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-TBD

MACHINE LEARNING-DRIVEN KINEMATIC SYNTHESIS OF DEFECT-FREE PLANAR FOUR-BAR LINKAGES FOR MOTION AND PATH GENERATION USING SHAPE OPTIMIZATION

Shrinath Deshpande, Anurag Purwar*

Computer-Aided Design and Innovation Lab
Department of Mechanical Engineering
Stony Brook University
Stony Brook, New York, 11794-2300

ABSTRACT

Success in Synthesis of mechanisms for motion generation refers to computing linkage type and dimensions to perform the task without disassembly. Majority of the methods formulate it as Precision Position problem (or Classical Burmester Problem) which inherently ignores the continuity information, resulting into solution linkages with branch, circuit and order defects. In this paper, we present a novel approach that addresses this issue into problem formulation and returns the solutions free from defects. At the heart of this approach lies an objective function that compares the motion as a whole, capturing designer's intent. In contrast to widely used structural error or loop-closer 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 invariant to similarity transformations. Actual scale, position and orientation of linkage components are obtained by subsequent trivial fitting.

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 k 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.

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. These problems have been dealt with by many approaches with the aim of finding acceptable solutions for practical situations. Several text books, such McCarthy and Soh [1], Sandor and Erdman [2], Hunt [3], Hartenberg and Denavit [4], Suh and Radcliffe [5], and Lohse [6] cover the science and art of planar four-bar and higher-order linkages. The majority of these theories don't account for circuit and branch defect in the synthesis process, which can render the solution useless for practical applications. These defects in single DOF mechanisms are thoroughly discussed by Chase and Mirth [7].

Original contribution of this paper are in 1) a novel objective function for defect-free motion generation problem, 2) discovering a representation of coupler motion and path which is invariant under similarity transformations (henceforth termed as motion signature and path signature respectively), 3) novel algorithm for partial matching of motions , 4) optimal storage data by means of auto-encoder neural network and efficient neighbor search in the clustered database. 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; ensuring

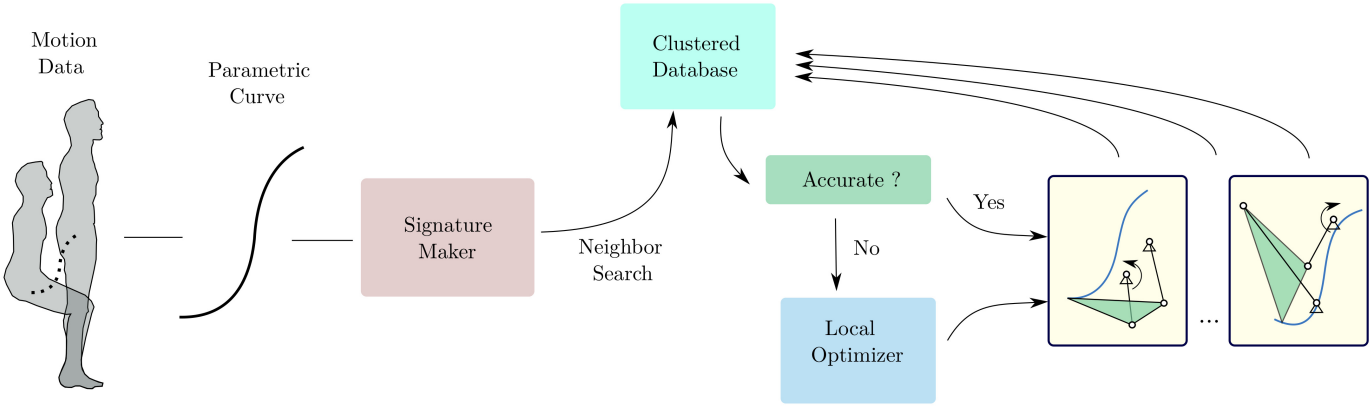

FIGURE 1: General Layout of the approach.

the output mechanism is defect free throughout the task.

Path synthesis methods based on Fourier analysis do take the continuity information of the coupler path into account, however majority of them are only defined on closed loop curves. Ullah and Kota [8] have presented an invariant approach towards representation and synthesis of closed paths through shape optimization. They use combination of global and local search methods by optimizing Fourier Deviant function to compute the dimensions of planar fourbar linkage without any initial guesses. Wu et al. [9] presented a synthesis method for path generation based on finite Fourier series for synthesis of open and closed paths. In the case of motion generation, methods that take the branching defect into consideration, limit their search in the configuration space where Grashof condition holds true. To our best knowledge, no other motion generation method truly takes the motion continuity into account.

Owing to the highly nonlinear nature of the problem, most optimization based methods require good initial condition to start. Thus, an atlas or a design library is used to provide good initial conditions. McGarva [10] has taken the earliest approach towards creating a library for coupler trajectories based on harmonic analysis. Wandling [11] 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. [12] 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 nearest neighbor search. The best match is returned as solution to input. Chu et al. [13] have showcased an atlas based method of synthesizing spatial fourbar linkage for function generation problem where orientation data is stored in terms of Fourier harmonics.

The above methods generate data based on uniform sampling in the linkage parameter space. Given the highly nonlinear mapping between linkage parameters and coupler trajectory, this way of sampling leads into non-uniform sampling of trajectory space causing under-representation of possible motions. We try to 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. A Hierarchy is created in the database by means of clustering, where the top level comprises of few data points called cluster centers which are representative of the cluster points in the lower levels. Also the methods [11], [12] have built the library 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 it as our representation facilitates partial matching. No other previous method facilitates this partial matching of open motion curve into another open or closed curve. Thus reducing the data requirement even further.

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. The neighbor is defined as motion or path whose part has a similar signature to the input. These k neighbors if needed, are subjected to fine tuning by local optimization to obtain set of solutions. Overall method depicted in Fig. 1 is codified in Algorithm 1.

Organization of the paper is as follows. Section 2 presents the computation of motion and path signatures. Section 3 is comprised of evaluation criterion for two signatures based on the shape similarity, which leads to form the error function for optimization. Section 4 discusses the nature of objective function via sensitivity analysis at a singularity. Section 5 explains the

[1] path or motion signature

database generation and clustering using auto-encoders for efficient sampling and query operations. Finally, two case studies are presented 6 to showcase 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 $\boldsymbol{l}$: l_1, l_2, ...

```
signature = calculateSignature(Input);
distances = [];
for centerPoint in clusterCenters do
    distances.push(getDistance(signature, centerPoint))
end
kNeighbors = getNeighbors(distances, k) for neighbor in kNeighbors do
    if threshold < neighbor.distance then
        return neighbor.LinkParameters
    else
        return Optimize(neighbor.LinkParameters)
    end
end
```

2 Signatures of Coupler Path and Motion

The main focus of the paper is a novel method for motion generation that takes a continuous task motion or path as the input and returns defect-free linkages producing similar motion or path. The objective function evaluation requires the input motion to be transformed into a specific form, which we call the signature of input motion. This signature representation is invariant to similarity operations, which are reflection, rotation, translation and scaling. Signature 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 [14]. For this task, our approach requires the input in terms of a parametric motion $(x : x(t), y : y(t), \theta : \theta(t))$. This section explains the procedure of signature calculation given in Algorithm 2.

Curvature $k(t)$ of the path of a parametric motion $(x : x(t), y : y(t), \theta : \theta(t))$ and its integral $K(t)$ is given by,

$$k(t) = \frac{\ddot{y}(t)\dot{x}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{(\frac{3}{2})}}, \tag{1}$$

$$K(t) = \int_0^t |k(t)| dt, \tag{2}$$

where $\dot{x}$, $\ddot{x}(t)$ are first and second order derivative w.r.t. parameter t. For an example, parametric motion could be a B-spline

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

```
k(t) = ComputeCurvature(x, y) using Eq. 1
K(t) = IntergrateCumulatively(k(t)) using Eq. 2
motionSignature = []
pathSignature = []
for i = 0 → max(K) do
    tmp = (value of t corresponding to which K has value i)
    i = i + 0.1
    motionSignature.push(k(tmp))
    pathSignature.push(k(tmp))
end
return PathSignature, MotionSignature
```

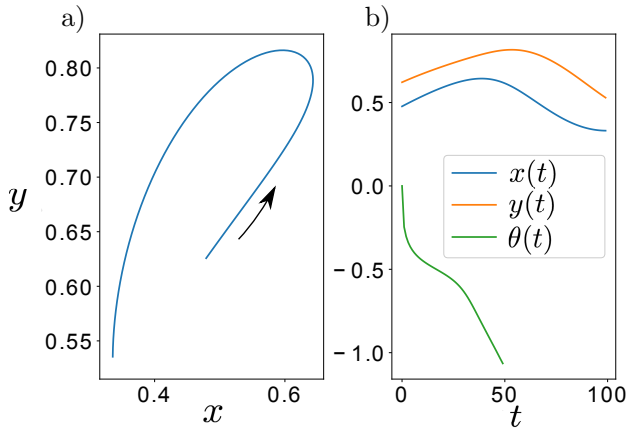

FIGURE 2: a) The path of the input motion along with direction of parameterization. b) Motion components $x(t), y(t), \theta(t)$ are plotted against parameter t.

motion as shown in fig 2. We compute the curvature of path along the direction of t, using Eq. 1 and the cumulative integral by Eq. 2. For implementation purposes, we use scipy [15] module for computations of B-splines and their derivatives. Figure 3 shows computed curvatures and its unsigned integral of coupler path shown in Fig. 2. It can be seen that curvature is very low 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 parameterization reverses.

Now we re-sample the curvature at equal intervals of $K(t)$, which is equivalent to plotting k vs K in Fig 4(a). This is done by finding the parameter values of t where K changes in the steps of some step size. For practical purposes we find array of parameter t such that K increments by 0.1. For each value of t in such array,

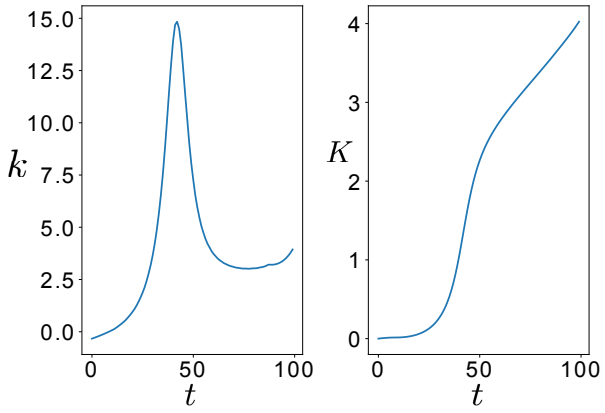

FIGURE 3: Curvature and its unsigned integral for the path shown in Fig. 2

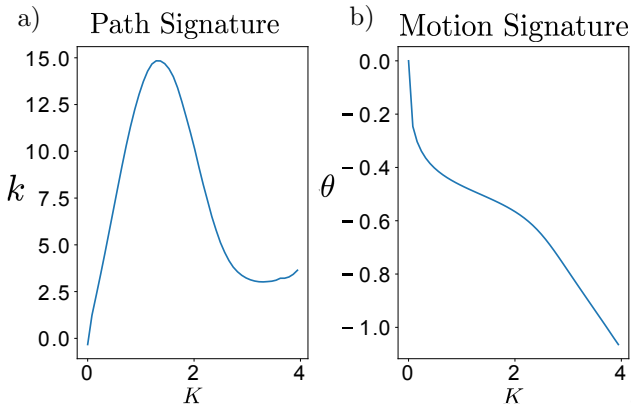

FIGURE 4: Path and motion signatures of the motion shown in Fig. 2

we compute $k(t)$ and $\theta(t)$ and store it as the path and motion signatures respectively.

These signatures are invariant under similarity transformations; for proof see [14]. The general intuition is as follows: we know that curvature changes inversely to the scale of the curve, so when it is integrated along the scaled curve, the scale factor cancels itself out. Reflection operation on the motion or the path, 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 signature depends on the direction along which we compute the curvature $k(t)$ as well as re-sample $\theta(t)$.

3 Signature Matching and Error Function

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

3.1 Partial Matching of Path Signatures

When a path query is raised, it can be very useful to know whether this path already exists as a part in some other path present in the database. This subsection presents a method for determining this partial similarity.

Lets consider two coupler paths; namely part and whole as shown in 5. Let p be path signature of the part path which acts as template, while W is path signature of the whole path that can completely embed the template 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 signature. The way this partial matching works is as follows:

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

$$Cn(j,p,W) = |\sum_{i}^{p_{span}} \frac{(W(i+j)-\bar{W}(j\rightarrow j+p_{span}))(p(i)-\bar{p})}{\sqrt{\sum_{i}^{p_{span}}(W(i+j)-\bar{W}_{p_{span}})^2 \sum_{i}^{p_{span}}(p(i)-\bar{p})^2}}|, \quad (3)$$

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

Here p acts as template that tries to find 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.

Maximum score of the matching ($Cn_{max}(p,W)$) represents similarity of the template in W, and offset index j at which maximum occurs is the start point for matching. As the signature reverses with reversal of the direction of sampling, we compute the correlation first in one direction and again after reversing p 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.

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

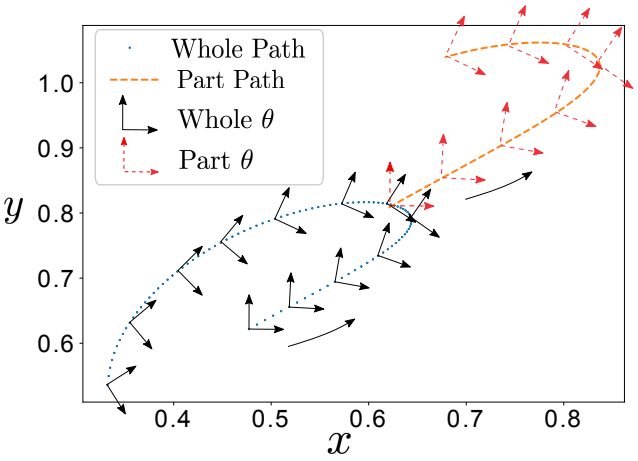

FIGURE 5: Part path is formed by trimming whole path followed by translation and scaling.

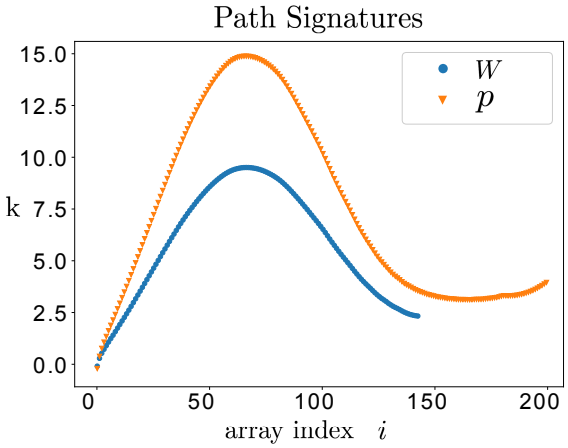

FIGURE 6: Path Signatures of part p and whole W from Fig. 5.The domain of path signature is scale invariant but the range still has a scaling factor, which is taken care by normalized cross-correlation.

signatures of part and whole motions respectively. All the steps are same as done for partial matching of path signature except the cross-correlation function which is given by,

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

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 best match against $W(i)$ while sliding over it. Similar to path signature, motion signature is dependent on the sampling direction along the motion. Thus we compute the dissimilarity for both direction and the take

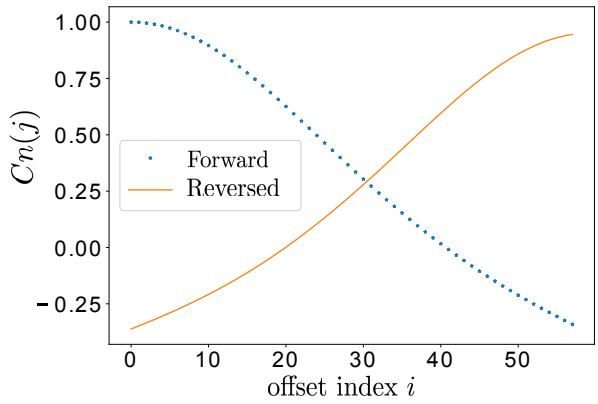

FIGURE 7: Normalized cross correlation of the signatures computed along each direction is shown. It can be seen that exact match is found at $j = 0$.

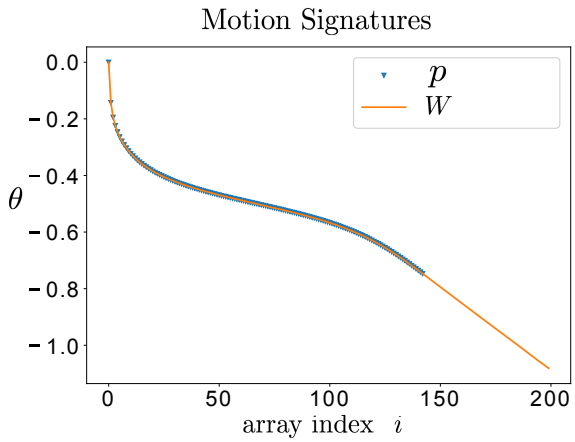

FIGURE 8: Motion Signatures of the trajectories shown in Fig.5. The domain as well as range of motion signature is invariant to similarity transformation.

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 itself is a point where the perfect match is occurred, which is consistent with the fact that we slice 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 target motion (or path). Thus we can formulate the path syn-

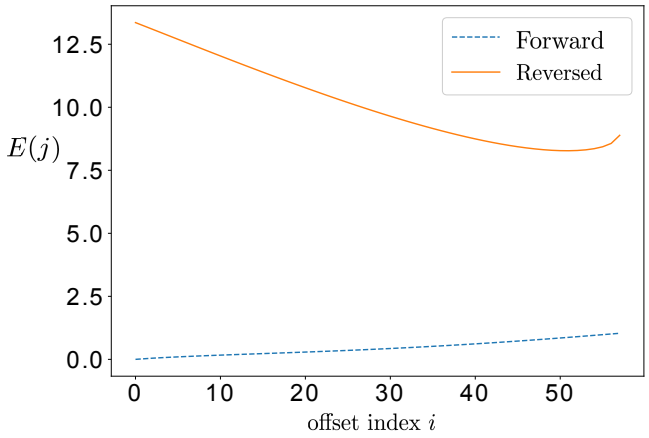

FIGURE 9: Dissimilarity function of two motion signatures along both directions. It can be seen that exact match is found at $j = 0$, where template is fully embedded inside the other motion.

thesis 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 curve taken as template for Normalized Cross-Correlation 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, $\boldsymbol{l} : l_1, l_2, l_3, l_4, l_5$, with 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 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 game changing. Thus we exploit machine learning techniques to create a database for finding many good initial guesses or the solution itself.

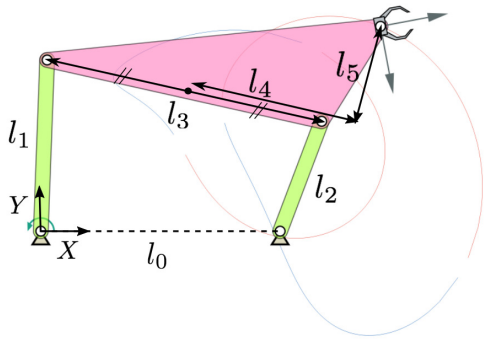

FIGURE 10: Parametric representation of four-bar linkage with all revolute joints. We set $l_0 = 1$ and one fixed joint at origin of the global frame along with making fixed length of the fourbar parallel to x axis.

4 Sensitivity Analysis of Signatures

Owing to the complex relationship between parameter space and generated motion, small change in linkage parameters can produce large and discontinuous structural change in the generated motions. For example, small change in crank length (l_1) can open a previously close coupler path. Most of the methods based on Fourier descriptors are if defined, can not capture the continuity at such singular locations, which adversely affect the optimization process. In contrast to this behavior, the signatures derived in Sec. 2 have smooth transition at these singular locations due to shape similarity between a closed and just opened curve or motion.

To make our point, we perform sensitivity analysis 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 fourbar are calculated using Eq. (6) . Figure 11 shows coupler motion of some of the fourbars while Fig. 13 depicts their motion signatures. It can be seen from Fig. 11 that there exists a discontinuity in topology of the 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.

5 Clustered Database of Planar Linkages

Having a invariant representation facilitating partial matching greatly reduces data required to sample all possible types 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.

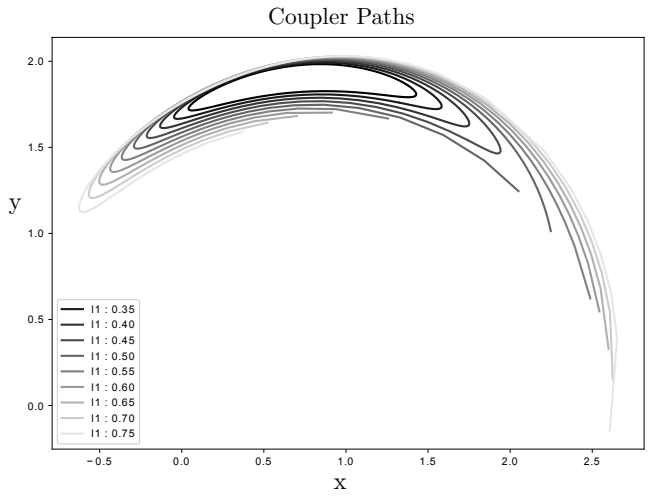

FIGURE 11: Coupler Motions of the fourbar linkage with variation of parameters l_1 and l_2. It can be seen that motion topology changes from close-loop Grashof to open loop Triple-Rocker.

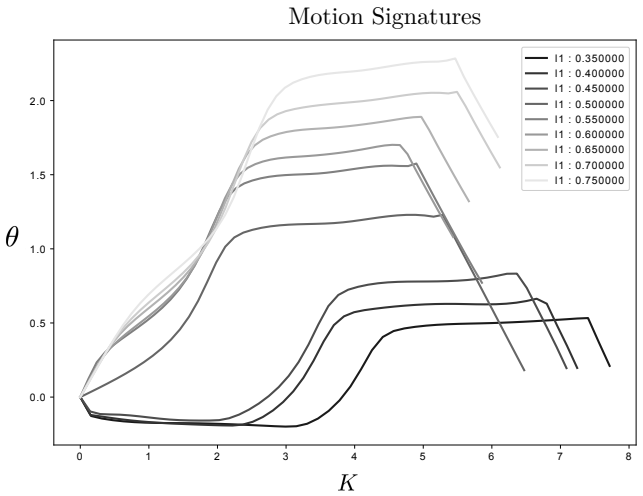

FIGURE 13: Motion signatures obtained by steps given in 2. Although topology difference is even more evident in this representation, it also signifies the similarity pattern between them.

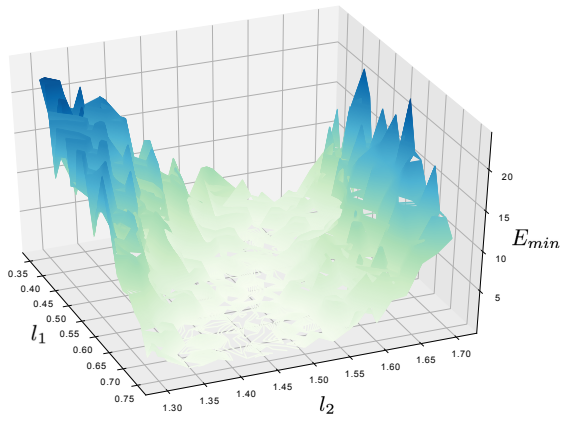

FIGURE 12: Distance (E_{min}) from Eq. (4) as the parameters l_1 and l_2 are varied. Although open loop breaks at l_1:0.55, l_3:1.5, there are no spikes of error function in the region near singularity, as the shape is very similar between the two topologies.

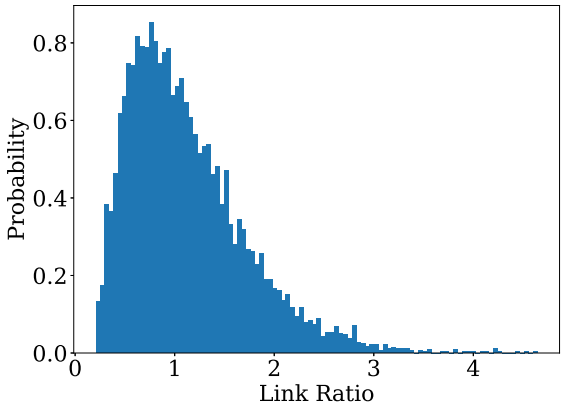

FIGURE 14: Probability Distribution function used in random sampling for parameters l_1, l_2 and l_3.

1. Sampling should maximize the uniformity of its distribution over the space of four-bar coupler motions.
2. Data generation can be parallelized.
3. Scalable with linkages with more number of links.

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 doesn't necessarily mean uniform sampling over motion space. Thus the efficient approach would be to sample more in the regions where sensitivity is maximum. Our observational intuition tells that whenever the link ratios of four-bar linkage are close to 1, the sensitivity of the shape of a coupler motion is higher than otherwise. Thus we have chosen Log-Normal probalitiy 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 database consists of discrete signature of motion, 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 Dimen-*

sionality [16], 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 [17]. This Non-linear mapping by auto-encoder can greatly improve the representation of data for clustering (For details refer [18]). 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. Objective of the training is to find out set of weights and biases that minimizes the error loss given by,

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

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

Once a network is trained, Output of the bottle-neck layer (h_{ib}) represents the compressed feature space (Z). As the bottleneck layer has 10 neurons and input is 100 dimensional vector, it is evident that information is compressed by a factor of 10, while achieving 95% reconstruction accuracy as a result of training. Standard clustering algorithms are performed on this latent[2] space for better clustering [18]. We use Agglomerative Clustering, a method of Hierarchical clustering, which is an approach to partitioning clustering for identifying groups in the dataset. *Ward* [19] 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,

[2]compressed output of bottleneck layer.

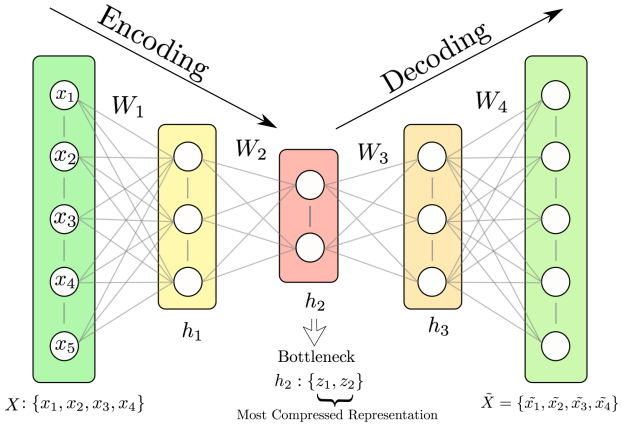

FIGURE 15: A small scale version of the Auto-Encoder. This network takes 5 dimensional input in the input layer. At each encoder layer, input is compressed into vector of lesser dimensions, lowest at the bottleneck layer.

it is very expensive to calculate it for entire database. Signatures with $O(m)$ points, take $O(m \log m)$ time for each comparisons and there are $O(N^2)$ number of comparisons to be made for 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.

6 Case Studies

This sections 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 stand from sitting position, it is required to generate linkages that can execute a sit-to-stand (STS) trajectory of hip 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 parameterization 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

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

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 fourbar mechanisms corresponding to nearest signatures to path signature of input. As it can be clearly seen, these linkages generate highly accurate paths without any defects throughout sit to stand activity.

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 parameterization in order to get 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 [20], [21] but obtained solutions suffer from circuit defect, which is not surprising as those methods do not account for continuity of the input positions.

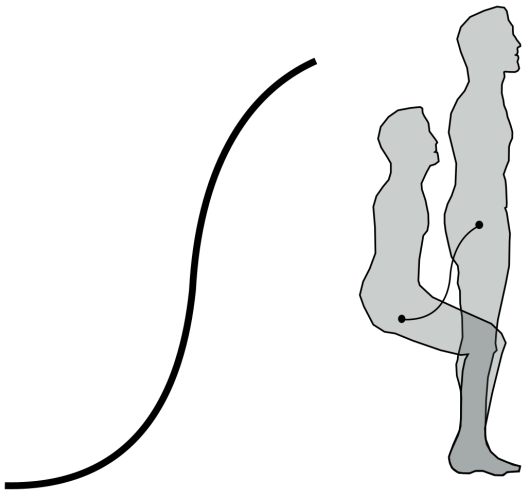

FIGURE 16: Case Study 1: Path traced by hip joint during Sit-to-Stand Motion.

Now we employ the approach presented in this paper. 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 the 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. Coupler 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 4^{th} 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.

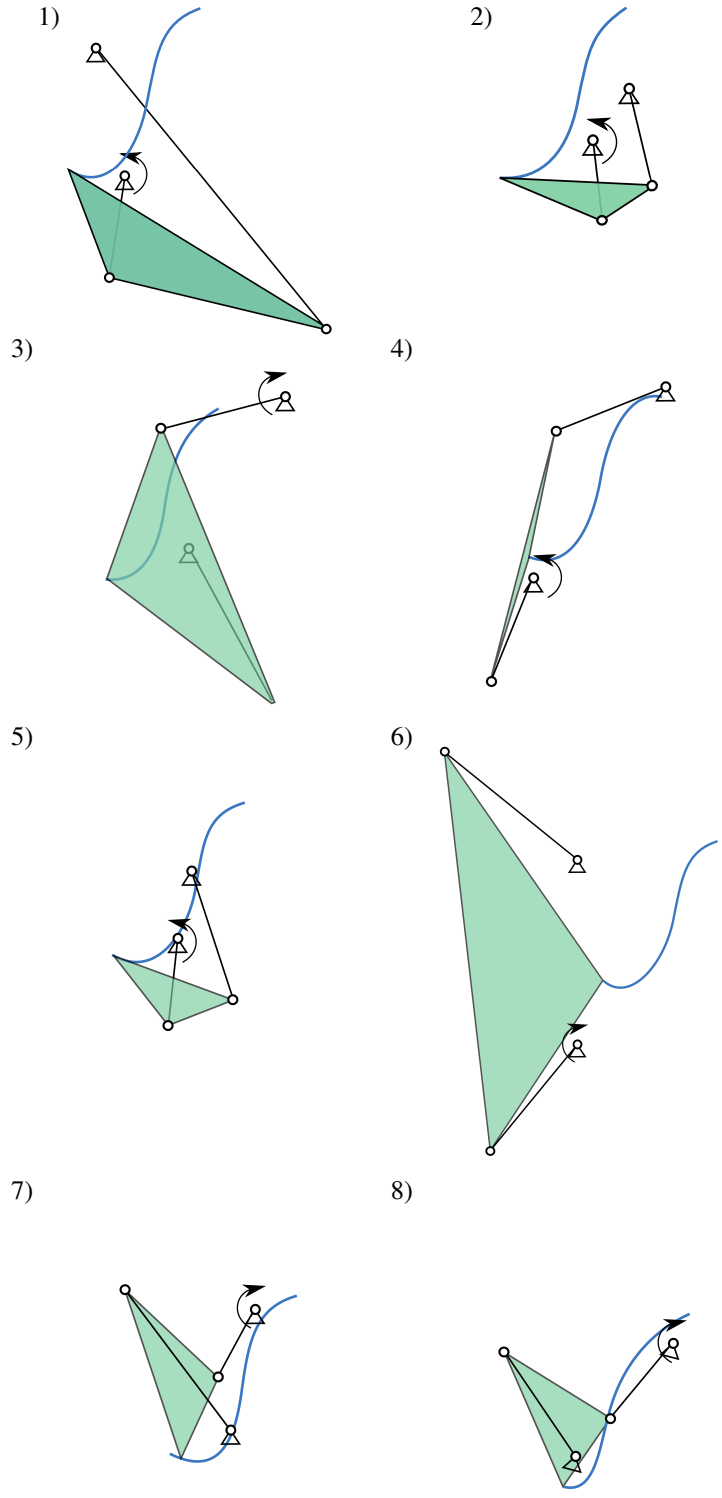

FIGURE 17: First 8 Linkages in the table 2 and their resultant coupler paths.

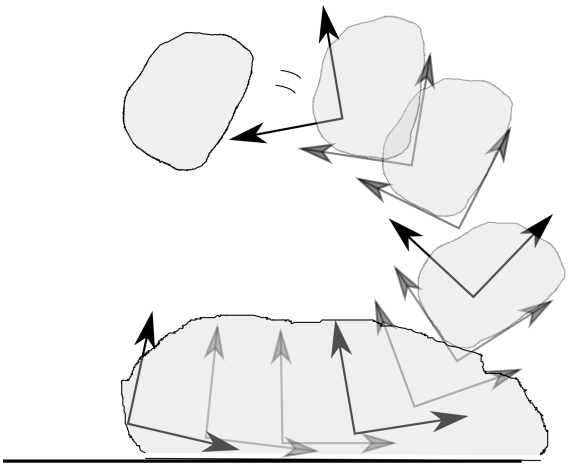

FIGURE 18: Case Study 2: User specified motion necessary for the snow shoveling task.

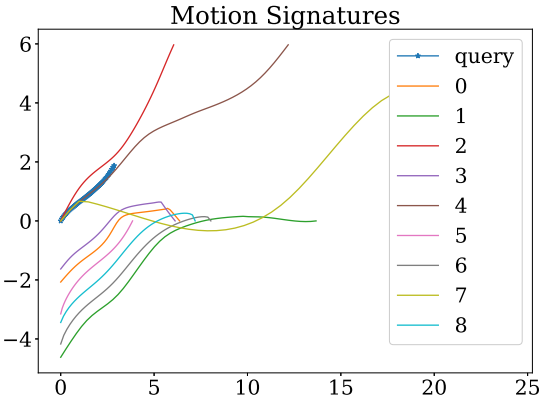

FIGURE 19: Case Study 2 - Query Result: Motion signatures in the dataset with highest similarity.

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				

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

Conclusion

This approach presents a novel optimization approach towards defect free linkage synthesis. The framework is highly data efficient due to similarity invariant representation of paths and motion, which also facilitates partial matching. The method always returns defect free solutions as it is inherently taken care of. Sensitivity analysis indicates that complexity of the objective function, although obviously multimodal and nonlinear, is not overly complex and certainly well behaved in the singular spaces. Hierarchically clustered database provides efficient query search. Finally, the effectiveness of 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 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 very few solutions.

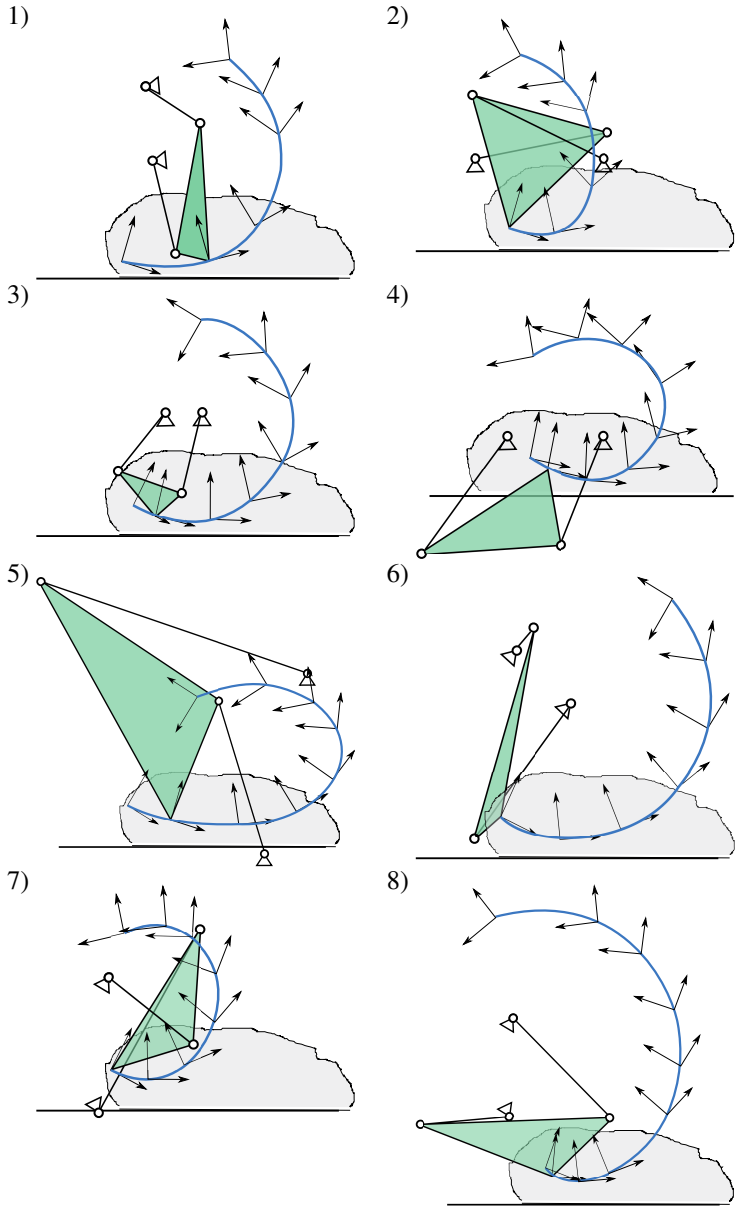

FIGURE 20: Case Study 2: First 8 Linkages in the table 4 and their resultant coupler motions.

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] McGARVA, J. R., 1994, “Rapid search and selection of path generating mechanisms from a library”, Mechanism and machine theory, **29(2)**, pp. 223–235.
[11] Wandling Sr, G. R., 2000, “Synthesis of mechanisms for function, path, and motion generation using invariant characterization, storage and search methods”, .
[12] 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.
[13] 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.
[14] 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.
[15] Jones, E., Oliphant, T., Peterson, P., et al., 2001–, “SciPy: Open source scientific tools for Python”, URL http://www.scipy.org/, [Online; accessed ¡today¿].
[16] MARIMONT, R. B. and SHAPIRO, M. B., 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, URL +http://dx.doi.org/10.1093/imamat/24.1.59.
[17] Hinton, G. E. and Salakhutdinov, R. R., 2006, “Reducing the dimensionality of data with neural networks”, science, **313(5786)**, pp. 504–507.
[18] 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.
[19] Ward Jr, J. H., 1963, “Hierarchical grouping to optimize an objective function”, Journal of the American statistical association, **58(301)**, pp. 236–244.
[20] 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.
[21] 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.

List of Tables

List of Figures