(Input);
2  distances = [];
3  for centerPoint in clusterCenters do
4      distances.push(getDistance(signature, centerPoint))
5  end
6  kNeighbors = getNeighbors(distances, k) for neighbor in
     kNeighbors do
7      if threshold < neighbor.distance then
8          return neighbor.LinkParameters
9      else
10         return Optimize(neighbor.LinkParameters)
11     end
12 end
```

2 Signatures of Coupler Path and Motion

Focus of the paper is on a novel method for mechanism synthesis that takes a parametric motion $(x : x(t), y : y(t), \theta : \theta(t))$ or path $(x : x(t), y : y(t))$ as the input, and returns defect-free linkages that produce similar motion or path. The input is transformed into a representation, termed as a signature, which is invariant to similarity operations, viz. reflection, rotation, translation, and scaling. Signatures for path and motion are termed as path signature and motion signature, respectively. For calculating path signature, we use formulations developed by Cui et.al [15].

Consider a motion given in parametric form as, $x : x(t), y : y(t), \theta : \theta(t)$. Where, $\theta(t)$ is the change in orientation along the path with respect to initial orientation. It should be noted that

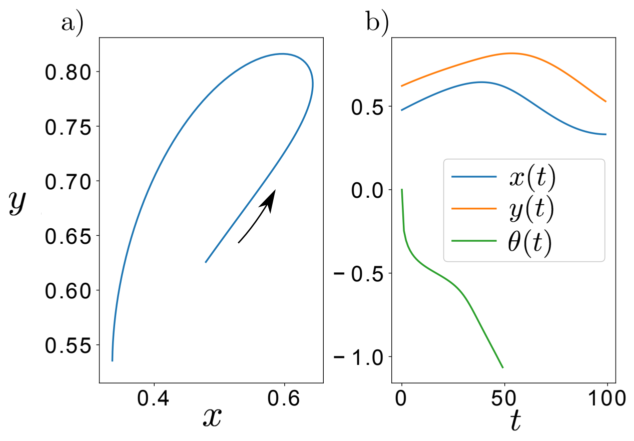

FIGURE 2: a) The path of the input motion along with direction of parametrization. b) Motion components $x(t), y(t), \theta(t)$ are plotted against parameter t.

$\theta(t)$ is a continuous curve with domain $(-\infty, \infty)$ in contrast to conventional domain i.e. $[-\pi, \pi]$.

Curvature $\kappa(t)$ of the path $(x : x(t), y : y(t))$ and its integral $K(t)$ is given by,

$$\kappa(t) = \frac{\ddot{y}(t)\dot{x}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{\left(\frac{3}{2}\right)}}, \quad (1)$$

$$K(t) = \int_0^t |\kappa(t)| dt, \quad (2)$$

where $\dot{x}(t), \ddot{x}(t)$ are first and second order derivative with respect to parameter t. As an example, the parametric motion could be a B-spline motion as shown in Fig 2. We compute $\kappa(t)$ and $K(t)$ along the direction of t, using Eq. (1) and Eq. (2) respectively. Figure 3 shows computed curvature and its unsigned integral of the coupler path shown in Fig. 2. It can be seen that the curvature is small at start, increases as the curve bends along the path and drops once again as the path straightens out. It is obvious that curvature plot will reverse if the direction of parameterizations reverses, while the integral is a monotonically increasing function.

Now, we re-sample the curvature at equal intervals of $K(t)$, which is equivalent to plotting κ vs K in Fig 4(a). This is done by finding the parameter values of t where K changes uniformly. For practical purposes, we find an array of the parameter t such that K increments by 0.1. For each value of t in that array, we compute $\kappa(K)$ and $\theta(K)$ and store it as the path and motion signatures respectively. We note that there is a one-to-one mapping between t and K.

These signatures are invariant under similarity transformations; for proof see [15]. We know that curvature changes inversely to scale of the curve, so when it is integrated along the

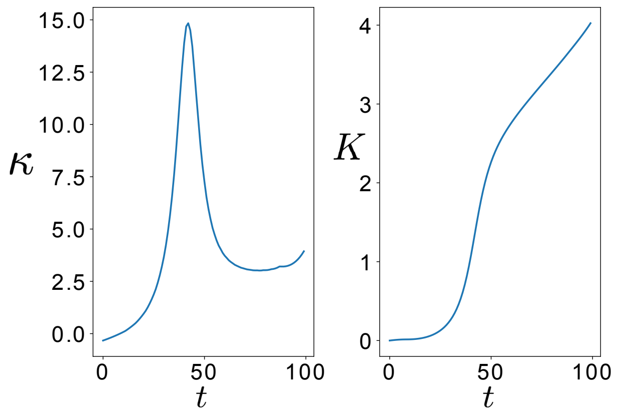

FIGURE 3: Curvature and its unsigned integral for the path shown in Fig. 2

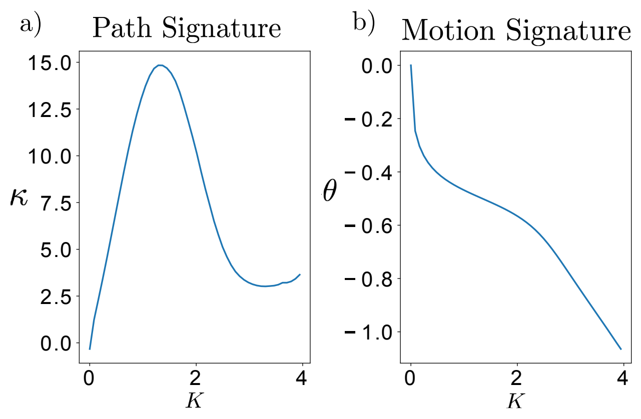

FIGURE 4: Path and motion signatures of the motion shown in Fig. 2

scaled curve, the scale factor cancels itself out. Reflection operation produces flipped path signature, but motion signature remains invariant. Figure 4 shows path and motion signature obtained for the motion depicted in Fig. 2. It is important to note, that the signature depends on the direction of parameter t. The procedure of signature calculation presented in this section is given in Algorithm 2.

3 Signature Matching and Error Function

The signatures obtained in previous steps contain important information about the shape of the trajectory. In this section, we formulate functions that evaluate the similarity between two trajectories based on their signatures. These distance functions can be used as error metric, which can be minimized using optimization methods.

Algorithm 2: Calculate Invariant Signatures

```
Input  : Twice Differentiable Parametric Representation
         of Motion $(x : x(t), y : y(t), \theta : \theta(t))$
Output: signature //discretized signal in form of an array
$\kappa(t)$ = ComputeCurvature$(x, y)$ using Eq. 1
$K(t)$ = IntergrateCumulatively$(\kappa(t))$ using Eq. 2
motionSignature = []
pathSignature = []
for $i = 0 \rightarrow max(K)$ do
    $tmp$ = (value of $t$ corresponding to which $K$ has value $i$)
    $i = i + 0.1$
    motionSignature.push($\kappa(tmp)$)
    pathSignature.push($\kappa(tmp)$)
end
return PathSignature, MotionSignature
```

3.1 Partial Matching of Path Signatures

When a path query is raised, it can be very useful to know whether this path matches with a part of a path from the database. This subsection presents a method for determining this partial similarity.

Let us consider two coupler paths; namely *Part* and *Whole* as shown in 5. Let p and W be their signatures respectively, where W completely contains p as shown in Fig. 6. The orientation information shown in Fig. 5 is ignored for path matching. It will be used later for matching of motion signatures. The partial matching works as follows:

1. p and W are expressed in terms of arrays and W must contain more points than p.
2. p is slid with offset index j along W.
3. For each offset j, we compute normalized cross-correlation [16] function given by,

$$Cn(j,p,W) = \left| \sum_{i}^{p_{span}} \frac{(W(i+j) - \bar{W}(j : j+p_{span}))(p(i) - \bar{p})}{\sqrt{\sum_i^{p_{span}} (W(i+j) - \bar{W}_{p_{span}})^2 \sum_i^{p_{span}} (p(i) - \bar{p})^2}} \right|, \tag{3}$$

where $Cn(j,p,W)$ is the normalized cross-correlation value when p is matched against W at j^{th} index; p_{span} is length of the array p and $\bar{W}(j : j+p_{span})$ is mean of the values of array W between index range of $(j, j+p_{span})$.

Here, p acts as a template that tries to find the best match against W while sliding over it along j. Domain of $Cn(j,p,W)$ is [0, 1], where 1 represents the complete embedment[2] of p inside W.

The maximum score of the matching $Cn_{max}(p,W)$ represents similarity of the template in W, and offset index j at which

[2] *Path* is identical to a portion of *Whole*.

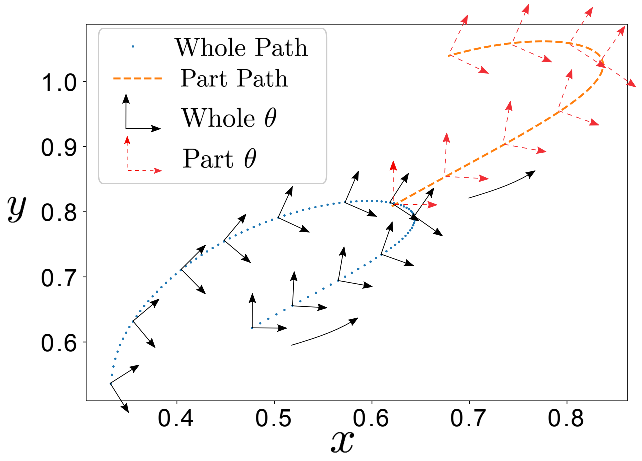

FIGURE 5: *Part* path is formed by trimming whole path followed by translation and scaling. Arrows indicate the increasing direction of parameter t.

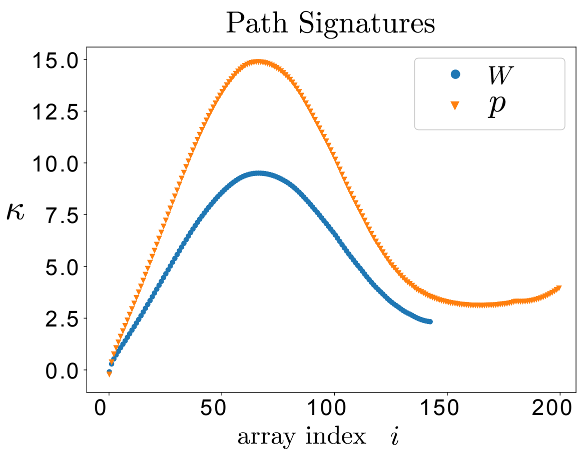

FIGURE 6: Path Signatures of part p and whole W from Fig. 5. The array index i corresponds to the index location for the array of the K. The domain of path signature is scale-invariant but the range still has a scaling factor, which is taken care of by normalized cross-correlation.

maximum occurs is the start point for matching. As we know that the signature reverses with reversal of the direction of t, we compute the correlation along both directions and select the best matching score, offset index and the matching direction of sampling. Figure 7 depicts normalized cross correlation function over the sliding domain j for *Part* and *Whole* curves.

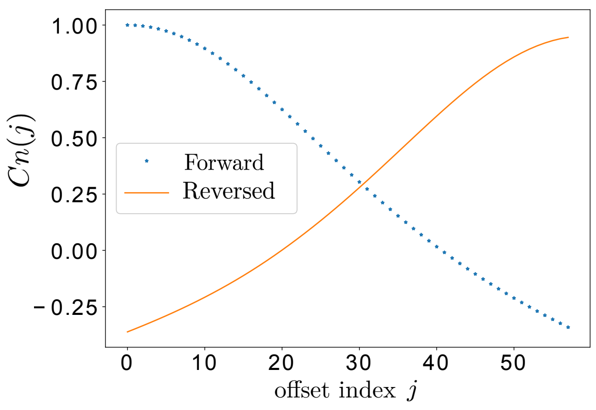

FIGURE 7: Normalized cross correlation of the signatures computed along each direction is shown. It can be seen that exact match is found at $j = 0$.

3.2 Partial Matching of Motion Signatures

This section presents how a template motion can be checked against other motion for potential matching. Consider *Part* and *Whole* motions shown in Fig. 5. Let p and W be the motion signatures of the *Part* and *Whole* motions, respectively as shown in Fig. 8. Similar to partial matching of path signatures, the cross-correlation function is given by,

$$E(j,p,W) = \sum_{i}^{p_{span}} \left((W(i+j) - \bar{W}(j:j+p_{span})) - (p(i) - \bar{p})\right)^2, \tag{4}$$

where $E(j,p,W)$ is the dissimilarity value when template p is matched to W at j^{th} index. Here p tries to find the best match against W while sliding over it. Similar to path signature, motion signature is dependent on the direction of t. Thus, we compute the dissimilarity for both directions and choose whichever is the least i.e. $E_{min}(p,W)$. Figure 9 depicts dissimilarity function over the sliding domain j. In this case, as shown in Fig. 9, we find that the first point is the matching point, which is consistent with the fact that we have essentially sliced the whole motion to obtain the part motion.

3.3 Objective Function for Synthesis

The functions in Eq. (3) and (4) presented in the sections 3.1, 3.2 can be used as the error measure for path and motion synthesis of any planar linkage, where the objective is to find a linkage that produces a motion whose part or whole corresponds to the target motion (or path). Thus we can formulate the path

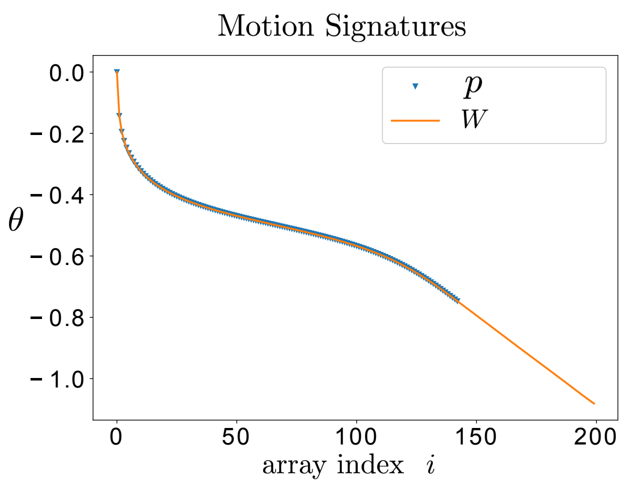

FIGURE 8: Motion Signatures of the trajectories shown in Fig. 5. The domain as well as range of motion signature is invariant to similarity transformation.

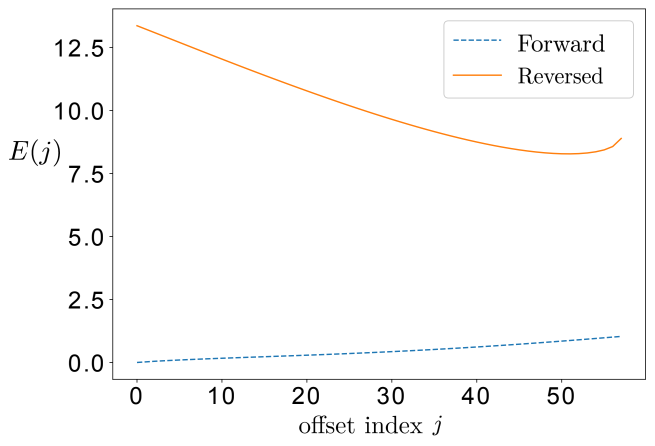

FIGURE 9: Dissimilarity function of two motion signatures along both directions. It can be seen that exact match is found at $j = 0$, where the template is fully embedded inside the other motion.

synthesis problem as,

$$\arg\min_{\boldsymbol{l},W_i}(1 - Cn_{max}(p, W_i)), \tag{5}$$

where $\boldsymbol{l}$ is the vector of linkage parameters for particular planar linkage, p is signature the of task path taken as the template and $\{W_i\}_{i=0}^{s}$ is the signature set of all s coupler paths generated by the linkage corresponding to $\boldsymbol{l}$. In case of four-bar, $\boldsymbol{l} : l_1, l_2, l_3, l_4, l_5$,

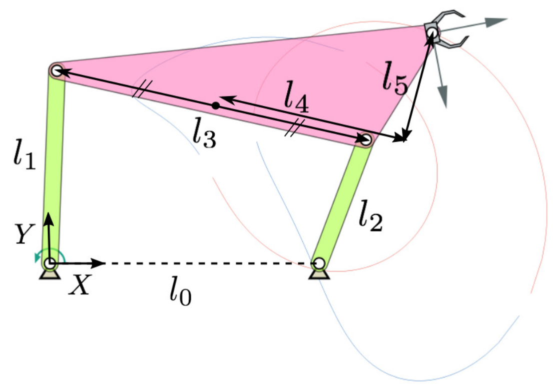

FIGURE 10: Parametric representation of four-bar linkage with all revolute joints. We set $l_0 = 1$ and one fixed joint at the origin of the global frame along with making the fixed length of four-bar parallel to the x-axis.

where l_i is link ratio of i^{th} link shown in Fig 10.

Similarly, we can formulate motion synthesis problem as,

$$\arg\min_{\boldsymbol{l},W_i}(E_{min}(p, W_i)). \tag{6}$$

Here, E is the Dissimilarity function from Eq. (4) while p and $\{W_i\}_{i=0}^{s}$ are motion signatures instead of path signatures.

Objective function evaluation step consists of calculation of coupler motion or path and finding its dissimilarity score. It is important to note that representation obtained in Section 2 reduces the size of the linkage parameters in optimization. This optimization problem can be solved using search methods which do not require gradient computation. We can employ global optimization methods such as differential evolution at the start, and local optimization methods link Powell's method towards the end for faster convergence [8].

Considering highly non-linear nature of the problem, finding a good initial guess proves to be daunting. Thus, we exploit machine learning techniques to create a database for finding many good initial guesses or the solution itself.

4 Sensitivity Analysis of Signatures

Due to the complex relationship between parameter space and generated motion, small changes in linkage parameters can produce large and discontinuous structural changes in the generated motions. For example, a small change in crank length (l_1) can open a previously closed coupler path. Most of the methods based on Fourier descriptors cannot capture the continuity at

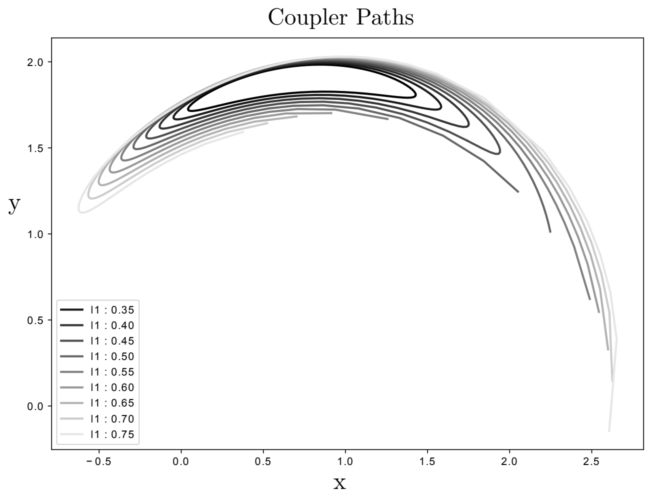

FIGURE 11: Coupler Motions of the four-bar linkage with variation of parameters l_1 and l_2. It can be seen that motion topology changes from close-loop Grashof to open loop Triple-Rocker.

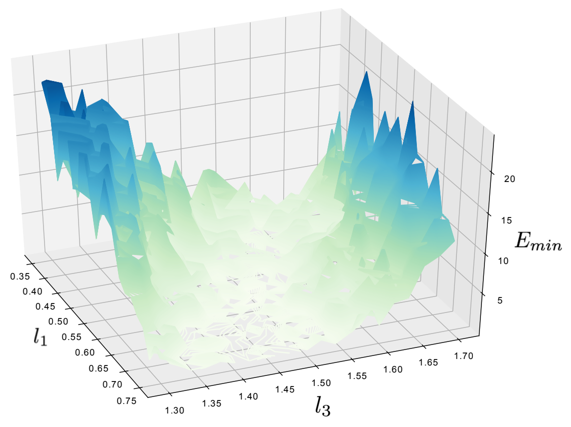

FIGURE 12: Distance (E_{min}) from Eq. (4) as the parameters l_1 and l_3 are varied. Although open loop breaks at l_1:0.55, l_3:1.5, there are no spikes of error function in the region near singularity, as the shape is very similar between the two topologies.

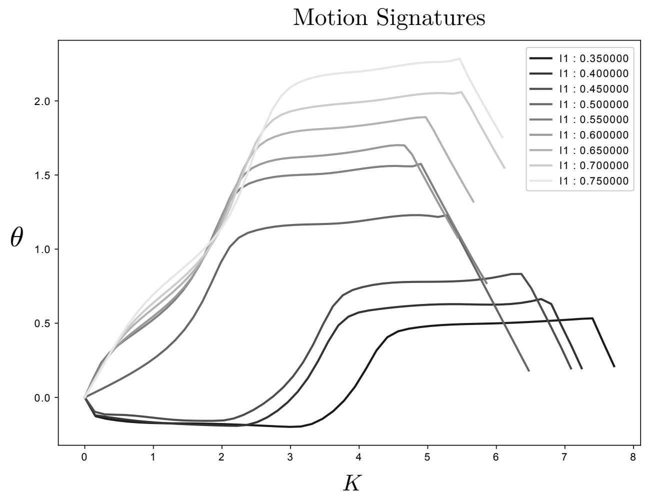

FIGURE 13: Motion signatures obtained by steps given in 2. Although topology difference is even more evident in this representation, it also signifies the similarity pattern between them.

such singular locations, which adversely affect the optimization process. In contrast to this behavior, the signatures derived in Sec. 2 have a smooth transition at these singular locations due to shape similarity between closed and just opened curve or motion.

To illustrate this, we perform sensitivity analysis in the vicinity of the singularity as follows: A four-bar with link ratios (l_1 : 0.55, l_2 : 1, l_3 : 1.5, l_4 : 1, l_5 : 1) is subjected to gradual change in parameters l_1 and l_3 by the amount (-0.2, 0.2) in steps of 0.01. Error function between motions of new and initial four-bar are calculated using Eq. (6) . Figure 11 shows coupler motion of some of the four-bars while Fig. 13 depicts their motion signatures. Please note that these two figures show the effect of changing only one parameter l_1. It can be seen from Fig. 11 that there exists a discontinuity in the topology of coupler curves even though their shapes have a continuous shift. Our method captures this continuity, which is shown by error function evaluations depicted in Fig. 12, where it is visible that surface is well behaved in the singularity region. This error function accounts for changes to both the parameters l_1 and l_3.

5 Clustered Database of Planar Linkages

Having an invariant representation facilitating partial matching greatly reduces data required to sample all possible types of shapes of coupler motion. We have built a database of planar four-bar linkages with revolute joints as an example, but the approach is same for any planar motion generating machine. We generate this database comprising of 40,000 linkages while taking following aspects into consideration.

1. Sampling should maximize the uniformity of its distribution over the space of four-bar coupler motions.
2. Data generation should be parallelized.
3. It should be scalable to higher-order linkages.

Figure 10 represents parametric representation of four-bar linkage with parameters (l_1, l_2, l_3, l_4, l_5). As mapping between four-bar linkage parameter space and coupler motion space is highly non-linear, uniform distribution over linkage parameter space does not necessarily mean uniform sampling over coupler motion space. Thus, an efficient approach would be to sample more in the regions where sensitivity is maximum. We have observed that whenever the link ratios of four-bar linkage are close to 1,

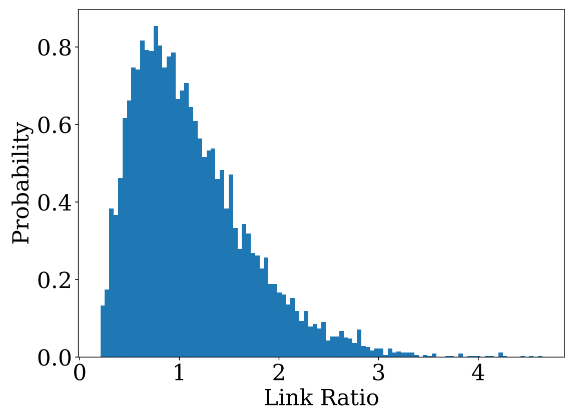

FIGURE 14: Probability Distribution function used in random sampling for parameters l_1, l_2 and l_3.

the sensitivity of shape of a coupler motion is higher than otherwise. Thus, we have chosen Log-Normal probability distribution ($\mu = 0, \sigma = 0.6$) for selecting the link ratios : (l_1, l_2, l_3) as shown in Fig. 14, and Normal Distribution ($\mu = 0, \sigma = 2$) for (l_4, l_5).

5.1 Dimensionality Reduction using Auto-Encoders

Each data point in the database consists of discrete signature, which is kept to be of 100 float digits. In order to have efficient query operations, we perform Hierarchical Clustering; a method that summarizes and creates a hierarchy in the database. Clustering in higher dimensions suffers from *Curse of Dimensionality* [17], thus we first perform dimensionality reduction using Auto-encoder Neural Networks. Auto-encoder is a powerful mapping model, which learns to encode the input data in very compact representation and can reconstruct the input with minimal error; performing much better than Principal Component Analysis [18]. This non-linear mapping by auto-encoder can greatly improve the representation of data for clustering [19]. Figure 15 shows a Neural Network architecture similar to the one we designed for the task. Our architecture consists of 100 neurons in input and output layer, while five hidden layers have (80, 50, 10, 50, 80) neurons respectively. Each neuron in the hidden layer is activated by Rectified Linear Unit ($ReLU$) activation function. In i^{th} hidden layer, $d^{(i-1)}$ dimensional vector output of the previous layer $h_(i-1)$ is fed as input to produce $d^{(i)}$ dimensional output h_i. Input-output relationship of a layer is given by,

$$h_i = ReLU(W_i h_{i-1} + b_i), \tag{7}$$

$$ReLU(x) = max(0, x), \tag{8}$$

where W_i is weight matrix with dimensions $(d^i, d^{(i-1)})$ and b_i is $d^{(i)}$ dimensional bias vector of i^{th} layer, which are computed in the process of training. Auto-encoders are trained to reconstruct the input, in this way each layer encodes the input, which is sufficient for the next layers to reconstruct the output. The objective of training is to find out the set of weights and biases that minimizes the error loss given by,

$$\underset{W,b}{\arg\min} \sum_{i=0}^{N} ||X_i - \tilde{X}_i||^2, \tag{9}$$

where X_i is input, $\tilde{X}_i$ is reconstructed output and N is number of training examples.

Once a network is trained, the output of the bottle-neck layer (h_{ib}) represents the compressed feature space (Z). As bottleneck layer has 10 neurons and input is a 100-dimensional vector, it is evident that information is compressed by a factor of 10, while achieving 95% reconstruction accuracy as the result of training. Standard clustering algorithms are performed on this latent[3] space for better clustering [19]. We use Agglomerative Clustering, a method of Hierarchical clustering, which is an approach to partitioning clustering for identifying groups in the dataset. *Ward* [20] criterion is used for clustering, which minimizes the variance of the clusters being merged. The distance metric used is the Euclidean distance in the latent space. Although the more accurate distance metric is distance function discussed in sec. 3, it is very expensive to calculate it for the entire database. Signatures with $O(m)$ points take $O(m \log m)$ time for each comparison and there are $O(N^2)$ number of comparisons to be made for the database of N points. When user raises one such query, we first find k nearest neighbors among 1500 cluster centers, then descend into their clusters if needed. Motion with highest similarity score is returned along with its corresponding linkage parameters, which if required, are further fine-tuned to match the query using local optimization methods. Computationally, on a 2.4 GHz Core i5 MacBook Pro with 8 GB memory, every query takes 23 seconds on average to find the sorted list of nearest neighbors among cluster centers.

6 Case Studies

This section presents two case studies presenting the effectiveness of our approach for path and motion synthesis applications.

6.1 Path Generation

In the design phase of a rehabilitation device that assists people to stand from sitting position, it is required to generate

[3]compressed output of bottleneck layer.

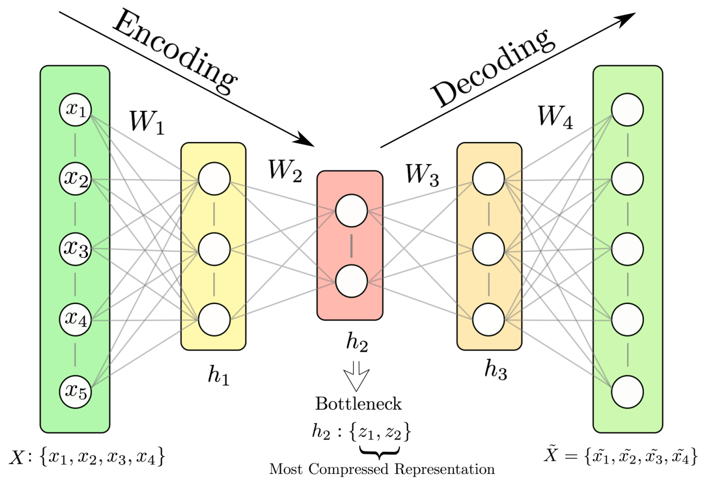
Encoding
Decoding
x_1
x_2
x_3
x_4
x_5
W_1
W_2
W_3
W_4
h_1
h_2
h_3
Bottleneck
$h_2 : \{z_1, z_2\}$
Most Compressed Representation
$X : \{x_1, x_2, x_3, x_4\}$
$\tilde{X} = \{\tilde{x_1}, \tilde{x_2}, \tilde{x_3}, \tilde{x_4}\}$

FIGURE 15: A small-scale version of the Auto-Encoder. This network takes 5-dimensional input in the input layer. At each encoder layer, the input is compressed into a vector of lesser dimensions, lowest at the bottleneck layer.

TABLE 1: Case Study 1 : Path Data

Point	x	y	Point	x	y
1	-7.81	-9.65	10	0.28	-1.31
2	-6.42	-9.81	11	0.64	1.07
3	-5.14	-9.62	12	0.98	2.73
4	-3.72	-8.99	13	1.47	4.30
5	-2.62	-8.14	14	2.73	6.58
6	-1.75	-7.13	15	3.46	7.41
7	-0.91	-5.67	16	4.07	7.95
8	-0.32	-4.10	17	4.70	8.41
9	-0.02	-2.92	18	5.32	8.76

TABLE 2: Linkage Parameters of Nine Nearest Neighbor Paths

linkage	l_1	l_2	l_3	l_4	l_5	$1 - Cn_{max}$
1	0.79	2.78	1.70	1.35	-0.71	0.0011
2	1.59	1.28	0.96	-1.74	-1.13	0.0015
3	0.99	0.71	1.66	-1.07	-0.85	0.0016
4	0.51	0.48	1.11	-0.02	-0.15	0.0017
5	0.93	0.75	2.18	-1.65	0.96	0.0018
6	1.29	1.98	1.02	-1.83	-1.33	0.0019
7	0.63	1.42	1.03	-1.90	-0.38	0.0020
8	0.66	0.85	0.84	-1.40	-0.13	0.0021
9	1.81	0.55	1.08	0.14	-0.04	0.0022

linkages that can execute a sit-to-stand (STS) trajectory of the hip joint as shown in Fig. 16. Table 1 presents the discretized path data. As our approach requires parametric representation of path, we first fit a cubic B-Spline with cord length parametrization through path data points to generate parametric curve shown in Fig. 16. We compute its path signature by the steps mentioned in Algorithm 2 and raise the query for nearest neighbors among 1500 cluster centers of our database. The distance metric for finding neighbors is $1 - Cn_{max}$ in Eq. (5). Table 2 tabulates the link ratios corresponding to obtained nine nearest neighbors. Next step is to compute actual parameters according to position, scale, and orientation of the path. It is done by comparing analogous points found by offset index j in Eq. 3. Figure 17 shows first eight four-bar mechanisms corresponding to nearest signatures to path signature of input. It can be clearly seen that these linkages generate highly accurate paths for the sit to stand activity. It is important to note that every solution is a result of partial matching of coupler paths, and otherwise would be very hard to search using other atlas-based approaches that only have the whole-to-whole matching facility.

6.2 Motion Generation

The task is to find a pool of linkage systems that can perform snow-shoveling with a motion shown in Fig. 18. The motion data is tabulated in Table 3 to which we fit a B-spline with cord length parametrization in order to get the parametric representation of motion. The task can also be treated as a finite position motion generation and solved for valid solutions. We try with our real-time computational methods of algebraic fitting [21], [22] but obtained solutions suffer from circuit defect, which is not surprising as those methods do not account for the continuity of input positions.

Now we employ the approach presented in this paper. The first step is to calculate motion signature of the task motion using steps mentioned in Algorithm 2. For that, we follow the steps given in section 2 to obtain the motion signature depicted in Fig. 19. Next, we raise the signature query for nearest signatures among cluster centers of the database. Fig. 19 shows nine nearest neighbor signatures along with the task signature. Table 4 presents the linkage parameters corresponding to the nearest neighbors along with their distance score from the task. Cou-

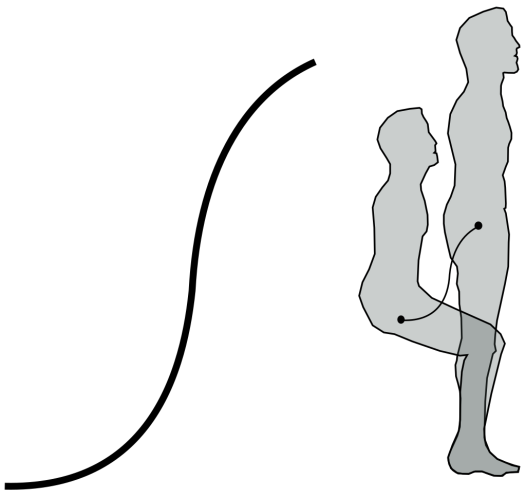

FIGURE 16: Case Study 1: Path traced by hip joint during Sit-to-Stand Motion.

TABLE 3: Case Study 2: Pose Data

Pose	x	y	θ	Pose	x	y	θ
1	0.03	0.07	6.06	6	1.39	0.47	0.73
2	0.38	0.01	6.18	7	1.36	0.77	1.03
3	0.71	-0.00	0.04	8	1.18	1.06	1.36
4	1.02	0.06	0.22	9	0.88	1.27	1.74
5	1.26	0.22	0.46				

pler motions of these linkages have a part, which matches with the shape of input motion query. Actual scaling and orientation of the linkage can be found out easily by comparing analogous points, which are given by the offset index j that corresponds to minimum distance(E_{min}) in Eq. (4). Figure 20 depict the solutions obtained after scaling and orienting the linkage to match required motion. All of these linkages satisfactorily perform the input task without any defect. As ground or fixed pivot locations should lie above the ground, all solutions except the 4th solution are suitable for the task. In light of these results, we can say that this approach produces a large variety of solutions, which otherwise would be very hard to find using precision point approach.

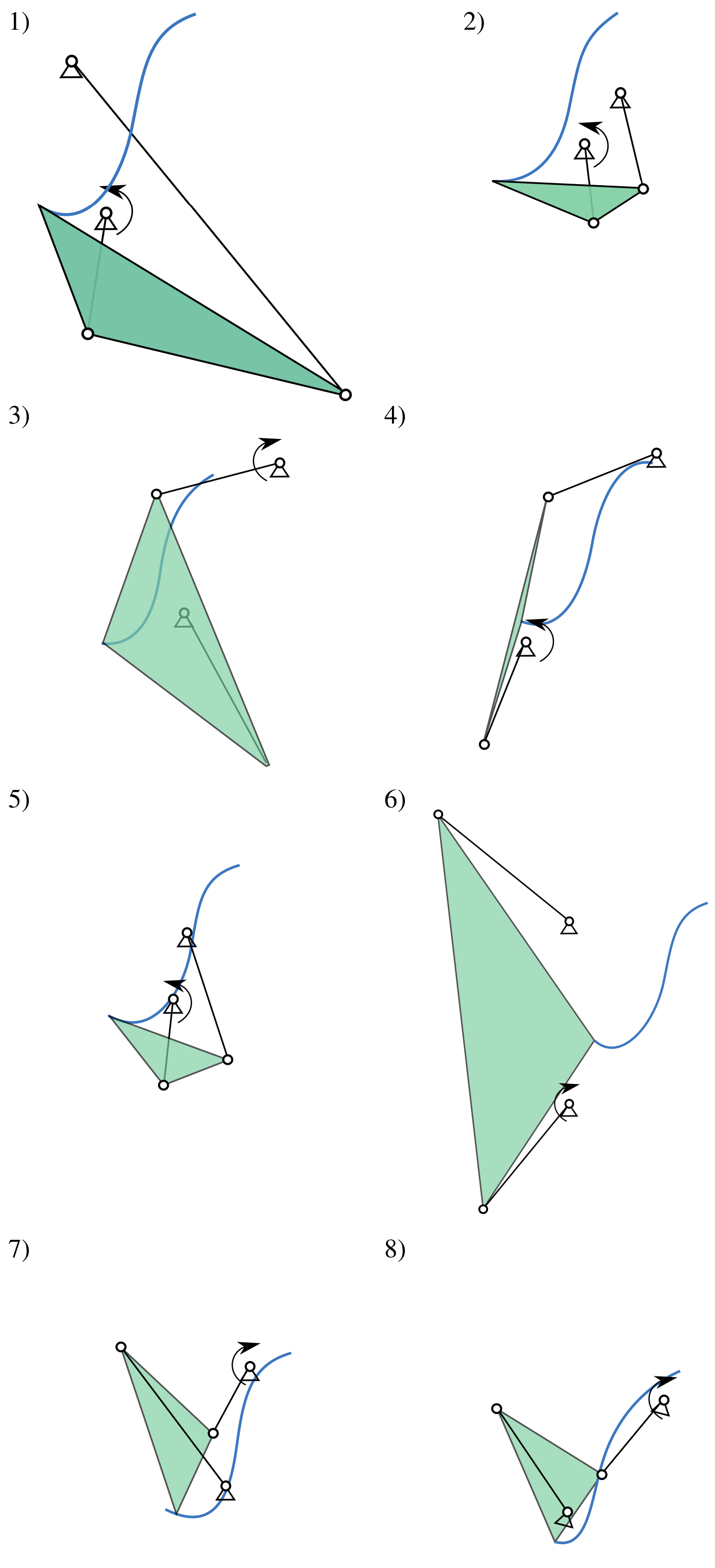

FIGURE 17: First eight linkages in the table 2 and their resultant coupler paths.

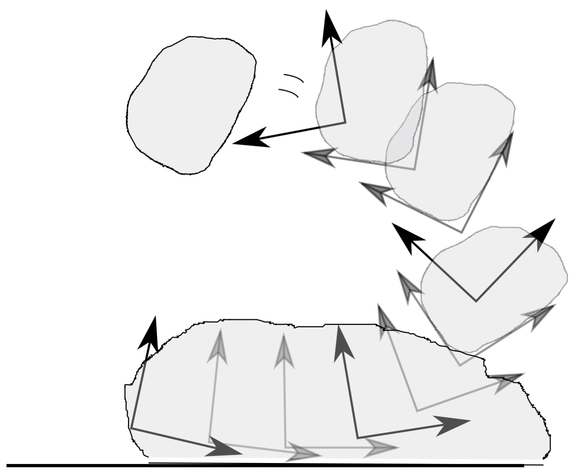

FIGURE 18: Case Study 2: User specified motion necessary for the snow shoveling task.

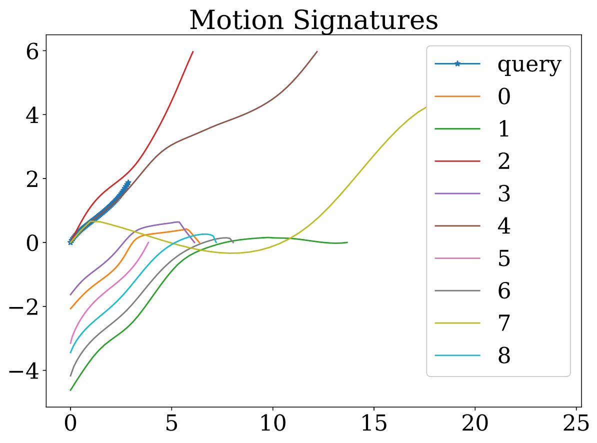

FIGURE 19: Case Study 2 - Query Result: Motion signatures in the dataset with highest similarity.

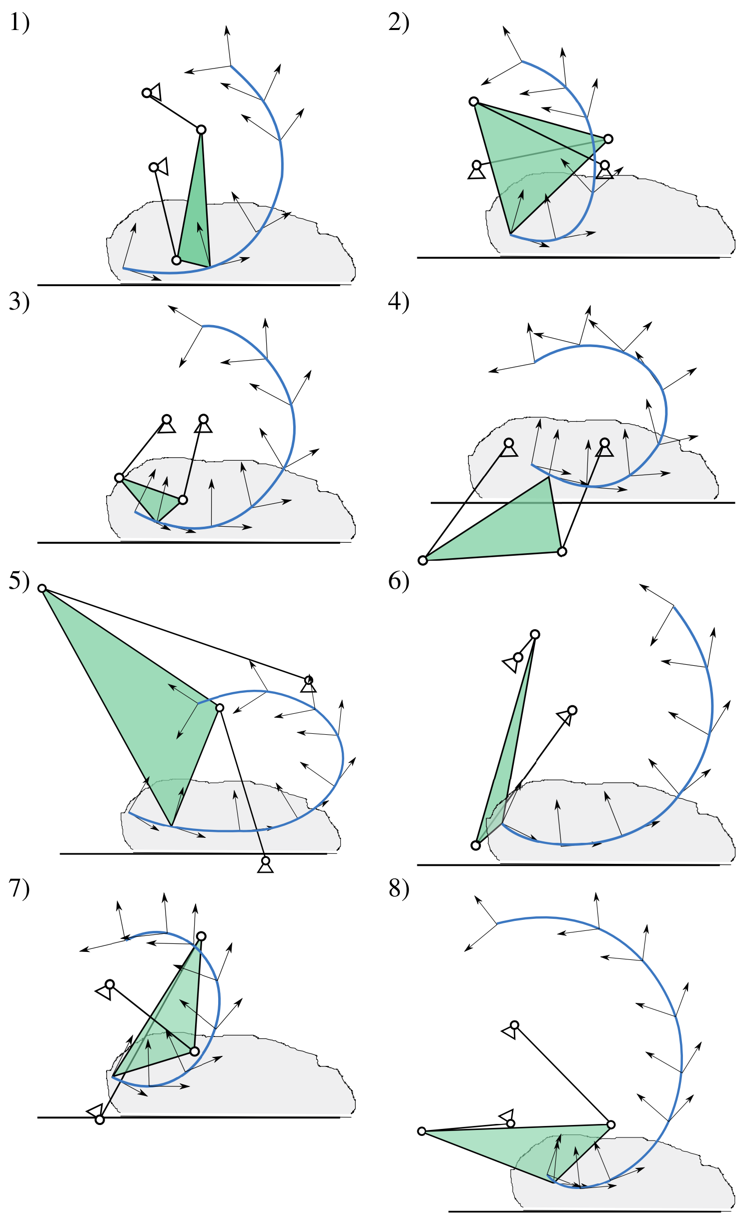

FIGURE 20: Case Study 2: First eight linkages in the table 4 and their resultant coupler motions.

Conclusion

This approach presents a novel machine-learning approach to generation of conceptual designs of defect-free linkage synthesis. The approach is highly data-efficient due to similarity invariant representation of paths and motion, which also facilitates partial matching. Sensitivity analysis indicates that complexity of the objective function, although multimodal and nonlinear, is not overly complex and certainly well behaved in the singular spaces. The hierarchically clustered database provides efficient query search. Finally, the effectiveness of the presented approach is showcased by two case studies for each of which nine defect-free solutions were found. Although our method can provide highly accurate results, our focus here is to generate a high number of defect-free solutions that can perform the task in more or less similar fashion; because more often than not, linkage synthesis algorithms are desired to be prolific in terms of concept generation than to produce highly accurate, but very few solutions. The approach is general enough to be extended to higher order linkage systems for which there are even fewer methods available for synthesizing defect-free solutions.

TABLE 4: Case Study 2: Linkage Parameters corresponding to Nine Nearest Neighbor Motions

linkage	l_1	l_2	l_3	l_4	l_5	E_{min}
1	1.28	0.88	1.77	-1.32	1.79	0.0186
2	1.05	1.14	1.09	0.35	-0.60	0.0362
3	2.06	2.28	1.84	-1.71	0.51	0.0378
4	1.52	1.22	1.46	0.05	0.12	0.0402
5	1.12	0.99	0.57	0.05	0.68	0.0467
6	1.58	0.88	1.17	0.08	-0.22	0.0481
7	2.17	0.38	2.87	-3.51	1.39	0.0578
8	1.55	0.79	0.85	-0.80	-0.52	0.0585
9	0.91	1.40	1.93	-0.93	-0.83	0.0605

ACKNOWLEDGMENT

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

REFERENCES

[1] McCarthy, J. M. and Soh, G. S., 2010, Geometric design of linkages, volume 11, Springer.

[2] Sandor, G. N. and Erdman, A. G., 1997, Advanced Mechanism Design: Analysis and Synthesis Vol. 2, Prentice-Hall, Englewood Cliffs, NJ.

[3] Hunt, K., 1978, Kinematic Geometry of Mechanisms, Clarendon Press, Oxford.

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

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

[6] Lohse, P., 2013, Getriebesynthese: Bewegungsablufe ebener Koppelmechanismen, Springer-Verlag.

[7] Chase, T. and Mirth, J., 1993, "Circuits and Branches of Single-Degree-of-Freedom Planar Linkages.", ASME Journal of Mechanical Design, **115(2)**, p. 223230.

[8] Ullah, I. and Kota, S., 1997, "Optimal synthesis of mechanisms for path generation using Fourier descriptors and global search methods", Journal of Mechanical Design, **119(4)**, pp. 504–510.

[9] Wu, J., Ge, Q., Gao, F., and Guo, W., 2011, "On the extension of a Fourier descriptor based method for planar four-bar linkage synthesis for generation of open and closed paths", Journal of Mechanisms and Robotics, **3(3)**, p. 031002.

[10] Li, X., Zhao, P., and Ge, Q., 2012, "A fourier descriptor based approach to design space decomposition for planar motion approximation", ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 699–708.

[11] McGarva, J., 1994, "Rapid search and selection of path generating mechanisms from a library", Mechanism and machine theory, **29(2)**, pp. 223–235.

[12] 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.

[13] Yue, C., Su, H.-J., and Ge, Q., 2011, "Path Generator via the Type-P Fourier Descriptor for Open Curves", Proceedings of 13th World Congress in Mechanism and Machine Science.

[14] Chu, J. and Sun, J., 2010, "A new approach to dimension synthesis of spatial four-bar linkage through numerical atlas method", Journal of Mechanisms and Robotics, **2(4)**, p. 041004.

[15] Cui, M., Femiani, J., Hu, J., Wonka, P., and Razdan, A., 2009, "Curve matching for open 2D curves", Pattern Recognition Letters, **30(1)**, pp. 1–10.

[16] Lewis, J. P., 1995, "Fast normalized cross-correlation", Vision interface, volume 10, pp. 120–123.

[17] Marimont, R. and Shapiro, M., 1979, "Nearest Neighbour Searches and the Curse of Dimensionality", IMA Journal of Applied Mathematics, **24(1)**, pp. 59–70, doi: 10.1093/imamat/24.1.59.

[18] Hinton, G. E. and Salakhutdinov, R. R., 2006, "Reducing the dimensionality of data with neural networks", science, **313(5786)**, pp. 504–507.

[19] Song, C., Liu, F., Huang, Y., Wang, L., and Tan, T., 2013, "Auto-encoder based data clustering", Iberoamerican Congress on Pattern Recognition, Springer, pp. 117–124.

[20] Ward Jr, J. H., 1963, "Hierarchical grouping to optimize an objective function", Journal of the American statistical association, **58(301)**, pp. 236–244.

[21] 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 Journal of Computing and Information Science in Engineering, 10.1115/1.4035528.

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