Data-Driven Approach To Variational Synthesis of Mechanisms

A Dissertation Presented

by

Shrinath Deshpande

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the degree of

Doctor of Philosophy

in

Department of Mechanical Engineering

Stony Brook University

May 2020

Stony brook University

The Graduate School

Shrinath Deshpande

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Dr. Anurag Purwar - Dissertation Advisor
Professor, Mechanical Engineering

Dr. Jeff Ge - Chairperson of Defense
Professor, Mechanical Engineering

Dr. Imin Kao
Professor, Mechanical Engineering

Dr. Nilanjan Chakraborty
Professor, Mechanical Engineering

Dr. Niranjan Balasubramanian
Professor, Computer Science

This dissertation is accepted by the Graduate School

Charles Taber

Dean of the Graduate School

Abstract of the Dissertation

Data-Driven Approach To Variational Synthesis of Mechanisms

by

Shrinath Deshpande

Doctor of Philosophy

in

Department of Mechanical Engineering

Stony Brook University

2020

Kinematic synthesis of mechanisms is an important phase in the conceptual design of machines. This requires the synthesis to be prolific in terms of concept generation to realize the potential of attainable design possibilities and to have the agility to adapt a design to evolving requirements. The goal of this research is to advance the science of kinematic synthesis by bringing together machine learning and theoretical kinematics to create a data-driven computational framework for kinematic synthesis of mechanisms.

At the heart of this novel framework lies a probabilistic generative model that learns joint probability distribution of various linkage parameters and their interdependence to perform useful inference tasks. In doing so, we leverage the emerging machine learning techniques to learn meaningful representations and combine them with simultaneous type and dimensional methods of kinematic synthesis to enhance users' computational creativity. The approach is particularly amenable to mechanism synthesis when the input from mechanism designers is deliberately imprecise or inherently uncertain due to the nature of the problem. The approach models the input as a probability distribution and employs a deep generative model to capture the inherent uncertainty in the input. In addition, it gives feedback on the input quality and pro-

vides corrections for a more conducive input. Lastly, the outputs are post-processed and the designer is presented with a set of distributions of solutions, where each set consists of a concept with different variations. We define this approach, where the input uncertainty is intelligently managed to generate a distribution of solutions as Variational Synthesis of Mechanisms.

Dedication

I dedicate this dissertation to the ladies of my life. First, I dedicate this to my beloved wife Prerna, who made great sacrifices so that we could be together during this challenging journey. Second, I dedicate this thesis to my grandma, who practically raised me. Third, I dedicate this to my mother, who has given her all for the family.

Acknowledgements

Undertaking this Ph.D. has been a life-changing experience for me and it would not have been possible to do without the support and guidance that I received from many people.

Working under the guidance of Prof. Anurag Purwar has been an amazing experience. I was fortunate to have him as my advisor. His approach to problem-solving and attention to detail have an everlasting impact on me and my work. It is due to his trust and constant support that I was able to research an area new to our research community. His patience and enthusiasm helped me in a great deal throughout my research. I express my sincere gratitude to my adviser and mentor.

I am grateful to Prof. Q Jeffrey Ge and Prof. Nilanjan Chakraborty for their continuous guidance and support from the beginning. I express my gratitude to Prof. Imin Kao, Prof. Niranjan Balasubramanian for their presence on the thesis committee.

This work is standing on the shoulders of my teachers, family, and friends. I express my deepest gratitude to my teachers, especially Dr. Phadke, Mr. Reddy, Mr. Nirmal, and Mrs. Dandekar. I am grateful to have a big supportive family. I owe a great deal of gratitude to my friends, who made this journey possible, especially Anshul, Shashank, Agranya, Bhushan, Digvijay, Amol, Shivraj, Snehil and Arunjot.

Contents

Chapter

1 Introduction **4**

1.1 Introduction to Research Statement 4

1.2 Motivation 5

1.3 Overview 7

1.4 Juxtaposition with Prior Art 10

2 Optimal Synthesis of Planar Four-bar Linkages for Extended Burmester Problem **15**

2.1 Introduction 15

2.2 Kinematic Mapping 19

2.3 Generalized (G-) Constraint Manifold 20

2.4 Task Driven Geometric Constraints 21

2.4.1 Pose Constraint 22

2.4.2 Point Constraint 22

2.4.3 Line Constraint 23

2.4.4 Quadratic Curve Constraint 23

2.5 Algebraic Fitting of Linear Constraints 24

2.6 Error Functions and Constraints 26

2.6.1 Inequality Constraints 27

2.6.2 Minimization using Lagrange multipliers 27

2.7 Examples . . . 29
2.7.1 Optimal solution for Five positions with no exact solution . . 29
2.7.2 Optimal Linkage for Four Precision Poses with Region Constraint 32
2.8 Conclusion . . . 37

3 Defect-Free Kinematic Synthesis by Partial Shape Matching in a Clustered Database **40**
3.1 Introduction . . . 40
3.2 Signatures of Coupler Path and Motion . . . 46
3.3 Signature Matching and Error Function . . . 50
3.3.1 Partial Matching of Path Signatures . . . 52
3.3.2 Partial Matching of Motion Signatures . . . 53
3.3.3 Objective Function for Synthesis . . . 55
3.4 Sensitivity Analysis of Signatures . . . 58
3.5 Clustered Database of Planar Linkages . . . 58
3.5.1 Dimensionality Reduction using Auto-Encoders . . . 63
3.6 Case Studies . . . 66
3.6.1 Path Generation . . . 66
3.6.2 Motion Generation . . . 66
3.7 Conclusion . . . 73

4 Generative Models for Mechanism Synthesis **74**
4.1 Introduction . . . 74
4.2 Review on Neural Network Architectures . . . 75
4.2.1 Convolutional Neural Networks . . . 75
4.2.2 CNN-VAE Architecture for Image Representation of Coupler Curves . . . 77
4.3 Theory of Variational Auto Encoders (VAE) . . . 77
4.4 Recognition Model . . . 81

4.5 Generator Model . . . 82
4.6 Variants of VAE . . . 82

5 Computational Creativity and Task Conditioning **83**
5.1 Introduction . . . 83
5.2 Task Conditioning Overview . . . 83
5.3 Representation of Tasks . . . 84
5.3.1 Data Normalization . . . 85
5.3.2 Representing Task as an Image . . . 86
5.4 VAE Training . . . 89
5.5 Recognition and Generation of Coupler Trajectories . . . 90
5.6 Interactive Shape Modification with VAE . . . 92
5.6.1 Type Synthesis via Recognition . . . 93
5.6.2 Example User-ML Interaction . . . 93
5.7 Feasibility Conditioning and Input Feedback on Image-based Task . . . 95
5.8 Latent Space Exploration . . . 97
5.8.1 Informed Latent Exploration . . . 100
5.9 Context Conditioning . . . 100
5.9.1 Context as Desired Fixed Pivot Regions . . . 103
5.10 Nearest Latent Neighbors for Variational Path Synthesis . . . 107
5.11 Linkage Clustering . . . 107

6 Conditional Generation of Linkages **109**
6.1 Introduction . . . 109
6.2 Conditional Variational Auto Encoders (C-VAE) . . . 109
6.2.1 State of the linkage . . . 111
6.2.2 Training C-VAE for Mechanism Synthesis . . . 114
6.3 End to End Variational Synthesis of Planar Linkages . . . 116
6.3.1 Case Study . . . 116

7 Conclusion **121**

Bibliography **123**

Appendix

Tables

Table

2.1 Example 2.7.1: Pose Data 32

2.2 Example 2.7.1: Four singular vectors obtained after SVD of the matrix $[A]$ of size 4×8. 32

2.3 Example 2.7.1: Two optimum dyad-vectors obtained as result of optimization 34

2.4 Example 2.7.2: Pose Data 34

2.5 Example 2.7.2: Four singular vectors obtained after SVD of the matrix $[A]$ of size 4×8. The vectors form basis for the null space 35

2.6 Example 2.7.2: Four optimum dyad-vectors obtained as result of optimization 35

2.7 Real Solutions for α_i, λ_i and μ 37

2.8 Optimality Evaluations for Dyads 37

3.1 Case Study 1 : Path Data 67

3.2 Linkage Parameters of Nine Nearest Neighbor Paths 67

3.3 Case Study 2: Pose Data 72

3.4 Case Study 2: Linkage Parameters corresponding to Nine Nearest Neighbor Motions 72

4.1 Recognition Network Architecture 77

4.2 Generative Network Architecture 78

5.1 Datasets used for Training VAE 91

6.1 VAE and C-VAE Models : Architectures (FB=Four-bar, SC=Slider-Crank, SB=Six-Bar) 114

6.2 VAE and C-VAE Models : Training Losses 114

Figures

Figure

1.1 Slightly different inputs result in entirely different mechanisms. The input on the left generates a defect-free four-bar linkage. The input in the middle is formed by perturbing the orientation of the fourth pose by 15 degrees. The dashed frame in the middle input represents the original unaltered pose. Increasing the number of poses does not help either as shown by the mechanism generated on the right. 6

1.2 The framework accepts the task and the context. AI-assistant performs task conditioning to compute a distribution of conducive inputs. Task conditioning also provides feedback on the quality of inputs. Samples from this distribution are fed to suitable synthesis algorithms to obtain a variety of concept solutions. 8

1.3 Crude Input from the user is passed through Recognition Model which captures the features based on which various further actions are determined. The user has the control to override the feature selection by inspecting reformed variational inputs. These variational inputs are passed to classical synthesis methods like [1],[2] to generate a multitude of solutions. Solutions are passed through another Recognition and Clustering Module to present the user with distinct distributions of solution concepts. In addition, an End-to-End deep generative model is trained that conditions coupler paths to linkage parameter distributions. 11

1.4 Overview: Raw input is normalized and discretized into an image. This image is passed through the recognition network of a trained VAE which computes a probability distribution of latent features describing conducive variations of the raw input. Samples from this distribution are queried to find nearest neighbors in a dataset of four-bar and six-bar linkages. The linkages corresponding to the nearest neighbors have coupler curves similar to the conducive variations. 12

2.1 Example 2.7.1: Optimal four-bar mechanism that minimizes algebraic fitting error for third pose. Although second pose lies on different circuit, this Grashof type four-bar produces desired continuous motion from first to last pose. 33

2.2 Example 2.7.2: Five landing gear positions are shown where the third position can be relaxed. Allowed region for fixed pivots of mechanism is also shown. 33

2.3 Example 2.7.2: First and second dyad in Table 2.6 are combined to form the linkage shown. It can be clearly seen that coupler curve fairly approximates the third pose while fixed pivots are inside the allowed region. 38

2.4 Example 2.7.2: Image-Space representation of intersection of third and fourth optimal constraint manifolds from Table. 2.6. Also shown are five image points as dark spheres representing the five poses; four of them lie exactly on the intersection of the two surfaces, while one is closest possible. 38

3.1 The four-bar mechanism obtained using the precision position approach suffers from circuit defect, as no coupler circuit passes through all precision positions. 41

3.2 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 defect-free conceptual design solutions for input problems. 44
3.3 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. 49
3.4 Curvature and its unsigned integral for the path shown in Fig. 3.3 . . 49
3.5 Path and motion signatures of the motion shown in Fig. 3.3 50
3.6 **Part** path is formed by trimming whole path followed by translation and scaling. Arrows indicate the increasing direction of parameter t. . 53
3.7 Path Signatures of part p and whole W from the Fig. 3.6. 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. 54
3.8 Normalized cross correlation of the signatures computed along each direction is shown. It can be seen that exact match is found at $j = 0$. 54
3.9 Motion Signatures of the trajectories shown in Fig. 3.6. The domain as well as range of motion signature is invariant to similarity transformation. 55
3.10 Dissimilarity function of two motion signatures along both directions. It can be seen that the exact match is found at $j = 0$, where the template is fully embedded inside the other motion. 56
3.11 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. 57

3.12 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. 59
3.13 Distance (E_{min}) from Eq. (3.4) 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. 59
3.14 Motion signatures obtained by steps given in 3.2. Although topology difference is even more evident in this representation, it also signifies the similarity pattern between them. 60
3.15 Probability Distribution function used in random sampling for parameters l_1, l_2 and l_3. 62
3.16 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 lower dimensions, the lowest at the bottleneck layer. 65
3.17 Case Study 1: Path traced by hip joint during Sit-to-Stand Motion. . 67
3.18 First eight linkages in the table 3.2 and their resultant coupler paths. 68
3.19 Case Study 2: User specified motion necessary for the snow shoveling task. 69
3.20 Case Study 2 - Query Result: Motion signatures in the dataset with highest similarity. 69
3.21 Case Study 2: First eight linkages in the table 3.4 and their resultant coupler motions. 70

4.1 A filter is a 2D matrix whose coefficients are the learnable parameters of the convolution network. During convolution, each of such filter it is convolved on the input followed by nonlinear ReLU activation resulting in k such 2D matrices as shown by the middle entity. Next, the output is max pooled to formulate the output to be passed on to the next layer. 76

4.2 Recognition Model encodes the observed data X into probabilistic latent coding z of dimension much smaller than x. In this case, we assume a multivariate Gaussian distribution for z. Generative Model takes samples from this distribution to generate output $\hat{x}$. 78

4.3 CNN-VAE Architecture for Image Representation of Coupler Curves. For encoder, each hidden layer consists of convolution, ReLU and max pooling operation. For decoder each hidden layer consists of transposed-convolution and relu operation. 78

5.1 Each curve is normalized for position, scale and orientation. 86

5.2 The extreme coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$ of a normalized coupler curve are shown. If a point lies in Cartesian span of pixel, it is given 1.0 probability. 87

5.3 A histogram of extreme coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$ of coupler curves. It can be seen that less than 1% of the normalized coupler curves have a point with an absolute coordinate larger than 3.5 units. 88

5.4 Reconstruction and KL Divergence losses for two architectures with z dimension 2 and 3. It can be seen that higher z-dimension enables capturing more variation in the database, which results in lower reconstruction losses. 89

5.5 The training progress is reflected in the reconstruction quality of test coupler curve images. The corresponding variational lower bound $L_{ElBo}^{X^i}$ is displayed above each reconstructed image. It can be seen that the probability assignment by VAE for each pixel improves over the training. 90

5.6 The raw user input X is passed through a recognition network, which captures the salient information in the form of multivariate distribution of latent features. Random samples from this distribution are fed to the generator to generate paths with a high likelihood of producing good solutions. Moreover, users can manipulate the sample location in latent space, which gives them a low-dimensional and higher level of control on modifying the shape of the path. 91

5.7 Samples generated from a generator from a two-dimensional latent vector which is a vertex in the two-dimensional uniform grid with limits shown in corners. It can be seen that samples generated from neighboring vertices are highly correlated, which gives away the indication that recognition network learns the salient information about the shape of coupler curves. 94

5.8 Visualization of 2-D feature embedding obtained by training a VAE on closed coupler paths of various planar linkages. The variety in features by means of spread over the space directly relates to the variety in the shape of respective linkage type. 94

5.9 C-VAE generates coupler motions corresponding to the path input. The orientation information of generated motion is not shown for cleaner illustrations. The generated motions are fed as inputs for the motion generation problem. The solutions solved by [1] are passed through a linkage recognition model which results predicts the latent distribution for each solution. This mixture of distributions is clustered to form distinctive concept distributions. The four-bar linkage in the bottom left corner depicts the optimal path synthesis solution using [3]. 96

5.10 Raw input image is passed through VAE which provides a feasible input image. The shift in the probabilities is highlighted in the rightmost images. The shift shown in dark indicates that the portion of raw input is highly unlikely for linkage from the dataset to interpolate. This provides visual and intuitive feedback to the user on the input. . 98

5.11 The raw input on the top is close to the true distribution of coupler curves, thus VAE retains almost 70% of the highest intensities as depicted by the histogram on the top. For the second input, VAE overrides user assignments on the input pixels by a large number. This is due to the highly spiral nature of the input. It can be also seen in Fig 5.10 that the inner portion of the input is highly penalized. . . . 99

5.12 Latent exploration in axisymmetric directions is depicted. The original latent vector z is translated in axisymmetric directions. The arrow towards $\delta z_{1(+}$ indicates z is translated in positive direction along $z1$ axis in ten-dimensional space. 99

5.13 Examples of linear and spherical interpolation between a pair of known coupler images. $G(z_l))$ and $G(z_s))$ denote reconstructions for linear and spherical interpolations respectively. It can seen that in some cases, the interpolation passes through infeasible region indicated by higher $\mathcal{L}_{\text{ELBO}}$. 101

5.14 Illustration of gaps in latent spaces that lead to higher reconstruction errors if a sample is drawn from those regions. Two interpolation strategies are shown, one of which passes through the infeasible region. 102

5.15 The fixed pivots for the linkage are transformed according to the normalization of the coupler curve. The location of each fixed pivot is discretized to create a one-hot vector of 49 dimensions, depicted in an image form. 104

5.16 Classifier predicts a probability for each of the 49 pixels. The probability indicates the likelihood that the region spanned by that pixel contains a fixed pivot. The pixels where the fixed pivot lie are given a significantly high probability than the rest of the pixels. 104

5.17 The approach start with a linkage with coupler curve image X and fixed pivots at $FP_{true}(X)$. A trained classifier $FP_{NN}(X)$ accurately predicts the true location of fixed pivots. The X is reconstructed using a trained generative model which will be used in context conditioning. The image titled $FP_{target}(X)$ highlights a portion of classifier which will be subjected to maximization in the optimization routine given in Algorithm 3. 105

5.18 Value of objective function over 100 steps of gradient based stochastic optimization. 105

5.19 Initial and final coupler curve images with their corresponding fixed pivot region predictions. It can be seen that only subtle changes in the coupler curve are needed to satisfy the desired fixed pivot predictions. 106

5.20 The linkages corresponding to the K nearest neighbor in latent space for the given raw input are depicted. It can be seen that the linkages exhibit a wide variety in terms of their coupler path; thereby capturing input uncertainty. 108

6.1 Recognition Model encodes the observed data X and observed property Y into probabilistic latent coding z of dimension much smaller than X. In this case, we assume a multivariate Gaussian distribution for z. Generative Model takes samples from this distribution and combines it with Y to generate output $\hat{X}$ 110

6.2 A four-bar linkage with the fixed link of unit magnitude and co-linear with X-axis. The polar coordinates of points P_1, P_2, and P_3 stacked together for m crank orientations constitute the state representation of the four-bar. 113

6.3 A Stephenson six-bar linkage with the fixed link of unit magnitude and co-linear with X-axis. The polar coordinates of points $\{P_i\}_{i=1}^{6}$ stacked together for m crank orientations constitute the state representation of the six-bar 113

6.4 Reconstruction and KL Divergence losses for C-VAE-FB and C-VAE-SB. The architectural details of these models are given in Table 6.1. . 115

6.5 Sample linkages generated by C-VAE-LF10 when it is supplied with the conditional coupler curve Y and 10 dimensional Gaussian multivariate z. Architecture of this C-VAE is presented in Table 6.1. 117

6.6 Stephenson Six-bars generated by C-VAE-LS15 (see Table 6.1) conditioned for coupler curve Y and 15 dimensional Gaussian multivariate z. 117
6.7 Stephenson Six-bars generated by C-VAE-LS15 (see Table 6.1) conditioned for coupler curve Y and 15 dimensional Gaussian multivariate z. The mechanisms having pivots in desired region are selected for the next step 118
6.8 Collection of Feasible Six-Bar Mechanisms with desirable properties. It can be seen that their fixed pivots are in the desired shaded region 119
6.9 Final Linkage Concept Solutions 120

Abbreviations

C-VAE Conditional Variational Auto-Encoders

pose Combination of position and orientation

VAE Variational Auto-Encoders

Nomenclature

prior a probability distribution that would express one's beliefs about this quantity before some evidence is taken into account

$f_{\textbf{context}}(X)$ a fitness function in terms of linkage trajectory X

$Q(X_{task})$ Recognition model of a VAE trained to recognize observed distribution of real data X

$\mathcal{L}_{\textbf{ELBO}}$ Lower Bound of Marginal Likelihood.

X A general term representing target property of linkage for which the kinematic synthesis is conducted. In case of motion (or path) generation problem, X is coupler motion (or path)

$\hat{X}_{\textbf{context}}$ A feasible task which is conditioned for a given context

$X_{\textbf{motion vector}}$ Vector representation of motion as a sequence of poses

$X_{\textbf{fourbar}}$ Vector representation of joint trajectories of fourbar

$X_{\textbf{path image}}$ Image-based representation of path

$X_{\textbf{path vector}}$ Vector representation of path as a sequence of points

$X_{\textbf{task}}$ a general term representing prescribed path for path generation problem, prescribed motion for motion generation problem and prescribed function for function generation problem

z Latent Feature Vector

Publications

(1) Loya, A., Deshpande, S., Purwar, A."Machine Learning Driven Individualized Gait Rehabilitation: Classification, Prediction, and Mechanism Design", ASME Journal of Engineering in Medical Diagnostics and Therapy, May 2020; 3(2): 021105

(2) Deshpande S, Purwar A. "Computational Creativity via Assisted Variational Synthesis of Mechanisms using Deep Generative Models", ASME Journal of Mechanical Design 2019; doi :10.1115/1.4044396

(3) Deshpande S, Purwar A. "A Machine Learning Approach to Kinematic Synthesis of Defect-Free Planar Four-Bar Linkages", Feb 2019, ASME Journal Computing and Information Science in Engineering, doi 10.1115/1.4042325

(4) Deshpande S, Purwar A. "A Task-Driven Approach to Optimal Synthesis of Planar Four-Bar Linkages for Extended Burmester Problem", ASME Journal of Mechanisms and Robotics. 2017; doi:10.1115/1.4037801

(5) Ge Q. J., Purwar A, Zhao P, Deshpande S. "A Task-Driven Approach to Unified Synthesis of Planar Four-Bar Linkages Using Algebraic Fitting of a Pencil of G-Manifolds", ASME Journal of Mechanisms and Robotics. 2017; doi:10.1115/1.4037801

(6) Purwar, A., Deshpande S., Ge, Q. J.'MotionGen: Interactive Design and Editing of Planar Four-Bar Motions for Generating Pose and Geometric Constraints", ASME Journal of Mechanism and Robotics, 2017. doi:10.1115/1.4035899

Chapter 1

Introduction

1.1 Introduction to Research Statement

A mechanism is defined to be a collection of rigid bodies connected by joints such as hinges or sliders in order to generate articulated motions. Kinematic synthesis of mechanism deals with computing type and the dimension of mechanisms that are useful for a particular task[1] (X_{task}). The past forty years of research in kinematic synthesis of mechanisms has witnessed an unprecedented volume of work in formulating and solving planar linkage synthesis problems. However, finding practical and useful mechanisms for the synthesis problems has proven to be a difficult feat.

Mechanism synthesis is a critical part of the conceptual design phase, which requires synthesis methods to be prolific in terms of concept generation to 1) realize the potential of attainable design possibilities, and 2) have the agility to adapt a design to evolving requirements. To achieve the above goal, the proposed research brings together machine learning and theoretical kinematics to create a data-driven computational framework for kinematic synthesis of mechanisms. This work comprises of algorithmic developments presented in Chapter 2 that solve some of the outstanding challenges faced by the previously developed state of the art methods for kinematic synthesis of mechanisms. In the next phase of the work, a data-driven approach is presented which couples the synthesis algorithms with novel machine learning models that enhance the prolificacy and transparency of the crucial conceptual design phase.

[1] a general term describing prescribed path for path generation problem, prescribed motion for motion generation problem and prescribed function for function generation problem

This is achieved by 1) inception of an End to End synthesis pipeline that accepts deliberately imprecise or inherently uncertain input from users and provides them with a large variety of acceptable solution concepts, and 2) an approach using state-of-the-art deep generative models that learn joint probability distribution of various linkage parameters and their interdependence to perform tasks such as input conditioning, imputation, and variational synthesis. In doing so, we leverage the emerging machine learning techniques to learn meaningful representations and combine them with simultaneous type and dimensional methods of kinematic synthesis to enhance users' computational creativity. This also enables us to answer the following questions: 1) what are the infeasible aspects of the input and how to modify them to make a more conducive input, 2) given a path or a motion task, how likely it is that a particular type of mechanism can perform the task, 3) for a given task, what is the distribution of linkage parameters with similar coupler motions (or, paths).

1.2 Motivation

Depending upon the nature of the task, synthesis problems have been divided into mainly three categories: 1) Function Generation, 2) Path Generation, and 3) Motion Generation. The Function Generation task demands a prescribed relationship between the rotation of input and output links of the mechanism. In the Path Generation task, the aim is to move an object through space along a prescribed path, whereas in Motion Generation a rigid body needs to be guided along a prescribed motion. Hereinafter, Motion is termed as a continuous sequence of poses, where a pose is a combination of position and orientation.

A general synthesis procedure can be represented as,

$$\textbf{Solver}(X_{\text{task}}) = \{L^i_{\text{para}}\}^{i=n}_{i=1}. \tag{1.1}$$

Here, a synthesis solver **Solver** takes the task X_{task} as the input and produces n

solutions. L^i_{para} is the set of linkage parameters for the i^{th} solution. X[2] is the targeted linkage property prescribed by X_{task}. Methods for mechanism synthesis start with taking input X_{task} in the form of a sequence of path-points, poses, or functional input-output relationship from the user. A large majority of Motion and Path Generation methods take the precision point approach, which optimally minimizes the least squared fitting error between X_{task} and X. This approach is highly susceptible to input precision points/poses and in most cases results in solutions with circuit or branch defects [4]. The root cause of sensitivity is in the underlying nature of the interpolation problem.

To illustrate this chaotic nature, let us consider the following example. Six poses are sampled from a four-bar linkage as shown in Fig. 1.1. As the poses are already

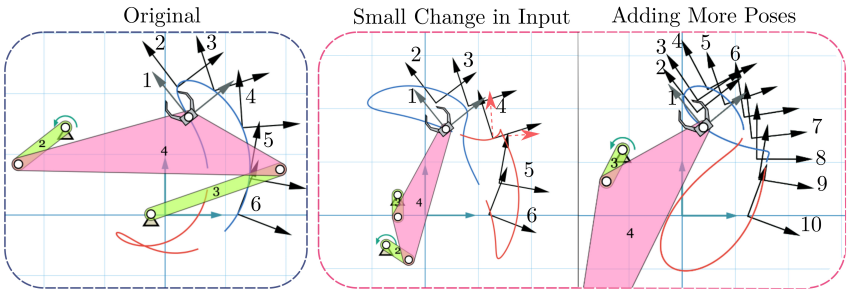

Figure 1.1: Slightly different inputs result in entirely different mechanisms. The input on the left generates a defect-free four-bar linkage. The input in the middle is formed by perturbing the orientation of the fourth pose by 15 degrees. The dashed frame in the middle input represents the original unaltered pose. Increasing the number of poses does not help either as shown by the mechanism generated on the right.

known to lie on the coupler motion, our motion generator algorithm[2] explained in chapter 2 obtains the original four-bar as expected. However, even a small change in the orientation of a pose results in an entirely different linkage suffering from branch defect, therefore making it unsuitable to perform the function. Moreover, increasing the number of positions by means of a B-spline interpolation through original poses does not help as shown in the last of Fig. 1.1. It is important to note that the

[2] X is a general term representing target property of linkage for which the kinematic synthesis is conducted. In case of motion (or path) generation problem, X is coupler motion (or path)

algorithm [2] used for synthesis in the example is a representative of the approaches based on the precision point approach. In the real world, when a user inputs a sequence and receives a result not suitable for the application, no informed decisions can be made to rectify the situation. To add more uncertainty, the task is often an approximation of the designer's intended motion. To accommodate this uncertainty, a common approach is to do random perturbations within the tolerance of target path or motion in the hope of getting a good solution. The probability of a random perturbation to find a valid input reduces exponentially with the number of precision positions as well as the tolerance range.

To perform informed and meaningful modification to X_{task}, it is necessary to possess knowledge about the properties of X. Let us define this knowledge as a **prior probability distribution**. In Bayesian statistical inference, a prior probability distribution, often simply called the **prior**, of an uncertain quantity is the probability distribution that would express one's beliefs about this quantity before some evidence is taken into account.

While the sensitivity to the input has been a problem in finding good solutions, it turns out that we can exploit the susceptibility of synthesis algorithms by providing them with a variety of preconditioned inputs so as to find a **diverse** range of defect-free solutions. By providing tools that can provide the designer with higher level control on input specification, we can help them be more creative as well.

1.3 Overview

The key contribution of the framework is in the development of an AI assistant which acts as an intermediary between designer and computational solvers. An overall view of the approach is depicted in Fig. 1.2. As it can be seen in Fig. 1.2, the framework accepts X_{task} and the context for the synthesis problem. The context provides a secondary prescription in the form of fitness function ($f_{\text{context}}(X)$) that

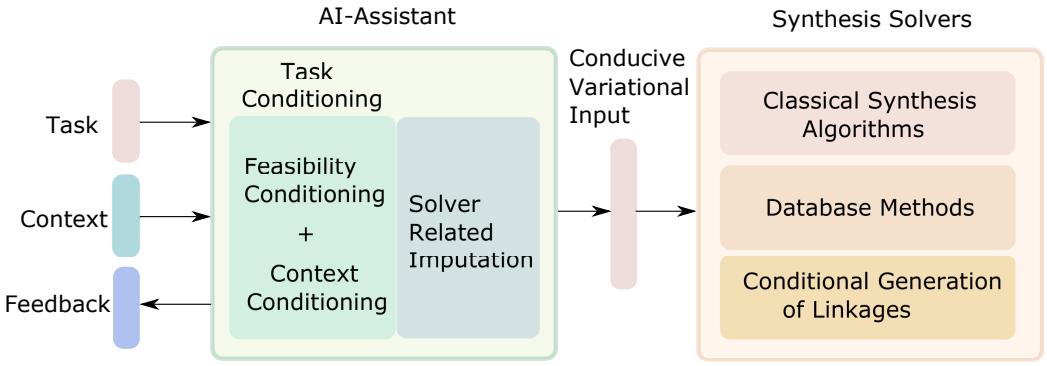

Figure 1.2: The framework accepts the task and the context. AI-assistant performs task conditioning to compute a distribution of conducive inputs. Task conditioning also provides feedback on the quality of inputs. Samples from this distribution are fed to suitable synthesis algorithms to obtain a variety of concept solutions.

evaluates the fitness of a solution. For example, it can be used to prescribe the desired region for fixed pivots to lie on. The framework allows for any such differentiable fitness function of X that is known or can be learned from data.

Next, X_{task} is subjected to task conditioning. This is done as follows. The framework captures the salient features z in the task by the means of variational inference and computes a distribution of conditioned inputs for computational solvers. The task is conditioned so that the likelihood of getting a desirable solution from the solver is maximized. The desirable solution is defined as the linkage which is free of defects and satisfies the conditions implied by the context. Here, we define feasibility conditioning as modifying the task to make it more conducive for obtaining defect-free solutions by synthesis. Whereas, the context conditioning is defined as modifying the task such that it is more likely to satisfy the imposed context.

In addition to the conditioning, the framework also incorporates missing information required by computational solvers. This work presents three novel algorithms for kinematic synthesis. Lastly, the outputs are post-processed and the designer is presented with a set of distributions of solutions, where each set consists of a concept with different variations. We define this approach, where the input uncertainty is intelligently managed to generate a distribution of solutions, as **Variational Synthesis of Mechanisms**. Figure 1.3 presents such an example depicting Variational Synthesis. The details of task conditioning are discussed in chapter 5.

Figure 1.3 depicts an example to demonstrate the efficacy of our approach, where we have presented solutions for path generation problem using our motion generation solver as used in the previous example; see Fig. 1.1. In the absence of using this approach, the solver would mostly produce impractical and defective solutions for general motion problems with a large number of poses. Thus, we have shown the effectiveness of the conditioning and input imputation by constructing valid motions from a crude input of path points. It should be noted that the main idea behind this

is not in solving the path generation problem using motion generation solvers, but to introduce the idea of an intermediary that handles user's incomplete and uncertain input and communicates the necessary numerical subtlety to a generic, susceptible computational solver. The work also presents an End-to-End deep neural network architecture, which approximates the conditional distribution of linkages with respect to the task.

In addition to the general framework, this work also presents a novel image-based approach for path generation, which is particularly amenable to mechanism synthesis when the input from mechanism designers is deliberately imprecise or inherently uncertain due to the nature of the problem. It models the input curve as a probability distribution of image pixels and employs a probabilistic generative model to capture the inherent uncertainty in the input. In addition, it gives feedback on the input quality and provides corrections for a more conducive input. The image representation allows for capturing local spatial correlations, which plays an important role in finding a variety of solutions with similar semantics as the input curve. Figure 1.4 shows an overview of this approach.

1.4 Juxtaposition with Prior Art

In the case of the Path Generation problem, Ullah and Kota [5] and Wu et al.[6] have tried to incorporate the prior on user input by means of finding lower harmonic Fourier Descriptors of the path followed by computing link dimensions using optimization methods. Li et al.[7] have developed a Fourier descriptor-based approach for approximate motion generation, which uses the same prior on coupler path. These methods are relatively robust to spatial variations in input but susceptible to variations in timing information provided by the user. Sharma and Purwar [3] have addressed this issue to some extent by providing a scheme to compute optimal timing for the input points. However, the above methods are only defined for the

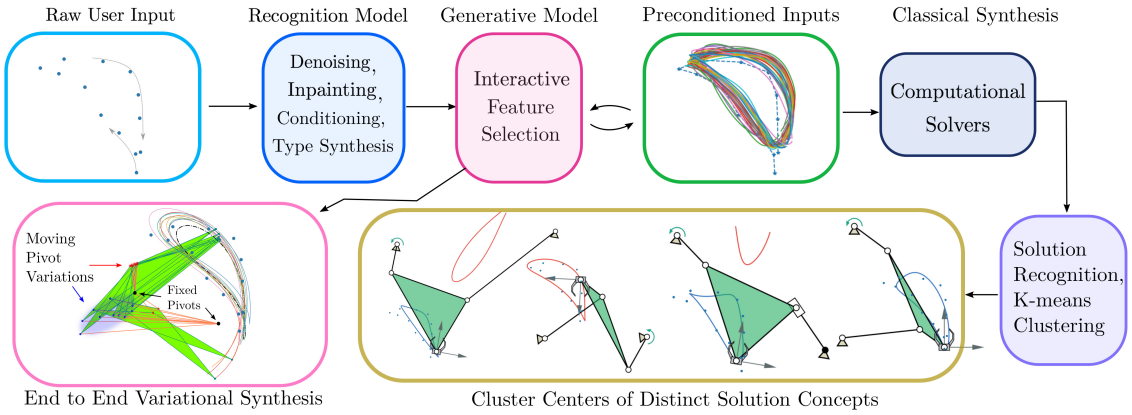

Figure 1.3: Crude Input from the user is passed through Recognition Model which captures the features based on which various further actions are determined. The user has the control to override the feature selection by inspecting reformed variational inputs. These variational inputs are passed to classical synthesis methods like [1],[2] to generate a multitude of solutions. Solutions are passed through another Recognition and Clustering Module to present the user with distinct distributions of solution concepts. In addition, an End-to-End deep generative model is trained that conditions coupler paths to linkage parameter distributions.

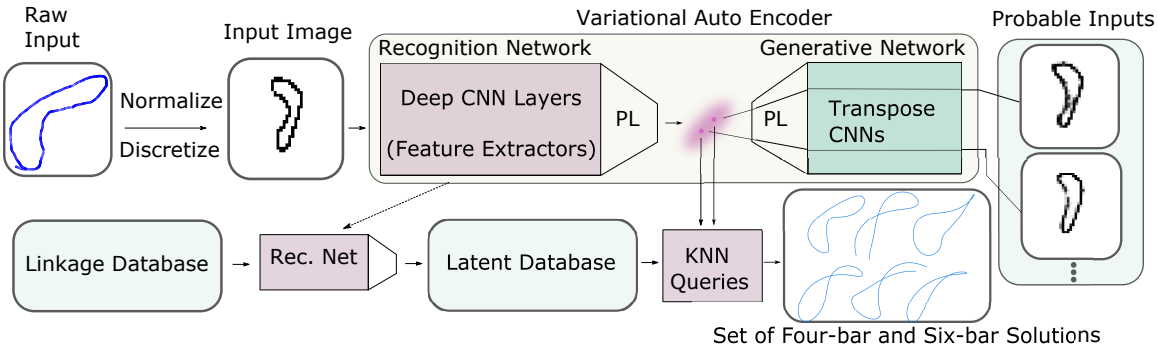

Figure 1.4: Overview: Raw input is normalized and discretized into an image. This image is passed through the recognition network of a trained VAE which computes a probability distribution of latent features describing conducive variations of the raw input. Samples from this distribution are queried to find nearest neighbors in a dataset of four-bar and six-bar linkages. The linkages corresponding to the nearest neighbors have coupler curves similar to the conducive variations.

synthesis of four-bar linkages with revolute joints. In contrast to this, our approach is general enough to condition any type of user input and scales to higher-order linkage mechanisms and spatial robots. A key difference in the previous approaches and our approach is that the conditioning on the input is performed by learning the joint probability distribution of input parameters from a database of linkages instead of relying on the fact that four-bar linkages produce curves which have specific harmonic content.

In order to learn such joint probability distributions, we employ a recently developed deep generative model called Variational Auto Encoder (VAE). It comprises of two neural networks: 1) Recognition Model (also called Encoder) and 2) Generative Model (also called Decoder). A trained VAE applies the learned prior probability distribution on observed input, thus acting as a posterior inference model. Additional benefits of using VAE are that the learned posterior inference model can also be used for a host of tasks such as denoising, representation, and imputation.

The proposed framework consists of two classes of VAEs. The first class is trained only on the coupler trajectories, which emphasizes representation learning of coupler paths (or, motions) for conditioning the task. The second class is trained to

learn the joint distributions of mechanisms in order to recognize their salient features, which is used for concept identification and clustering. A variant of VAE called conditional C-VAE is also developed which can be used to perform End-to-End variational synthesis with deep learning models.

Applications of powerful function approximators like neural networks are not new to the domain of mechanism synthesis. Vasiliu and Yannou [8] have used Artificial Neural Network (ANN) to interpolate the map between the path and link lengths for Grashof four-bar linkages with revolute joints. Here, ANNs are mainly utilized to memorize the mapping as a means to replace an atlas. Khan et al.[9] presented an approach where an ANN is used for mapping between Fourier coefficients corresponding to a coupler path and corresponding linkage parameters. Galan et al.[10] have used a similar approach but instead of using Fourier Coefficients, they have used wavelet descriptors to represent the shape of the path. All of the above methods use ANNs just as a mapping tool from a closed coupler path to mechanism link ratios. In contrast to the black box mapping approach of the above methods, we facilitate user interaction with the network by means of interactive manipulation of latent space. The approach taken by our methods is unsupervised and semi-supervised learning, with an emphasis on representation learning and understanding the probability distribution of coupler trajectories.

Although our approach can work with any number and type of linkages, we have implemented the methods for following topologies 1) four-bar with all revolute joints (RRRR), 2) slider-crank linkage (RR-PR) and its inversion (RR-RP) and 3) Stephenson I and IIIa six-bar with revolute and prismatic joints 4) Janson's Eight bar.

The organization of the dissertation is as follows. Chapter 2 presents the algorithmic developments that incorporate some of the practical constraints in a previously developed task driven simultaneous type and dimensional synthesis frame-

work [11], [12]. chapter 3 presents a novel contribution to database-driven methods for defect-free path and motion synthesis of planar linkages. Chapter 4 presents the theory behind the choice of using generative models. Chapter 5 presents the application of VAE in task conditioning, which is one of the key contributions of the dissertation. Chapter 6 presents the end-to-end machine learning based synthesis solver that learns a conditional distribution of linkages with respect to the task.

Chapter 2

Optimal Synthesis of Planar Four-bar Linkages for Extended Burmester Problem

2.1 Introduction

According to an article by Prof. McCarthy [13], over the past forty years, a total number of 2,688 U.S. patents have been awarded that involve four-bar linkages, while in the same period, the next natural extension to more complex linkages, viz., six-bar linkages have been awarded a mere 84 patents. This is a clear demonstration of the popularity and ubiquity of planar four-bar linkages. Therefore, it is not surprising that the four-bar linkage synthesis and analysis problem still receives a considerable attention of researchers.

Several text books, such McCarthy and Soh [14], Sandor and Erdman [15], Hunt [16], Hartenberg and Denavit [17], Suh and Radcliffe [18], and Lohse [19] cover the science and art of planar four-bar and higher-order linkages.

Kerle et.al.[20] have given a historical overview of the development of mechanisms for motion generation. Despite a glut of literature on this topic, simultaneous computation of type and dimensions and accommodating practical geometric constraint for the motion generation problem has not been explored much. Erdman et al. in their seminal Mechanism Design and Analysis text [21] clearly mention that assuming the wrong type (linkage topology and type of joints) to compute the dimensions of a linkage system may result in either none or sub-optimal solutions.

The work presented in this chapter is a continuation of our research [1, 22, 11],

wherein we have presented a task-driven approach to simultaneous type and dimensional synthesis of planar dyads for the motion generation problem. A four-bar linkage is constructed as combination of any two of the synthesized dyads. This dyadic construction simplifies the synthesis process and renders the method as modular building block for synthesis of mechanisms with more links such as six-bar mechanisms [1]. By using the concepts of kinematic mapping (Blaschke [23], Grünwald [24]) and planar quaternions (Bottema and Roth [25], McCarthy [26]), we obtained a unified form of kinematic constraints of the planar dyads and created an algorithm for unified type and dimensional synthesis of planar four-bar linkages. This is accomplished via a two-step process. The first step is algebraic fitting of image points on a pencil of G-manifolds using Singular Value Decomposition (SVD). This pencil of G-manifolds forms a candidate solution space for constraints accounted for in the SVD process. In the second step, we impose two fundamental quadratic conditions on the candidate solutions to extract the dyad types and their dimensions.

Ravani and Roth [27, 28] were the first to use kinematic mapping approach for mechanism synthesis. Thereafter, Bodduluri and McCarthy [29], Bodduluri [30], Ge and Larochelle [31], Larochelle [32, 33], Husty et al. [34], Hayes et al. [35, 36], Wu et al. [37], Purwar and Gupta [38], and Shrocker et al. [39] have used this approach for the motion generation problem.

The original contribution of the work presented in this chapter is in reformulation of the above framework in a general way so that classic Burmester problem can be extended to accommodate both pose- and other practical-geometric constraints, and so that it solves problems which our previous approach could not solve. These problems are 1) finding optimal approximate solutions for Burmester problem with no or sub-optimal solutions, and 2) finding optimal linkages that minimize the algebraic fitting error of non-linear geometric constraints. We note that in this work, we do not restrict Burmester problem to only five poses. Burmester [40] showed that only a finite

set of four-bars can be synthesized for five precision pose motion generation problem. In this chapter, we show that other than poses, other practical geometric constraints can also be accommodated. For more than five geometric constraints, typically only an approximate motion synthesis can be performed. Holte et al.[41], Sabada et al. [42], Venkataraman[43], have presented some techniques for mixed exact-approximate synthesis of planar mechanisms, albeit only for poses. Larochelle[44] has presented dimensional synthesis technique for solving the mixed exact and approximate motion synthesis problem for planar RR kinematic chains. Song [45] has presented Pole Curve Transformation based approach for motion synthesis of planar mechanisms. Al-Widyan et al.[46] have presented a numerically robust algorithm to solve the classic Burmester problem. Bourrelle et al. [47] presented a graphical user interface that uses the algorithm developed in [46] to solve the classic Burmester problem.

In general, there are two basic approaches to linkage synthesis for motion generation problem: 1) precision pose approach, and 2) error minimization approach. In this chapter, we use a combination of both methods in our framework.

Optimal synthesis of linkages is often a constrained non-linear and multi-modal problem in a multi-dimensional design space. In case of optimal linkage synthesis, relatively more work has been reported for path and function generation problem than for motion generation. Mariappan and Krishnamurty[48] used a generalized exact gradient method for planar mechanism synthesis. Vallejo[49] developed an optimization method for planar mechanisms with lower pairs of any type which uses error function as deformation of dimensions that mechanism has to undergo to perform the task. Yao and Angeles[50] solved path-generation problem using least squared error function. They employ contour method to find out all stationery points of the problem. Kramer and Sandor [51] have presented a method of optimal design of planar mechanisms called Selective Precision Synthesis (SPS). SPS can incorporate design requirements such as link ratios, fixed pivot locations, and transmission

angle range. Gogate and Matekar [52] used method of differential evolution to find optimal linkages based on three error functions which ensure that the mechanism is crank rocker, branch and circuit defect free with minimized positional and angular deviations. Venkataraman et al.[43] used tolerance based approach for four position problem. They searched for optimal design parameters along suitable ranges of center point curve. Cabrera et al.[53] have presented genetic algorithm on planar 4R mechanism synthesis for its simple implementation and fast convergence. Hegedus et al. [54] recently presented synthesis of spatial 6R linkages for interpolating four given poses using factorization of motion polynomials. All of the above said methods first select the type of linkage to be optimized, while our approach takes unified representation for all types of planar dyads into account. In contrast to other methods, which directly use linkage parameters, we formulate the optimization problem in terms of intermediate parameters obtained by geometric fitting. This allows us to use a two-step process wherein we first perform the least-squares fitting of the geometric constraints and then use a Lagrange multiplier method with additional linear and non-linear constraints to extract the dyad types and dimensions. This approach allows formulation of an objective function consisting of squared error for the constraints to be minimized, while keeping certain other constraints exactly satisfied. Ultimately, the minimization procedure leads to a system of quadratic equations, which can be solved using a Computer algebra softwares like Mathematica. Each real solution of this system of equation corresponds to one optimum dyad; all such dyads are computed and then combined pairwise to obtain a set of four-bar linkage solutions.

The organization of the chapter is as follows: Section 2.2 reviews the concept of kinematic mapping and planar quaternions as a special case of dual quaternions. Section 2.3 reviews unified form of kinematic constraints of the planar dyads as G-manifolds in the image space, while section 2.4 presents various geometric constraints in the image space. Section 2.5 presents how we algebraically fit various linear geomet-

ric constraints to the G-manifolds using SVD. Section 2.6 presents the Lagrange Multiplier method to minimize error functions while keeping certain linear- or non-linear constraints satisfied exactly. We finally present two practical examples in section 2.7.

2.2 Kinematic Mapping

A planar displacement consisting of a translation (d_1, d_2) and a rotation angle ϕ from a moving frame M to a fixed frame F is represented by a planar quaternion $\mathbf{Z} = (Z_1, Z_2, Z_3, Z_4)$ where (see [27, 26] for details),

$$\begin{aligned} Z_1 &= \tfrac{1}{2}(d_1 \cos\tfrac{\phi}{2} + d_2 \sin\tfrac{\phi}{2}), \quad & Z_2 &= \tfrac{1}{2}(-d_1 \sin\tfrac{\phi}{2} + d_2 \cos\tfrac{\phi}{2}), \\ Z_3 &= \sin\tfrac{\phi}{2}, & Z_4 &= \cos\tfrac{\phi}{2}. \end{aligned} \tag{2.1}$$

The components (Z_1, Z_2, Z_3, Z_4) define a point in a projective three-space called the *image space* of planar displacements [27]. Then, a planar displacement represented as a homogeneous transformation of point $\mathbf{x} = (x_1, x_2, x_3)$ or line $\mathbf{l} = (l_1, l_2, l_3)$ from M to F can be given by

$$\mathbf{X} = [H]\mathbf{x}, \tag{2.2}$$

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

$$\mathbf{L} = [\overline{H}]\mathbf{l}, \tag{2.3}$$

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

where $Z_3^2 + Z_4^2 = 1$ and $\mathbf{X} = (X_1, X_2, X_3)$ and $\mathbf{L} = (L_1, L_2, L_3)$ are corresponding point and line coordinates in F.

2.3 Generalized (G-) Constraint Manifold

In this section, we review a unified form of the kinematic constraints of four types of dyads (**RR**, **PR**, **RP**, and **PP**) in the image space; see [1] for details. A point **X** or line **L** on the coupler of a four-bar linkage can be geometrically constrained in one of the following four ways: 1) for an **RR** dyad, the point is constrained to be on a circle with center and radius given as homogeneous coordinates (a_0, a_1, a_2, a_3), 2) for a **PR** dyad, the point is constrained to be on a fixed line having coordinates (L_1, L_2, L_3), 3) for an **RP** dyad, a moving line (l_1, l_2, l_3) is constrained to be tangent to a circle (a_1, a_2, a_3), and 4) for **PP** dyad, a point on line (L_1, L_2, L_3) is constrained to move along another line $(2a_1, 2a_2, a_3)$. In [1], we have shown that all of these constraints reduce to a single quadratic equation in the Cartesian space. When the fixed frame coordinates of point and line from Eqns. (2.2) and (2.3) are substituted in this quadratic condition, we obtain following generalized equation:

$$\begin{aligned} &p_1(Z_1^2 + Z_2^2) + p_2(Z_1Z_3 - Z_2Z_4) + p_3(Z_2Z_3 + Z_1Z_4) \\ &\quad + p_4(Z_1Z_3 + Z_2Z_4) + p_5(Z_2Z_3 - Z_1Z_4) + p_6Z_3Z_4 \\ &\quad + p_7(Z_3^2 - Z_4^2) + p_8(Z_3^2 + Z_4^2) = 0, \end{aligned} \tag{2.4}$$

where the eight coefficients p_i are not independent but must satisfy two quadratic conditions

$$p_1p_6 + p_2p_5 - p_3p_4 = 0, \quad 2p_1p_7 - p_2p_4 - p_3p_5 = 0. \tag{2.5}$$

This is because p_i are related to the geometric parameters of the dyad by

$$\begin{aligned} &p_1 = -a_0, \quad p_2 = a_0x \quad p_3 = a_0y, \quad p_4 = a_1, \quad p_5 = a_2, \\ &p_6 = -a_1y + a_2x, \quad p_7 = -(a_1x + a_2y)/2, \\ &p_8 = (a_3 - a_0(x^2 + y^2))/4, \end{aligned} \tag{2.6}$$

where (a_0, a_1, a_2, a_3) are the homogeneous coordinates of the constraint circle expressed in fixed reference frame and (x, y) are the coordinates of the circle point

expressed in moving reference frame. Here, a_0 is the homogenizing factor. Equation (2.4) represents a generalized quadric (G-manifold) in the image space whose actual form would depend on the dyad type. For **RR** dyad, the quadric is a hyperboloid of one-sheet, while for other dyads, it is a hyperbolic paraboloid [26].

Inverse computation of these p_i into geometric parameters is done as follows:

$$\begin{aligned} l_1 : l_2 : l_3 &= p_2 : p_3 : 2p_8, \\ x_1 : x_2 : x_3 &= (p_6p_5 - 2p_7p_4) : -(p_6p_4 + 2p_7p_5) : (p_5^2 + p_4^2), \\ a_0 : a_1 : a_2 &= (p_2^2 + p_3^2) : (-p_3p_6 - 2p_2p_7) : (p_2p_6 - 2p_3p_7), \end{aligned} \tag{2.7}$$

Eqns. (2.4) and (2.5) are said to define the constraint manifold of all types of dyads (for details see [1]). For a PR dyad, we have $a_0 = 0$ and therefore, $p_1 = p_2 = p_3 = 0$; for the **RP** dyad, we have $p_1 = p_4 = p_5 = 0$; and for the **PP** dyad, we have $p_1 = p_2 = p_3 = p_4 = p_5 = 0$. In all of these cases, the quadratic conditions in (2.5) are clearly satisfied. Thus, all planar dyads can be represented in the same form by Eqns. (2.4) and (2.5), and we can determine the type of a planar dyad by looking at the zeros in the coefficients p_i (called signature of a dyad). This leads to a unified algorithm for simultaneous type and dimensional synthesis of planar dyads. In our approach, we first obtain the homogeneous coordinates p_i, determine the dyad type from the signature of coefficient array p_i, and then compute the dyad parameters using inverse relationships in (2.6). The coefficient array p_i forms an eight-dimensional vector, henceforth it is called a dyad-vector **p**.

2.4 Task Driven Geometric Constraints

In this section, we present various geometric constraints for the motion generation problem and show their representation in the image space. Some such constraints can be classified in following ways: 1) pose (position and orientation) constraint on the coupler, 2) constraint on moving or fixed pivot locations namely, point, line or general quadratic curve.

2.4.1 Pose Constraint

Precision pose synthesis problem requires coupler to go through given poses defined by Cartesian space parameters (d_1, d_2, ϕ). Converting these parameters into planar quaternions $(Z_{1p}, Z_{2p}, Z_{3p}, Z_{4p})$ using Eq. (2.1) and then substituting into Eq. (2.4) gives us a pose constraint expressed by,

$$\begin{aligned} &p_1(Z_{1p}^2 + Z_{2p}^2) + p_2(Z_{1p}Z_{3p} - Z_{2p}Z_{4p}) + p_3(Z_{2p}Z_{3p} + Z_{1p}Z_{4p}) \\ &+p_4(Z_{1p}Z_{3p} + Z_{2p}Z_{4p}) + p_5(Z_{2p}Z_{3p} - Z_{1p}Z_{4p}) + p_6 Z_{3p}Z_{4p} \\ &+p_7(Z_{3p}^2 - Z_{4p}^2) + p_8(Z_{3p}^2 + Z_{4p}^2) = 0, \end{aligned} \tag{2.8}$$

For the classic Burmester problem, five poses would be specified, each of which would give one such equation.

2.4.2 Point Constraint

Specifying locations on fixed or moving pivots of the mechanism proves to be useful in practice. Let (X_f,Y_f) be one of these specified fixed pivot locations. Constraining each coordinate of pivot gives rise to one linear equation which in terms of p_i can be given as,

$$\begin{aligned} X_f p_1 + p_4 = 0, \\ Y_f p_1 + p_5 = 0. \end{aligned} \tag{2.9}$$

The above equations follow directly from Eq. (2.6), where $X_f = a_1/a_0, Y_f = a_2/a_0$. It is worth noting that all type of dyads may not satisfy the imposed point constraints. This is due to fact that all RP dyads have dyad coefficients p_1, p_4, and p_5 zero, so they automatically satisfy (2.9) but do not necessary have fixed pivot on specified location. This problem can be easily tackled by filtering out the extraneous solutions. Moving pivot locations can also be provided in the same way as fixed pivot locations. They too form two linear equations given by

$$\begin{aligned} x_m p_1 + p_2 = 0, \\ y_m p_1 + p_3 = 0, \end{aligned} \tag{2.10}$$

where (x_m, y_m) are the coordinates of moving pivot location in moving reference frame.

2.4.3 Line Constraint

Line constraint for the fixed pivots constrains the center point (X_f, Y_f) of an **RR** dyad to a line $L_1X_f + L_2Y_f + L_3 = 0$. Using inverse relationships in (2.6), we obtain a linear equation in p_i given by,

$$-L_1p_4 - L_2p_5 + L_3p_1 = 0. \tag{2.11}$$

A similar constraint equation is obtained when the moving pivot of an **RR** dyad is constrained to a line (l_1, l_2, l_3) attached to the moving frame given by

$$-l_1p_2 - l_2p_3 + l_3p_1 = 0. \tag{2.12}$$

2.4.4 Quadratic Curve Constraint

Let (X_f, Y_f) be the location of a fixed pivot of **RR** or **RP** dyad constrained to lie on a general quadratic curve given by,

$$AX^2 + BXY + CY^2 + DX + EY + F = 0 \tag{2.13}$$

Then, substituting for $(X = X_f = -p_4/p_1, Y = Y_f = -p_5/p_1)$ into above Eq. (2.13), we obtain an image space representation of the constraint as follows:

$$Ap_4^2 + Bp_4p_5 + Cp_5^2 - Dp_4p_1 - Ep_5p_1 + Fp_1^2 = 0 \tag{2.14}$$

The above is a homogeneous equation of a quadric in the image space and its degenerate forms reduce to the point and line constraints given in Eqns. (2.9) and (2.11). A similar equation can be obtained when the moving pivot is constrained to the quadratic curve of the form in (2.13). Without any loss of generalization, for the demonstration purposes, let us look at an elliptical-curve constraint now.

Consider an axisymmetric ellipse whose major axis is x_1 and minor axis is y_1. Say, positive x_1 axis makes θ angle with positive x-axis of fixed reference frame and the origin of x_1-y_1 frame is at (x_e, y_e) with respect to fixed reference frame. Equation of ellipse is given by

$$\frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} = 1, \tag{2.15}$$

where r_1 and r_2 are radii of major and minor semi-axes and (x, y) are co-ordinates of a point on the ellipse w.r.t. x_1-y_1 frame. These coordinates relate to the coordinates in the fixed frame by

$$\begin{aligned} x &= (X - x_e)\cos\theta + (Y - y_e)\sin\theta, \\ y &= (-X + x_e)\sin\theta + (Y - y_e)\cos\theta \end{aligned} \tag{2.16}$$

Substituting (2.9) and (2.16) into (2.15) gives,

$$\begin{aligned} &\frac{((-p_4 - p_1 x_e)\cos\theta + (-p_5 - p_1 y_e)\sin\theta)^2}{r_1^2} \\ &+ \frac{((p_4 + p_1 x_e)\sin\theta + (-p_5 - p_1 y_e)\cos\theta)^2}{r_2^2} = p_1^2 \end{aligned} \tag{2.17}$$

All **RP** dyads ($p_1 = p_4 = p_5 = 0$) trivially satisfy (2.17), so extraneous **RP** dyads need to be filtered out. A similar constraint for moving pivots could be found out, which is given by,

$$\begin{aligned} &\frac{((-p_2 - p_1 x_e)\cos\theta + (-p_3 - p_1 y_e)\sin\theta)^2}{r_1^2} \\ &+ \frac{((p_2 + p_1 x_e)\sin\theta + (-p_3 - p_1 y_e)\cos\theta)^2}{r_2^2} = p_1^2 \end{aligned} \tag{2.18}$$

All **PR** dyads ($p_1 = p_2 = p_3 = 0$) trivially satisfy (2.18), so extraneous **PR** dyads need to be filtered out.

2.5 Algebraic Fitting of Linear Constraints

In this section, we show how we can solve the extended Burmester problem via a two-step algebraic fitting process. A straight-forward extension of the classic

Burmester problem may specify up to five linear geometric constraints of the forms given by Eqns. (2.8), (2.9), (2.10), (2.11), (2.12). From all geometric constraints we select five or less linear constraints and assemble them into a matrix given by,

$$[A]\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_8 \end{bmatrix} = 0, \text{ where } [A] = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \cdots & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & \cdots & A_{28} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & A_{n4} & \cdots & A_{n8}, \end{bmatrix}. \tag{2.19}$$

where constraints are linear equations in p_i of the form,

$$\sum_{j=1}^{n} A_{ij} p_j = 0, \tag{2.20}$$

where A_{ij} depends upon the type of i^{th} input constraint i.e. pose-, line- or point-constraint or an equivalent linear constraint. This system of equations is solved using SVD, which minimizes the least-squares error of fitting. This is a zero-eigenvalue problem and for $n = 5$ constraints, unique solutions for the dyad vector $\mathbf{p}$ can be obtained, if they exist. As the matrix [A] is $n \times 8$, where $n \leq 5$, nullity of the matrix is $8 - n$. Hence, there are $8 - n$ singular vectors which define null space of solution vectors given by $\{\mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_n\}$. Any vector $\mathbf{p}$ that lies inside this vector space is a solution to algebraic fitting of n geometric constraints given by,

$$\mathbf{p} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + ... + \alpha_{8-n} \mathbf{v}_{8-n} \tag{2.21}$$

where α_i are $(8 - n)$ homogeneous parameters, thus without loss of generality we can assume $\alpha_1 = 1$. For $n = 5$, the nullity of the matrix $[A]$ is three, therefore, three singular vectors corresponding to the smallest singular-values are selected to form the dyad vector $\mathbf{p}$. Substituting for $\mathbf{p}$ in the quadratic conditions given by Eq. (2.5), one can solve for two unknowns α_2, α_3. Examining the signature of $\mathbf{p}$, the dyad type is determined and by inverse computation in (2.7), the mechanism parameters are obtained. This fitting process also works for $n > 5$ cases, however,

only an approximate solution may be obtained in this case. Typically, for large values of n, such solutions mostly produce unsatisfactory results.

Now, consider the possibility that no real or unacceptable solutions emerge from the above fitting process for $n = 5$ case. This is not an impractical scenario – very often, for five pose problems, the fitting process yields none, non-Grashof, or defective linkages. Another possibility is that the designer does not care about the five poses to be interpolated exactly. Manufacturing errors in links and play at joints would never lead to an exact interpolation of constraints anyways. For a pick and place operation, the first and last poses may be critical, while in between poses may be desired to be reached in a **minimum-error** sense. Moreover, the designer may want to introduce more important constraints in the problem, such as adding a location for pivots. The SVD also does not admit a non-linear constraint of the form (2.13). In that case, the above formulation breaks down. We now need a method to relax certain constraints while simultaneously satisfying exact constraints. In the next section, we show how such problems are solved.

2.6 Error Functions and Constraints

This section presents the error function that is to be minimized during optimization. The error function would be created from the relaxed constraints. If linear geometric constraints like pose, line or point constraints are relaxed for minimization, then the algebraic error of measure can be defined by substituting expression for $\mathbf{p}$ from Eq. (2.21) into their respective constraint equations, which can be written as

$$f_1(\alpha_i; i = 1 \ldots 8 - n) = k_1\alpha_1 + k_2\alpha_2 + \ldots + k_{8-n}\alpha_{8-n}, \tag{2.22}$$

where k_i are defined in terms of the parameters of dyad vector $\mathbf{p}$ and the known constraint parameters; e.g., X_f, Y_f are the known constraint parameters for the fixed-pivot constraint.

Similarly, the algebraic error that evaluates distance of the fixed or moving

pivots from a known ellipse is given by

$$f_2(\alpha_i; i = 1 \dots 8-n) = \sum_{i=1,j=1}^{i=8-n,j=8-n} k_{ij}\alpha_i\alpha_j + \sum_{i=1}^{i=8-n,} k_i\alpha_i + k_0 \tag{2.23}$$

where k_0, k_i, k_{ij} are also defined in terms of the parameters of dyad vector $\mathbf{p}$ and the constraint parameters.

In addition, dyad vector parameters p_i also need to satisfy two quadratic relations in (2.5). Thus, two quadratic equality constraints obtained by substituting (2.21) into (2.5) are of the form:

$$\begin{aligned} h_1(\alpha_i; i = 1 \dots 8-n) &= \textstyle\sum_{i=1,j=1}^{i=8-n,j=8-n} k_{1ij}\alpha_i\alpha_j, \\ h_2(\alpha_i; i = 1 \dots 8-n) &= \textstyle\sum_{i=1,j=1}^{i=8-n,j=8-n} k_{2ij}\alpha_i\alpha_j, \end{aligned} \tag{2.24}$$

where k_{1ij}, k_{2ij} are defined in terms of dyad vector parameters p_i.

2.6.1 Inequality Constraints

Sometimes user does not have strict requirements on exact location of the pivots but a region where they should lie, such requirements can be modeled as inequality constraints on pivots. One example for bounded regional constraint is the inner region of an ellipse, given by

$$\frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} - 1 \leq 0, \tag{2.25}$$

Substituting such constraints into dyad vector parameters and subsequently in coefficients α_i using Eq. (2.21), we get a form given by

$$g = \sum_{i=1,j=1}^{i=8-n,j=8-n} k_{ij}\alpha_i\alpha_j + \sum_{i=1}^{i=8-n,} k_i\alpha_i + k_0 \leq 0 \tag{2.26}$$

2.6.2 Minimization using Lagrange multipliers

In the previous section, we created algebraic fitting function for various linear and non-linear constraints. Now, we present the Lagrange multiplier method to solve the optimization problem.

For a task of n linear and m non-linear geometric constraints with u linear and v non-linear error functions with corresponding weight ω_{1i} and ω_{2j},

$$Minimize f(\alpha_i) = \sum_{i=1}^{i=u} {f_{1_i}}^2 \omega_{1i} + \sum_{j=1}^{j=v} {f_{2_j}}^2 \omega_{2j} \tag{2.27}$$

subjected to constraints

$$h_1(\alpha_i) = 0 \qquad h_2(\alpha_i) = 0,$$

$$h_3(\alpha_i) = \sum_{i=1,j=1}^{i=8-n,j=8-n} k_{ij}\alpha_i\alpha_j + \sum_{i=1}^{i=8-n,} k_i\alpha_i + k_0 = 0, \tag{2.28}$$

where $h_3(\alpha_i)$ is a non homogeneous quadratic equation, which exists if nonlinear geometric constraints like (2.17), (2.18) are present. Weights ω_{1i} and ω_{2j} can be chosen by the user according to the synthesis requirements by weighing the relative importance of the constraints. The method of Lagrange multipliers is used to obtain a set of optimum solutions. In this method, the Lagrange objective function is defined as,

$$F(\alpha_i, \lambda_i) = -f(\alpha_i) - \lambda_1 h_1 - \lambda_2 h_2 - \ldots \lambda_{(m+2)} h_{(m+2)} - \mu g \tag{2.29}$$

Here, there are $(8-n)$ number of unknowns α_i and $(2+m)$ number of λ_i. Taking partial derivatives with respect to all α_i (except α_1 which is 1) and λ_i, we obtain a system of equations as follows:

$$\begin{gathered}
\tfrac{\partial}{\partial\alpha_2}(F) = 0, \\
\tfrac{\partial}{\partial\alpha_3}(F) = 0, \\
\vdots \\
\tfrac{\partial}{\partial\alpha_{(8-n)}}(F) = 0, \\
\tfrac{\partial}{\partial\lambda_1}(F) = 0, \\
\tfrac{\partial}{\partial\lambda_2}(F) = 0, \\
\vdots \\
\tfrac{\partial}{\partial\lambda_{(m+2)}}(F) = 0.
\end{gathered} \tag{2.30}$$

for the case when inequality constraints are present, solutions also need to satisfy Karush-Kuhn-Tucker condition [55] given by,

$$\mu \frac{\partial}{\partial \mu}(F) = 0. \tag{2.31}$$

while conditions for feasibility and optimality are respectively given by Eq. (2.32) and Eq. (2.33) ,

$$g \leq 0, \tag{2.32}$$

$$\mu \geq 0. \tag{2.33}$$

The system of polynomial equations comprised of Eq. (2.30) and Eq. (2.31) is solved by computation of Groebner basis followed by eigen-system methods to extract numerical roots. This is achieved using Wolfram Mathematica's NSolve [56] routine. Solutions obtained are subjected to feasibility test using Eq. (2.32) to obtain feasible solutions, which give rise to a set of dyad vectors by substituting them into (2.21). Examining the signature of the dyad vector $\mathbf{p}$, we can determine the dyad type and using the inverse relations in (2.7), we can obtain the mechanism parameters. A pool of mechanical dyads is obtained from the set of dyad-vectors using inverse relations given in (2.7). Any two of these mechanical dyads can be combined to form a four-bar linkage as the solution for motion generation problem.

2.7 Examples

Now we present two examples, which demonstrate efficacy of our framework. We do not presume linkage types and determine best types and dimensions from the task requirements.

2.7.1 Optimal solution for Five positions with no exact solution

Table 2.1 contains five precision poses as the input to the classic Burmester problem. Unfortunately for this problem, no exact planar four-bar solutions of any

type exist. Thus, only approximate solutions can be computed by relaxing some of the constraints. Our previous approach in [1] fails to find exact or approximate solution for this problem because SVD can not find approximate solutions for fully constrained problem. The new algorithm presented in this chapter can deal with such problems. If the first and last poses are critical, algorithm can treat in-between poses to be approximate. However, when one exact constraint is relaxed, solution space increases by ∞^1 along with number of optimization variables α_i. Thus, the recommended approach is to relax as few constraints as possible to keep the optimization computationally cheaper. Hence, third precision pose is relaxed.

Now redefined task is to find a four-bar that interpolates 4 precision poses exactly and approximates third pose as closely as possible. The sense of approximation here is termed as algebraic fitting error of pose and constraint manifold of each dyad. First step is the algebraic fitting of geometric constraints, which in this case are first two and last two poses. Therefore, matrix $[A]$ of size 4×8 is formed using (2.19) and four singular vectors corresponding to near-zero singular values are obtained by SVD which are tabulated in Table 2.2. None of these singular vectors correspond to a mechanical dyad as they do not satisfy conditions in (2.5).

The error function is formed using Eq. (2.8), which evaluates algebraic fitting error of third pose. For a pose, the algebraic error is given by Eq. (2.22). Thus, the error function is

$$f = 0.224(-1 + 0.675\alpha_2 - 0.369\alpha_3 + 2.299\alpha_4). \tag{2.34}$$

Since there are no nonlinear geometric constraints, f is subjected to two equality

constraints h_1 and h_2 given by,

$$\begin{aligned} h_1 &= 0.20 - 0.373\alpha_2 - 0.045\alpha_3 + 0.46\alpha_2\alpha_4 \\ &\quad +0.13\alpha_3\alpha_4 - 0.33\alpha_4 + 0.15\alpha_2^2 + 0.011\alpha_3^2 \\ &\quad -0.0095\alpha_2\alpha_3 - 0.16\alpha_4^2, \end{aligned} \tag{2.35}$$

$$\begin{aligned} h_2 &= -0.13 + 0.31\alpha_2 + 0.10\alpha_3 - 0.44\alpha_4 \\ &\quad -0.22\alpha_2^2 - 0.013\alpha_3^2 + 0.39\alpha_2\alpha_4 - 0.079\alpha_2\alpha_3 \\ &\quad -0.11\alpha_3\alpha_4 + 0.0067\alpha_4^2, \end{aligned} \tag{2.36}$$

Using steps presented in section 2.6.2, we form F given by

$$F = -f^2 - \lambda_1 h_1 - \lambda_2 h_2 \tag{2.37}$$

and system of equations by partial differentiation given by,

$$\begin{aligned} \tfrac{\partial}{\partial\alpha_2}(F) &= \alpha_2(-0.31\lambda_1 + 0.43\lambda_2 - 0.046) + \alpha_3(0.0095\lambda_1 \\ &\quad +0.079\lambda_2 + 0.025) - 0.46\alpha_4\lambda_1 - 0.39\alpha_4\lambda_2 \\ &\quad -0.16\alpha_4 + 0.37\lambda_1 - 0.31\lambda_2 + 0.068 = 0, \end{aligned} \tag{2.38}$$

$$\begin{aligned} \tfrac{\partial}{\partial\alpha_3}(F) &= \alpha_2(0.0095\lambda_1 + 0.079\lambda_2 + 0.025) + \alpha_3(-0.021\lambda_1 \\ &\quad +0.026\lambda_2 - 0.014) - 0.13\alpha_4\lambda_1 + 0.10\alpha_4\lambda_2 \\ &\quad +0.086\alpha_4 + 0.045\lambda_1 - 0.10\lambda_2 - 0.037 = 0, \end{aligned} \tag{2.39}$$

$$\begin{aligned} \tfrac{\partial}{\partial\alpha_4}(F) &= -0.32\lambda_1\alpha_4 - 0.013\lambda_2\alpha_4 - 0.53\alpha_4 + 0.33\lambda_1 \\ &\quad +\alpha_2(-0.46\lambda_1 - 0.39\lambda_2 - 0.16) + \alpha_3(-0.13\lambda_1 \\ &\quad +0.10\lambda_2 + 0.086) + 0.44\lambda_2 + 0.23 = 0, \end{aligned} \tag{2.40}$$

$$\begin{aligned} \tfrac{\partial}{\partial\lambda_1}(F) &= -0.20 + 0.37\alpha_2 + 0.044\alpha_3 + 0.33\alpha_4 \\ &\quad -0.15\alpha_2^2 - 0.011\alpha_3^2 - 0.46\alpha_2\alpha_4 \\ &\quad +0.0094\alpha_2\alpha_3 - 0.13\alpha_3\alpha_4 + 0.16\alpha_4^2 = 0, \end{aligned} \tag{2.41}$$

$$\begin{aligned} \tfrac{\partial}{\partial\lambda_2}(F) &= 0.13 - 0.31\alpha_2 - 0.10\alpha_3 + 0.44\alpha_4 \\ &\quad +0.22\alpha_2^2 + 0.012\alpha_3^2 - 0.39\alpha_2\alpha_4 \\ &\quad +0.078\alpha_2\alpha_3 + 0.10\alpha_3\alpha_4 - 0.01\alpha_4^2 = 0 \end{aligned} \tag{2.42}$$

Table 2.1: Example 2.7.1: Pose Data

Poses	X	Y	ϕ (degree)
Pose 1	-5.74803	-0.00787402	88.5679
Pose 2	-4.12598	0.795276	2.16642
Pose 3	-2.72441	1.67717	356.968
Pose 4	-1.54331	0.433071	1.03102
Pose 5	1.22835	-0.590551	345.624

Solving this system of equations using Mathematica's NSolve routine produces two sets of real solutions given by,

$$\begin{aligned} &\alpha_2 = 0.502, \quad \alpha_3 = -1.79, \quad \alpha_4 = -2.47, \\ &\alpha_2 = 1.20, \quad \alpha_3 = -0.505, \quad \alpha_4 = 0.997. \end{aligned} \tag{2.43}$$

Dyad-vectors are computed by substituting these solutions in (2.21). Table 2.3 contains these two dyad vectors. The four-bar linkages obtained by assembling these two dyads with coupler is shown in Fig. 2.1, where we can clearly see that coupler approximates third pose while interpolating remaining poses.

Table 2.2: Example 2.7.1: Four singular vectors obtained after SVD of the matrix $[A]$ of size 4×8.

Dyad Vector	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
$\mathbf{p}_1$	-0.018	-0.251	0.517	-0.398	0.0223	0.0361	0.498	0.509
$\mathbf{p}_2$	0.00770	0.428	-0.323	0.504	-0.0222	-0.0626	0.482	0.468
$\mathbf{p}_3$	0.0287	0.0807	0.111	0.148	-0.012	0.977	-0.0397	0.0138
$\mathbf{p}_4$	0.0756	-0.162	0.024	0.190	0.964	-0.00903	-0.00775	0.0122

2.7.2 Optimal Linkage for Four Precision Poses with Region Constraint

Figure 2.2 shows five positions of a landing gear moving from the landing position to the retracted position. Table 2.4 contains position and orientation data for five poses. It is desirable that fixed pivots should lie inside the circle of radius 2.3 with center located at (3.33, 2.04). The task is to synthesize a mechanism which

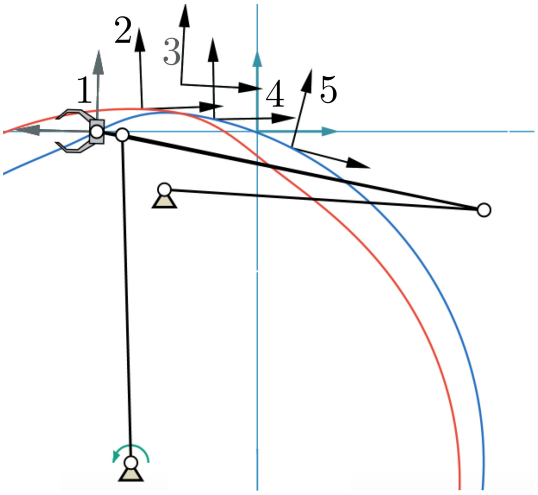

Figure 2.1: Example 2.7.1: Optimal four-bar mechanism that minimizes algebraic fitting error for third pose. Although second pose lies on different circuit, this Grashof type four-bar produces desired continuous motion from first to last pose.

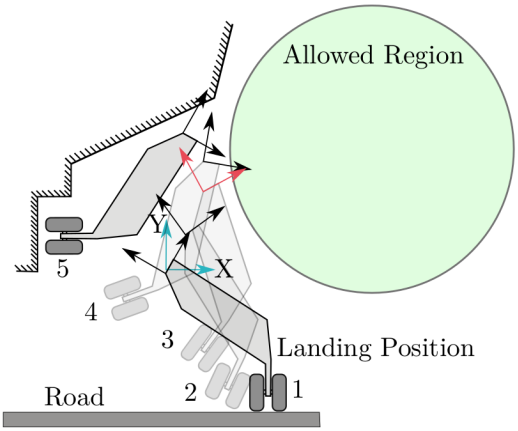

Figure 2.2: Example 2.7.2: Five landing gear positions are shown where the third position can be relaxed. Allowed region for fixed pivots of mechanism is also shown.

Table 2.3: Example 2.7.1: Two optimum dyad-vectors obtained as result of optimization

Vector	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
$\mathbf{s}_1$	0.4175	1.025	5.818	1.388	0.8691	17.21	7.760	8.773
$\mathbf{s}_2$	0.08080	0.007451	0.07404	0.3656	0.9573	0.2468	0.4554	0.4875

interpolates through precision poses (1,2,4,5) and minimizes the algebraic error for the third pose while keeping fixed pivot locations inside the allowed region as shown in the figure.

First step is to extract all four geometric constraints, i.e. four precision poses and form matrix $[A]$ using Eq. (2.19). Here $n = 4$ which means solution space consists of 4 singular vectors which are obtained using SVD and tabulated in Table 2.5. Once this linear algebraic fitting is done, optimization problem can be formulated.

The error function is linear error function f_1 for third pose, which is evaluated using Eq. (2.22). Substituting singular vectors into dyad coefficients followed by substituting them in terms of α_i using Eq. (2.21), we get final objective function given by

$$f = f_1^2, \tag{2.44}$$

where $f_1 = -0.0598\alpha_2 + 0.0294\alpha_3 - 0.100\alpha_4 + 0.0876$. The circular region for fixed

Table 2.4: Example 2.7.2: Pose Data

Poses	X	Y	ϕ (degree)
Pose 1	-0.0125	-0.0374	66.3
Pose 2	0.303	0.634	35.5
Pose 3	0.599	1.83	352.
Pose 4	0.268	2.30	331.
Pose 5	0.606	1.31	22.2

Table 2.5: Example 2.7.2: Four singular vectors obtained after SVD of the matrix $[A]$ of size 4×8. The vectors form basis for the null space

Vector	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
$\mathbf{p}_1$	-0.585	-0.0211	-0.506	0.165	0.134	-0.350	0.368	0.313
$\mathbf{p}_2$	0.0640	-0.330	0.190	-0.804	0.236	-0.264	0.194	0.204
$\mathbf{p}_3$	-0.280	0.0484	-0.466	-0.398	0.232	0.603	-0.136	-0.329
$\mathbf{p}_4$	-0.137	0.145	0.469	0.250	0.747	0.229	0.259	0.0111

pivots is modeled as an inequality constraint using Eq. (2.25) and (2.26) given by,

$$\begin{aligned} g = 0.091{\alpha_2}^2 + (0.25\alpha_3 + 0.11\alpha_4 + 0.25)\alpha_2 \\ +0.35{\alpha_3}^2 + 0.049{\alpha_4}^2 + \alpha_3(0.043\alpha_4 + 1.0) \\ -0.048\alpha_4 - 0.19 \leq 0 \end{aligned} \tag{2.45}$$

Objective function also has two quadratic equality constraints given by,

$$\begin{aligned} h_1 = 0.058{\alpha_2}^2 + (-0.25\alpha_3 + 0.17\alpha_4 - 0.36)\alpha_2 - 0.34{\alpha_3}^2 \\ -0.041{\alpha_4}^2 + \alpha_3(0.23\alpha_4 - 0.38) - 0.033\alpha_4 + 0.29, \\ h_2 = -0.29{\alpha_2}^2 + \alpha_2(-0.15\alpha_3 - 0.074\alpha_4 - 0.048) + 0.20{\alpha_3}^2 \\ +\alpha_3(0.18\alpha_4 + 0.12) - 0.46{\alpha_4}^2 - 0.11\alpha_4 - 0.36 \end{aligned} \tag{2.46}$$

We follow steps presented in section 2.6.2 and form Lagrange objective function F given by,

$$F = -{f_1}^2 - \lambda_1 h_1 - \lambda_2 h_2 - \mu g \tag{2.47}$$

and obtain equations by partial differentiation as well as equation corresponding to

Table 2.6: Example 2.7.2: Four optimum dyad-vectors obtained as result of optimization

Vector	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
$\mathbf{s}_1$	0.0254	0.192	-0.202	-0.0965	0.00562	0.723	-0.387	-0.490
$\mathbf{s}_2$	0.276	-0.406	0.211	-0.425	-0.161	-0.561	0.251	0.360
$\mathbf{s}_3$	0.315	-0.356	0.189	-0.397	-0.318	-0.598	0.129	0.324
$\mathbf{s}_4$	0.208	-0.302	0.141	-0.335	-0.321	-0.691	0.134	0.368

Karush-Kuhn-Tucker condition as follow:

$$\begin{aligned}\frac{\partial}{\partial\alpha_2}(F) = 0 = (-0.12\alpha_2 + 0.25\alpha_3 - 0.17\alpha_4 + 0.36)\lambda_1 + (0.57\alpha_2 \\ +0.15\alpha_3 + 0.074\alpha_4 + 0.048)\lambda_2 - (0.18\alpha_2)\mu \\ -(0.25\alpha_3)\mu - (0.11\alpha_4)\mu - 0.25\mu + 0.060\end{aligned} \tag{2.48}$$

$$\begin{aligned}\frac{\partial}{\partial\alpha_3}(F) = 0 = \lambda_1(0.25\alpha_2 + 0.69\alpha_3 - 0.23\alpha_4 + 0.38) + \lambda_2 \\ (0.15\alpha_2 - 0.41\alpha_3 - 0.18\alpha_4 - 0.12) - 0.25\alpha_2\mu \\ -0.71\alpha_3\mu - 0.043\alpha_4\mu - 1.0\mu - 0.029\end{aligned} \tag{2.49}$$

$$\begin{aligned}\frac{\partial}{\partial\alpha_4}(F) = 0 = \lambda_1(-0.17\alpha_2 - 0.23\alpha_3 + 0.081\alpha_4 + 0.033) \\ +\lambda_2(0.074\alpha_2 - 0.18\alpha_3 + 0.92\alpha_4 + 0.11) - 0.11\alpha_2\mu \\ -0.043\alpha_3\mu - 0.097\alpha_4\mu + 0.048\mu + 0.10\end{aligned} \tag{2.50}$$

$$\begin{aligned}\frac{\partial}{\partial\lambda_1}(F) = 0 = -0.058\alpha_2^2 + (0.25\alpha_3 - 0.17\alpha_4 + 0.36)\alpha_2 \\ +0.34\alpha_3^2 + 0.041\alpha_4^2 + \alpha_3(0.38 - 0.23\alpha_4) + 0.033\alpha_4 - 0.29\end{aligned} \tag{2.51}$$

$$\begin{aligned}\frac{\partial}{\partial\lambda_2}(F) = 0 = 0.29\alpha_2^2 + (0.15\alpha_3 + 0.074\alpha_4 + 0.048)\alpha_2 \\ -0.20\alpha_3^2 + 0.46\alpha_4^2 + \alpha_3(-0.18\alpha_4 - 0.12) + 0.11\alpha_4 + 0.36\end{aligned} \tag{2.52}$$

$$\begin{aligned}\mu\frac{\partial}{\partial\mu}(F) = 0 = \mu(-0.091\alpha_2^2 + (-0.25\alpha_3 - 0.11\alpha_4 - 0.25)\alpha_2 \\ -0.35\alpha_3^2 - 0.049\alpha_4^2 + \alpha_3(-0.043\alpha_4 - 1.0) + 0.048\alpha_4 + 0.19)\end{aligned} \tag{2.53}$$

Solving these equations followed by filtering on the basis of feasibility using Eq. 2.32 yields four unique and feasible solutions tabulated in Table 2.7. All of these solutions satisfy Karush-Kuhn-Tucker Condition for optimality given by Eq. (2.33). Dyad vectors are calculated by substituting these solutions into (2.21) and are given in Table 2.6. Any of these four dyad-vectors when substituted in Eq. (2.4) forms a quartic equation, which when projected on hyperplane $Z_4 = 1$ represents a quadric surface. Fig. 2.4 shows intersection of hyperboloid and hyperbolic paraboloid formed from first and second dyad-vectors. The intersection curve represents workspace of the corresponding four-bar linkage. Table 2.8 contains the minimized algebraic fitting

Table 2.7: Real Solutions for α_i, λ_i and μ

Dyad	α_1	α_2	α_3	α_4	λ_1	λ_2	μ
$\mathbf{s}_1$	1	-0.40	-0.027	1.09	0.0	0.0	0.0
$\mathbf{s}_2$	1	-2.04	-1.81	2.06	0.0	0.0	0.0
$\mathbf{s}_3$	1	-3.64	-4.24	5.37	0.0	0.0	0.0
$\mathbf{s}_4$	1	5.08	-2.58	-2.94	0.0	0.0	0.0

error of objective function. From this table, we can see that dyad $\mathbf{s}_4$ has least pose fitting error. All dyads except $\mathbf{s}_1$ are of RR type dyads while $\mathbf{s}_1$ is an RP dyad. Figure 2.3 shows a branch defect free four-bar mechanism formed by combining $\mathbf{s}_1$ and $\mathbf{s}_2$.

2.8 Conclusion

In this chapter, we presented a task-driven approach to unified and optimal synthesis of planar four-bar linkages for extended Burmester problem. In this formulation, various geometric constraints are treated equivalently, which in turn leads to a much simpler two-step based algorithm for computing planar dyads of four-bar linkages. Original contributions of this chapter have been into reforming a mixed exact-approximate algebraic fitting problem into problem of task oriented optimal fitting of algebraic manifold. The framework presented here can accommodate linear as well as non-linear equality and inequality geometric constraints and minimize objective functions that can be expressed in terms of dyadic parameters. Although adding non-linear geometric constraints increase computational complexity, computer algebra software like Mathematica could be used to compute solutions of quadratic

Table 2.8: Optimality Evaluations for Dyads

Dyad	3^{rd}Pose-Fitting Error	Inequality Constraint
$\mathbf{s}_1$	0.0102	-0.000650
$\mathbf{s}_2$	0.0415	-0.00126
$\mathbf{s}_3$	0.0212	-0.000424
$\mathbf{s}_4$	-1.8×10^{-8}	-0.396

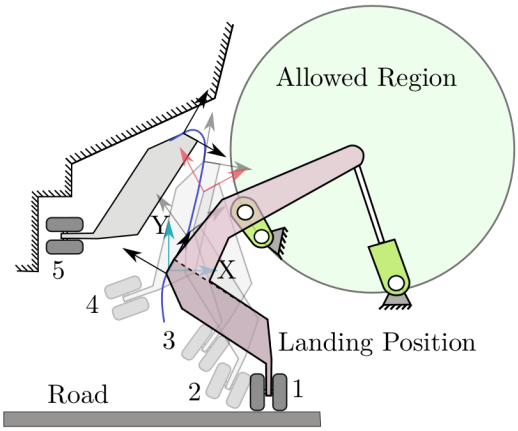

Figure 2.3: Example 2.7.2: First and second dyad in Table 2.6 are combined to form the linkage shown. It can be clearly seen that coupler curve fairly approximates the third pose while fixed pivots are inside the allowed region.

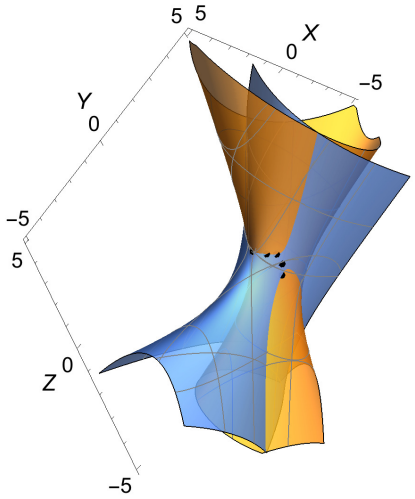

Figure 2.4: Example 2.7.2: Image-Space representation of intersection of third and fourth optimal constraint manifolds from Table. 2.6. Also shown are five image points as dark spheres representing the five poses; four of them lie exactly on the intersection of the two surfaces, while one is closest possible.

system of equations in a reasonable amount of time. Experimentations show that Mathematica takes less than 3 seconds on a MacBook Pro with 2.4GHz Intel core i5 processor and 8GB RAM for computing solutions for the system of seven quadratic equations. The framework also preserves previously achieved real-time solutions for linear geometric constraints with no optimality criterion. Two examples demonstrating computation of optimal type and dimensions of dyads that minimize task oriented objective function are presented.

This work has been published in ASME Journal of Mechanisms and Robotics, 2017 [2].

Chapter 3

Defect-Free Kinematic Synthesis by Partial Shape Matching in a Clustered Database

3.1 Introduction

A large majority of mechanism synthesis methods are based on the precision position approach. This approach is a clever approximation trick to solve the above-mentioned problems, where the task is discretized into precision positions. These positions instead of the actual task, are required to be interpolated or approximated by the designed mechanism. However, this representation loses critical information about the actual continuous task and can lead to mechanisms with the order, circuit and branch defects; see Chase and Mirth[4] for a thorough discussion on such defects. Unfortunately, these defects render mechanisms useless for their intended application. For an example, consider a motion generation problem shown in Fig. 3.1, where the objective is to synthesize a four-bar mechanism that can perform the prescribed motion going continuously from position 1 to 5. Instead of dealing with an infinite number of positions from initial to final one, currently this problem is simplified to design a four-bar that goes through all the five positions without any guarantee on the in-between motion. Burmester [40] showed that a four-bar can go through at most five precision positions and even in the best case scenario, there are a limited number of solutions. In this case, only one solution is obtained as shown in the Fig. 3.1. Although it can be seen that the coupler of the four-bar passes exactly through five precision positions, it can not do so without changing the circuit. A circuit represents

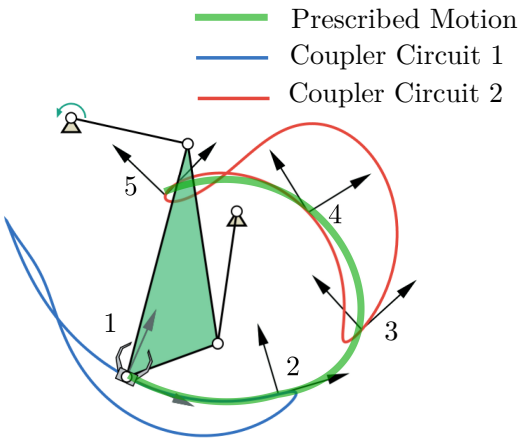

Figure 3.1: The four-bar mechanism obtained using the precision position approach suffers from circuit defect, as no coupler circuit passes through all precision positions.

an assembly mode in which the mechanism is put together and to transition from one circuit to another, the mechanism has to be taken apart and reassembled. This phenomenon is called circuit defect in the linkage, which makes the linkage useless for the prescribed task. To deal with it, an approach proposed in the literature is to tweak precision positions in a brute-force way within some tolerance until a solution is found. Even if a circuit defect-free solution is found, the coupler motion in-between the precision points may go through undesired poses or in an incorrect order. This is an outcome of discarding the functional aspect of continuous motion and turning the problem into an interpolation problem bereft of important details.

Instead of brute-force search within tolerance regions, some approaches apply separate constraints and form an optimization problem of non-differentiable objective function. These methods employ metaheuristic algorithms like Differential-evolution (DE), Particle Swarm Optimization (PSO), Cuckoo Search (CS). Cabrera et al.[53] used Genetic Algorithm for optimization in mechanism synthesis. Sardashti et al.[57] used PSO towards the defect-free synthesis of four-bar linkage with joint clearance for path generation problem. Ebrahimi and Payvandy[58] presented an application of Imperialist Competitive Algorithm (ICA) for synthesizing path generating four-bars having desired workspace limits. Bulatovic[59] used Cuckoo Search for solving the problem of optimum synthesis of a six-bar double dwell linkage.

Path synthesis methods based on Fourier analysis do take the continuity information of coupler path into account. However, most of them are defined only for closed loop curves. Ullah and Kota[5] 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.[6] presented a method based on finite Fourier series for open and closed path generation of four-bar mechanisms. In the case of motion generation, Li

et al.[7] have developed a Fourier descriptor-based approach for approximate motion generation. Buśkiewicz et al.[60] used the curvature of the coupler curve for path synthesis using Genetic Algorithms. Khan et al.[9] presented on approach where an artificial neural network is used for mapping between Fourier coefficients corresponding to a coupler path and corresponding linkage parameters.

Instead of the global search, an alternative approach is to start from a good initial guess based on an atlas and use local search methods. We adopt this approach and combine with our novel formulation to generate a diverse set of conceptual design solutions. McGarva[61] took the earliest approach towards creating a library for coupler trajectories based on the harmonic analysis. Wandling[62] 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.[63] 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 the input. Chu et al.[64] 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 the highly nonlinear mapping between linkage parameters and coupler trajectory, this way of sampling leads to a nonuniform sampling of trajectory space, which causes under-representation of possible motions. We address this issue by employing log-normal distribution in the linkage parameter space to generate the data samples. Then, we perform a 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 data points called cluster centers, which are representative of the cluster points in lower levels. Wandling [62] and

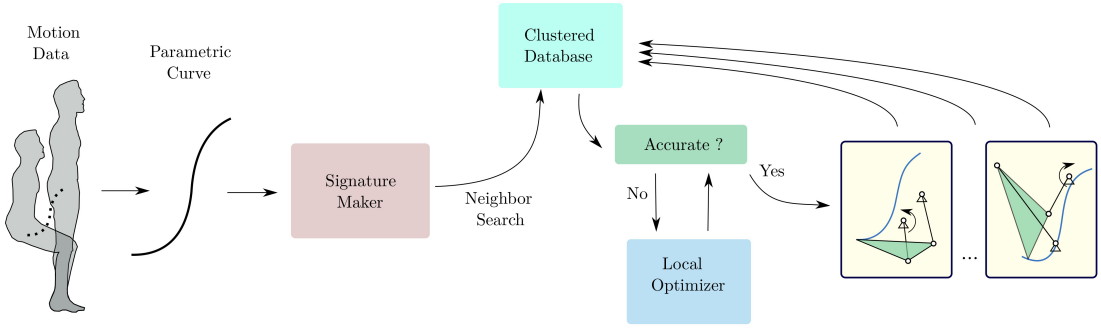

Figure 3.2: The Machine Learning approach begins by creating an invariant signature for the path and the motion data, which facilitates a compact and hierarchical clustered database and an auto-encoder Neural Network trained to elicit good, defect-free solutions or subjected to local, fast optimization. The results are defect-free conceptual design solutions for input problems.

Yu et al. [63] 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 that represents all the segments in it. This is because our representation facilitates part-to-whole 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.

In this chapter, we represent the given task as a parametric continuous function of poses or path-points. The objective is to find a linkage which has a coupler motion or path compatible with the given task. We develop a compatibility measure invariant to similarity transformations so that position, scaling, and orientation of the given path or motion do not convolute the optimization. Next step is to conduct a search in the space of linkage parameters to find linkages with coupler motions compatible with the prescribed task. Although global search methods can be applied for finding solutions, we employ an efficiently clustered database and Powell's local search method to come up with a variety of different solutions. The motivation behind using a clustered database is rooted in the broader objectives of machine design. Mechanism synthesis is a critical part of the conceptual design phase, which requires synthesis method to be prolific in terms of concept generation to 1) realize the potential of attainable design possibilities, and 2) have the agility to adapt a design to evolving requirements. Our method only deals with the coupler curves and is not dependent on the linkage type. Thus, it is readily scalable to any type of planar linkage.

The synthesis routine starts by creating a continuous parametric representation of a prescribed path or motion. We employ pattern recognition and computational shape analysis to create an invariant signature for the prescribed path and motion. A query representing an invariant signature of the prescribed path or motion 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. The objective function that drives the synthesis process computes a distance measure of dissimilarity between the task and the coupler motion or path 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. Figure 3.2 illustrates an overview of our method, which is codified in Algorithm 1.

The original contributions of the work are in 1) creating a perceptive problem formulation for path and motion generation, which solves the issues associated with the precision position approach, 2) exploiting the nonlinear nature of the relationship between the linkage parameters and coupler motions to create a sensitive, wide-ranging, compact, and efficient database with hierarchical clustering, and 3) developing a novel algorithm for partial matching of motions and paths which significantly improves the synthesis.

Rest of this chapter is organized as follows. Section 3.2 presents the computation of motion and path signatures. Section 3.3 is comprised of evaluation criterion for signatures based on the shape similarity, which leads to the formulation of error function for optimization. Section 3.4 discusses the nature of objective function via sensitivity analysis at a singularity. Section 3.5 presents the database generation and clustering using auto-encoders for efficient sampling and query operations. Finally, two case studies are presented in section 3.6 to illustrate the efficiency and efficacy of the method.

3.2 Signatures of Coupler Path and Motion

Focus of the chapter 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

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**, ...**

```
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
```

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 the path signature, we use the formulation developed by Cui et.al[65].

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 $\theta(t)$ 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,

$$\kappa(t) = \frac{\ddot{y}(t)\dot{x}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{(\frac{3}{2})}}, \tag{3.1}$$

$$K(t) = \int_0^t |\kappa(t)| dt, \tag{3.2}$$

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 3.3. We compute $\kappa(t)$ and $K(t)$ along the direction of t using Eq. (3.1) and Eq. (3.2), respectively. Figure 3.4 shows computed curvature and its unsigned integral for the coupler path shown in the Fig. 3.3. It can be seen that the curvature is small at the start, increases as the curve bends along the path and drops once again as the path straightens out. It is obvious that the 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 3.5(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. Although it may seem natural to use planar quaternions [26, 25] for representing motion and finding its signature, it

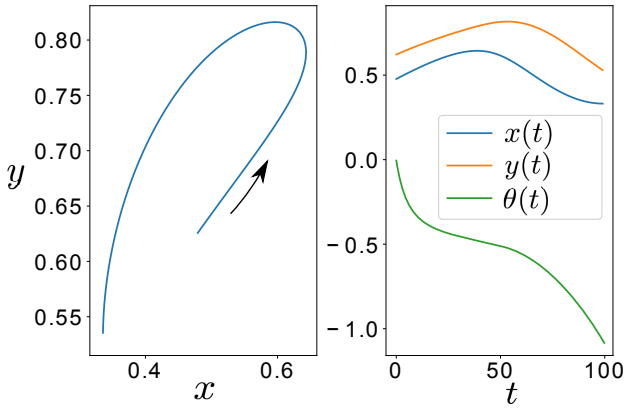

Figure 3.3: a) The path of the input motion along with direction of parametrization, b) motion components $x(t), y(t), \theta(t)$ are plotted against parameter t.

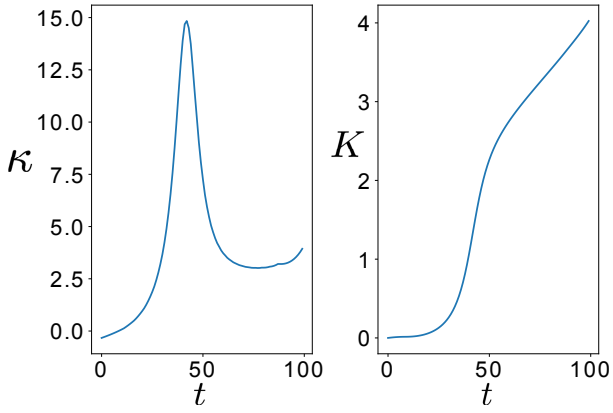

Figure 3.4: Curvature and its unsigned integral for the path shown in Fig. 3.3

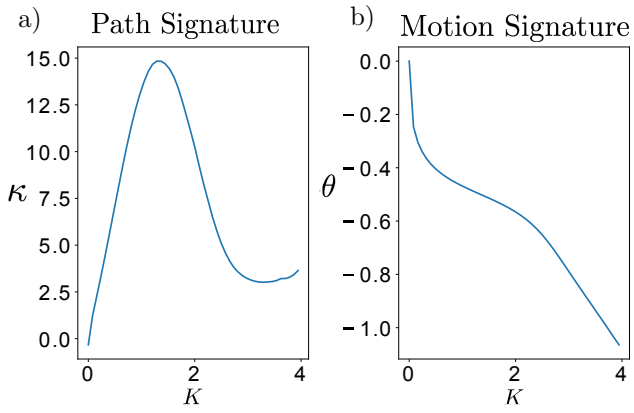

Figure 3.5: Path and motion signatures of the motion shown in Fig. 3.3

couples orientation with the path in such a way that the signatures formed no longer remain invariant to similarity transformation.

These signatures are invariant under similarity transformations; for proof see[65]. 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 produces flipped path signature, but motion signature remains invariant. Figure 3.5 shows the path and motion signature obtained for the motion depicted in Fig. 3.3. 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 the Algorithm 2.

3.3 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 an 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. 3.1
$K(t)$ = IntergrateCumulatively$(\kappa(t))$ using Eq. 3.2
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$(\theta(tmp))$
    pathSignature.push$(\kappa(tmp))$
end
return PathSignature, MotionSignature

3.3.1 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 Fig. 3.6. Let p and W be their signatures respectively, where W completely contains p as shown in Fig. 3.7. The orientation information shown in Fig. 3.6 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 slided with offset index j along W.

(3) For each offset j, we compute normalized cross-correlation function [66] given by,

$$Cn(j,p,W) = |\sum_{i}^{p_{sp}} \frac{(W(i+j) - \bar{W}(j:j+p_{sp}))(p(i)-\bar{p})}{\sqrt{\sum_{i}^{p_{sp}} \left(W(i+j) - \bar{W}_{p_{sp}}\right)^2 \sum_{i}^{p_{sp}} \left(p(i)-\bar{p}\right)^2}}|, \tag{3.3}$$

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

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 of p inside W, i.e., **Part** is identical to a portion of **Whole**.

The 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 starting 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,

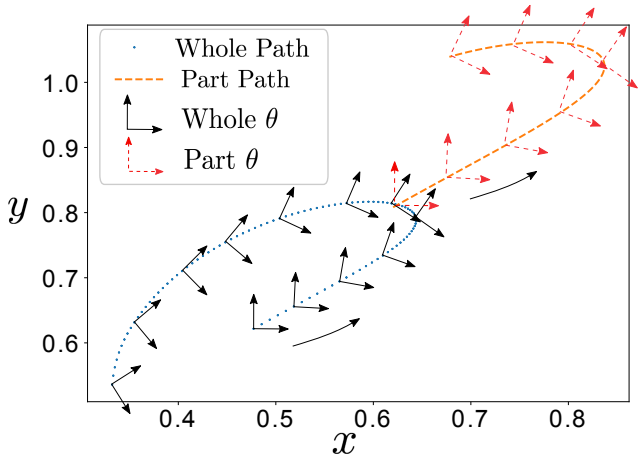

Figure 3.6: **Part** path is formed by trimming whole path followed by translation and scaling. Arrows indicate the increasing direction of parameter t.

offset index and the matching direction of sampling. Figure 3.8 depicts normalized cross correlation function over the sliding domain j for **Part** and **Whole** curves.

3.3.2 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. 3.6. Let p and W be the motion signatures of the **Part** and **Whole** motions, respectively as shown in Fig. 3.9. Similar to partial matching of path signatures, the cross-correlation function is given by,

$$E(j, p, W) = \sum_{i}^{p_{sp}} \left((W(i+j) - \bar{W}(j : j + p_{sp})) - (p(i) - \bar{p}) \right)^2, \tag{3.4}$$

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 3.10 depicts dissimilarity function over the sliding domain j. In this case, as shown in Fig. 3.10, we find that the first point is the matching point,

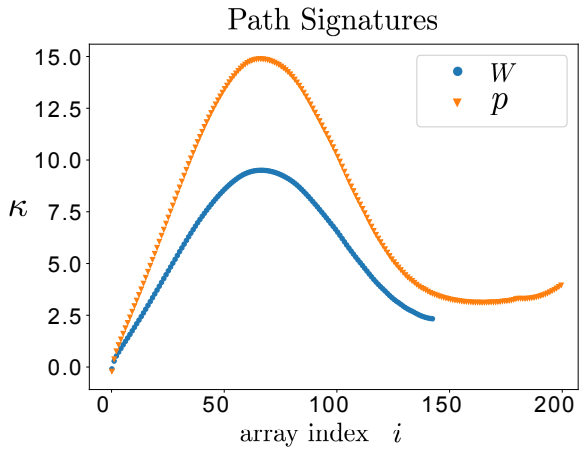

Figure 3.7: Path Signatures of part p and whole W from the Fig. 3.6. The array index i corresponds to the index location for the array of the K. The domain of path signature is scale-invariant but the range still has a scaling factor, which is taken care of by normalized cross-correlation.

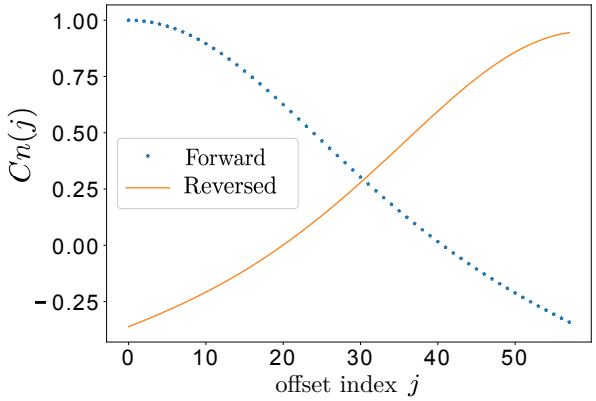

Figure 3.8: Normalized cross correlation of the signatures computed along each direction is shown. It can be seen that exact match is found at $j = 0$.

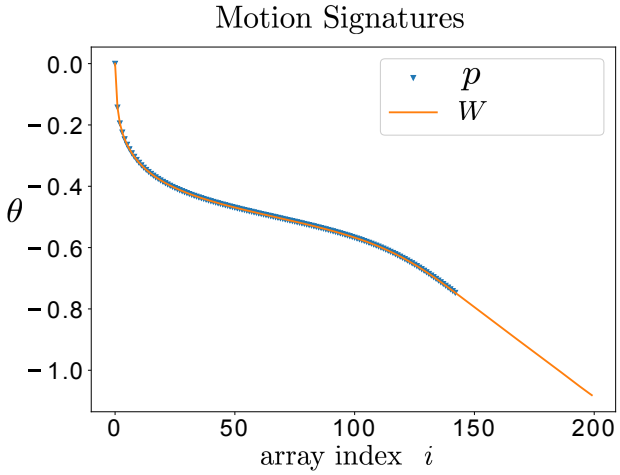

Figure 3.9: Motion Signatures of the trajectories shown in Fig. 3.6. The domain as well as range of motion signature is invariant to similarity transformation.

which is consistent with the fact that we have essentially sliced the whole motion to obtain the part motion.

3.3.3 Objective Function for Synthesis

The functions in Eq. (3.3) and (3.4) presented in the sections 3.3.1 and 3.3.2 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 synthesis problem as,

$$\arg\min_{\mathbf{l},W_i}(1 - Cn_{max}(p, W_i)), \tag{3.5}$$

where $\mathbf{l}$ 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 $\mathbf{l}$. In case of four-bar, $\mathbf{l} : l_1, l_2, l_3, l_4, l_5$, where l_i is link ratio of i^{th} link shown in Fig 3.11.

Similarly, we can formulate motion synthesis problem as,

$$\arg\min_{\mathbf{l},W_i}(E_{min}(p, W_i)). \tag{3.6}$$

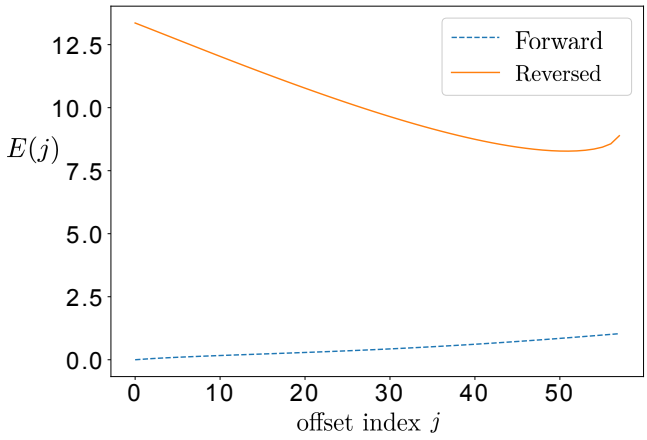

Figure 3.10: Dissimilarity function of two motion signatures along both directions. It can be seen that the exact match is found at $j = 0$, where the template is fully embedded inside the other motion.

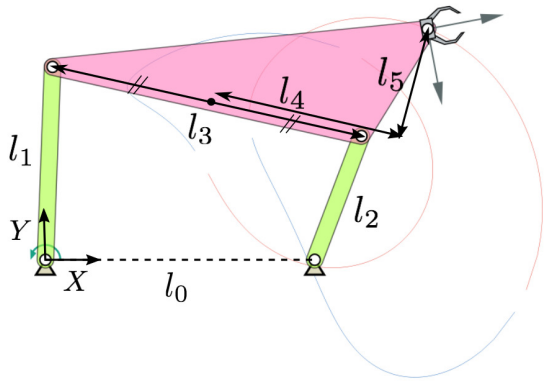

Figure 3.11: Parametric representation of four-bar linkage with all revolute joints. We set $l_0 = 1$ and one fixed joint at the origin of the global frame along with making the fixed length of four-bar parallel to the x-axis.

Here, E is the Dissimilarity function from Eq. (3.4) 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 3.2 reduces the number of 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 approach, such as Powell's method towards the end for faster convergence[5].

Considering the highly nonlinear nature of the problem, finding a good initial guess proves to be daunting. In addition, generating a large set of solutions requires a diverse and large number of good initial guesses. Thus, we exploit machine learning techniques to create a database for finding many good initial guesses or the solution itself. Section 3.5 presents the details of this approach.

3.4 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 such singular locations, which adversely affect the optimization process. In contrast to this behavior, the signatures derived in Sec. 3.2 have a smooth transition at these singular locations due to shape similarity between closed and just opened curve or motion.

To illustrate this via an example, we perform sensitivity analysis in the vicinity of a 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. The link ratios are chosen such that, small changes in some parameters lead to the topological change in the coupler curve. Error function between motions of new and initial four-bar are calculated using Eq. (3.6). Figure 3.12 shows coupler motion of some of the four-bars, while Fig. 3.14 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. 3.12 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. 3.13, 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.

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

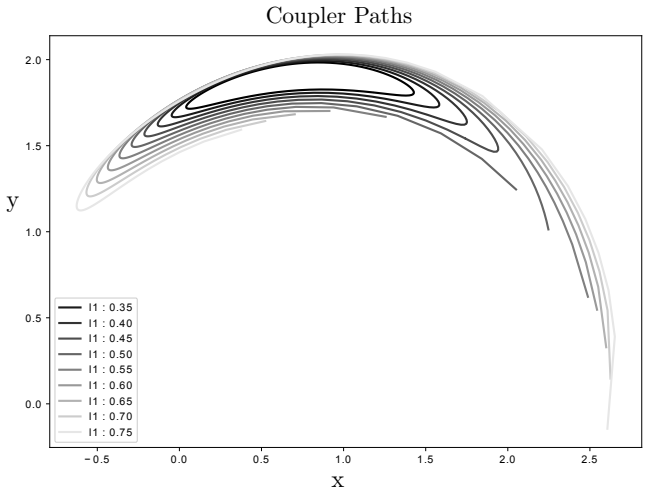

Figure 3.12: Coupler Motions of the four-bar linkage with variation of parameters l_1 and l_2. It can be seen that motion topology changes from close-loop Grashof to open loop Triple-Rocker.

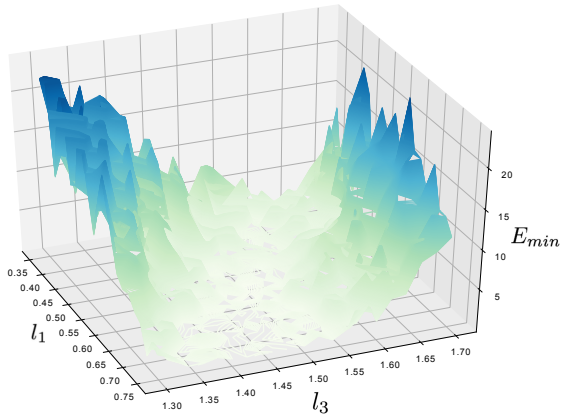

Figure 3.13: Distance (E_{min}) from Eq. (3.4) as the parameters l_1 and l_3 are varied. Although open loop breaks at l_1:0.55, l_3:1.5, there are no spikes of error function in the region near singularity, as the shape is very similar between the two topologies.

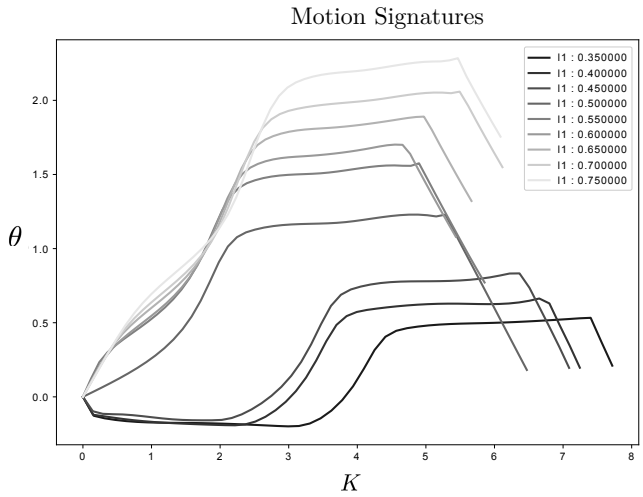

Figure 3.14: Motion signatures obtained by steps given in 3.2. Although topology difference is even more evident in this representation, it also signifies the similarity pattern between them.

the approach is the same for any planar motion generating mechanism. 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 3.11 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 nonlinear, 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, 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. 3.15, and Normal Distribution ($\mu = 0, \sigma = 2$) for (l_4, l_5).

We use machine learning techniques such as clustering and data-compression using auto-encoder neural networks to come up with good initial guesses for local optimization. First, We cluster the database using a hierarchical clustering algorithm. Then, we find a representative data point in each cluster called cluster centers and form their set. This set of cluster centers represent a diverse group of linkages. When a query is raised in the database, the first step is to search for neighbors in the set of cluster centers. This often yields a diverse set of neighbors and is used as the set of independent initial guesses for local optimization.

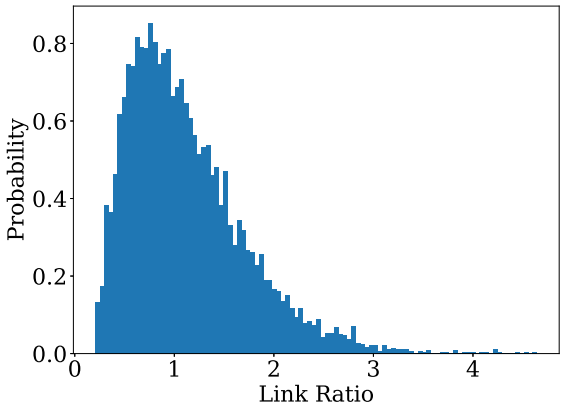

Figure 3.15: Probability Distribution function used in random sampling for parameters l_1, l_2 and l_3.

3.5.1 Dimensionality Reduction using Auto-Encoders

Each data point in the database consists of a 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**[67], 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[68]. This nonlinear mapping by auto-encoder can greatly improve the representation of data for clustering [69]. Figure 3.16 shows a Neural Network architecture similar to the one we designed for the task. Our architecture consists of 100 neurons in the input and output layer, while the 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,

$$h_i = ReLU(W_i h_{i-1} + b_i), \tag{3.7}$$

$$ReLU(x) = max(0, x), \tag{3.8}$$

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,

$$\arg\min_{W,b} \sum_{i=0}^{N} ||X_i - \tilde{X}_i||^2, \tag{3.9}$$

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[1] space for better clustering[69]. We use Agglomerative Clustering, a method of hierarchical clustering, which is an approach to partitioning clustering for identifying groups in the dataset. **Ward**[70] criterion is used for clustering, which minimizes the variance of the clusters being merged. The distance metric used for clustering is the Euclidean distance in the latent space. Although the more accurate distance metric is the distance function discussed in the section 3.3, 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.

Now, when the user raises a query, we use the distance function from the section 3.3 for finding k nearest neighbors among 1500 cluster centers. If a cluster center is not sufficiently close, we descend into its corresponding cluster to find the closest data point. Motion with highest similarity score is returned along with its corresponding linkage parameters. If required, the parameters are 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.

[1] compressed output of bottleneck layer.

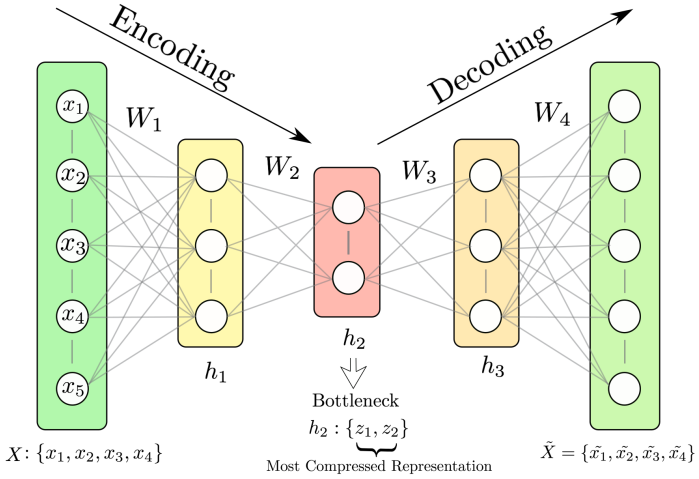

Figure 3.16: A small-scale version of the Auto-Encoder. This network takes 5-dimensional input in the input layer. At each encoder layer, the input is compressed into a vector of lower dimensions, the lowest at the bottleneck layer.

3.6 Case Studies

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

3.6.1 Path Generation

In the design phase of a rehabilitation device that assists people to stand from sitting position, it is required to generate linkages that can execute a sit-to-stand (STS) trajectory of the hip joint as shown in Fig. 3.17. Table 3.1 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. 3.17. 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 among cluster centers is $1-Cn_{max}$ in Eq. (3.5). Table 3.2 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 the offset index j in Eq. 3.3. Figure 3.18 shows the 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.

3.6.2 Motion Generation

The task is to find a pool of linkage systems that can perform snow-shoveling with a motion shown in Fig. 3.19. The motion data is tabulated in Table 3.3 to which we fit a B-spline with cord length parametrization in order to get the parametric

Table 3.1: 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 3.2: 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

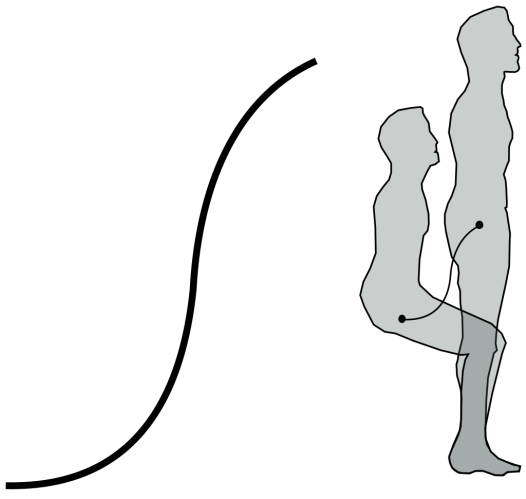

Figure 3.17: Case Study 1: Path traced by hip joint during Sit-to-Stand Motion.

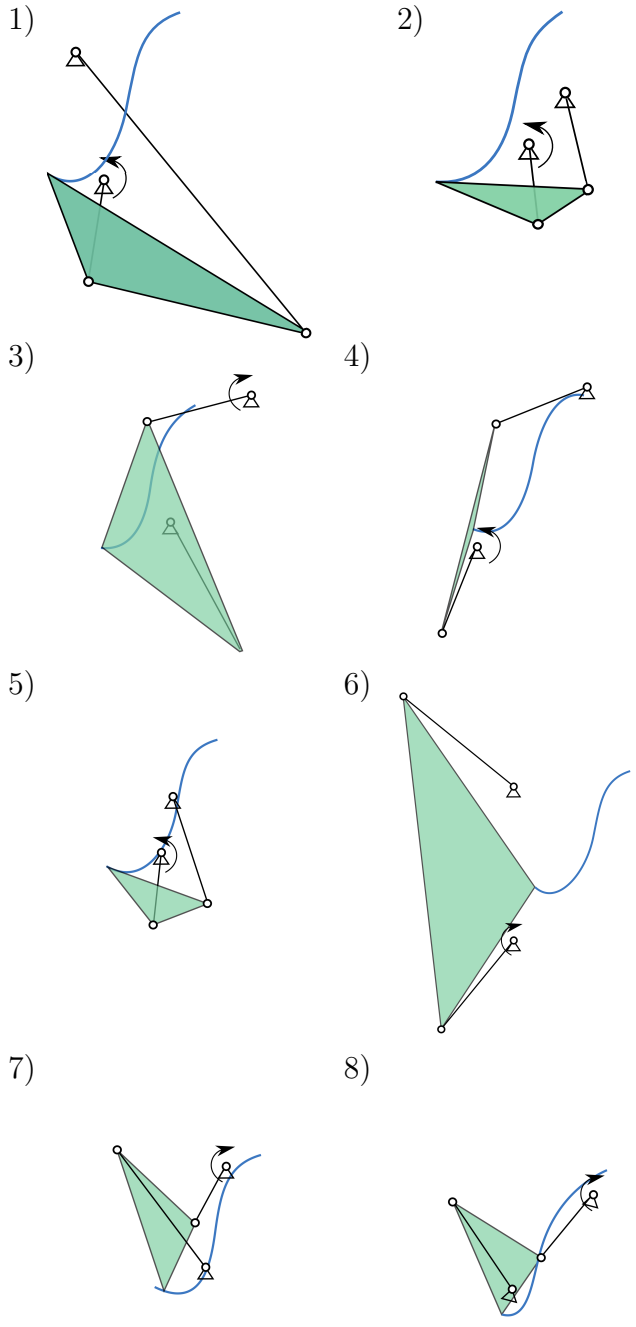

Figure 3.18: First eight linkages in the table 3.2 and their resultant coupler paths.

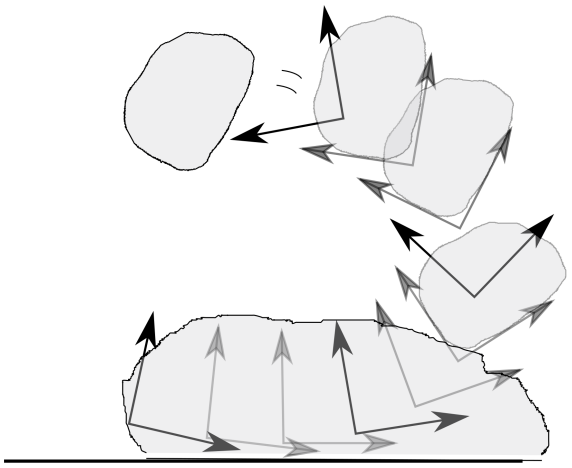

Figure 3.19: Case Study 2: User specified motion necessary for the snow shoveling task.

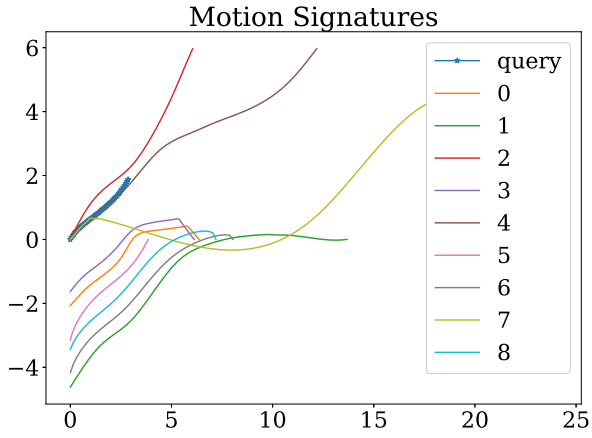

Figure 3.20: Case Study 2 - Query Result: Motion signatures in the dataset with highest similarity.

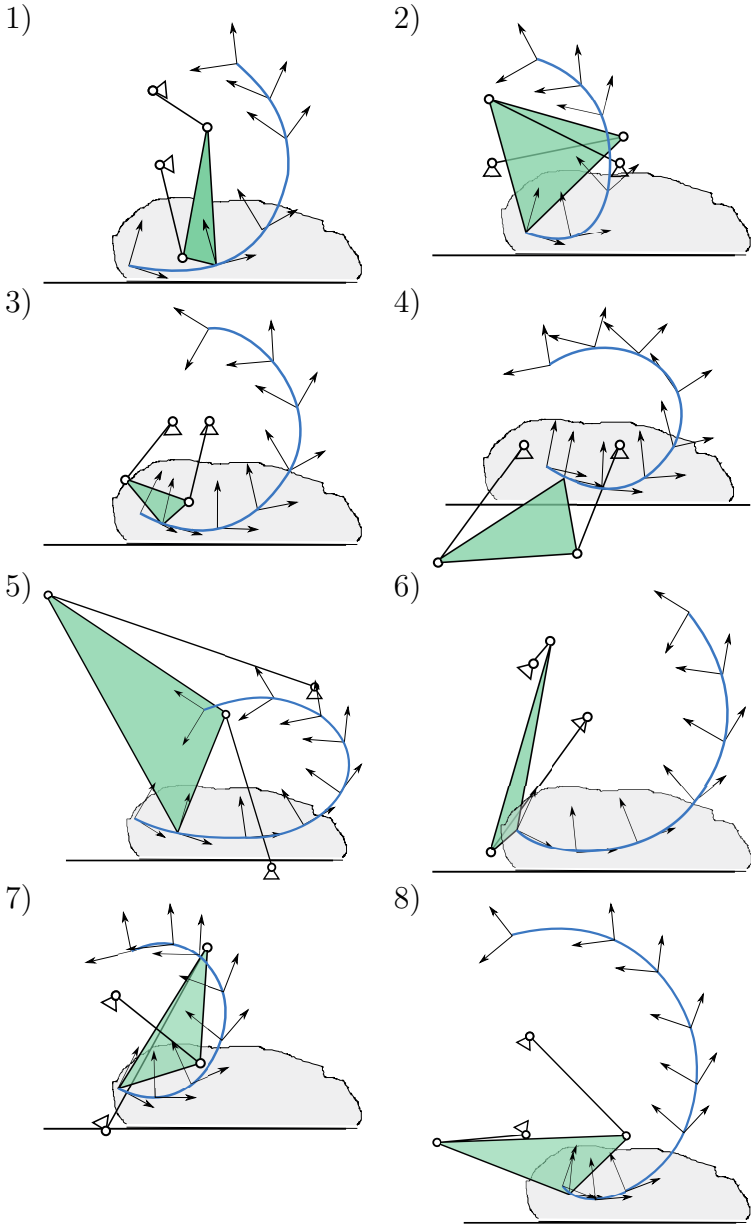

Figure 3.21: Case Study 2: First eight linkages in the table 3.4 and their resultant coupler motions.

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[1],[2] but obtained solutions suffer from circuit defect, which is not surprising as those methods do not account for the continuity of input positions. Also, It is obvious that the coupler should not drop the snow during its entire motion except at the end. Although the prescribed motion entails this information, precision point approach cannot capture it.

Now we employ the approach presented in this chapter. The first step is to calculate the motion signature of the task motion using steps mentioned in Algorithm 2. For that, we follow the steps given in section 3.2 to obtain the motion signature depicted in Fig. 3.20. Next, we raise the signature query for nearest signatures among cluster centers of the database. Fig. 3.20 shows nine nearest neighbor signatures along with the task signature. Table 3.4 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 the 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. (3.4). Figure 3.21 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. Although Fig. 3.21 shows that 5^{th} and 7^{th} solution may slightly interfere with the ground, it can be rectified by lifting the mechanism slightly up or making small changes in coupler dimensions. 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 the precision point approach.

Table 3.3: 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 3.4: 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

3.7 Conclusion

The methods based on precision point approach do not capture continuity of the task. This causes the solutions to have the branch, circuit and order defects. Also, the formulation fails to detect undesired properties of the coupler motion in the region between precision points. Thus, we present a perceptive problem formulation, by considering the entire prescribed task. We solve the proposed formulation by employing machine-learning techniques and generate a large number of defect-free conceptual designs. The approach is highly data-efficient due to similarity invariant representation and partial matching. Sensitivity analysis indicates that the complexity of the objective function is well behaved at the singular locations. The hierarchically clustered database provides an efficient query search. Finally, the effectiveness of the presented approach is showcased by two case studies. Every solution presented in the examples section is a result of part-to-whole matching. The other atlas-based approaches facilitate only whole-to-whole matching, hence they would need a very large amount of data to find these results.

Although the partial matching metric is more accurate, it is expensive in terms of computation cost. Thus, we use the Euclidean metric in the latent space of compressed data for the hierarchical clustering of the database. The problem formulation is invariant with respect to translation, orientation, and scaling. Hence, constraints like geometric restrictions on pivots have to be addressed after finding feasible solutions for the task. 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. However, the database size increases with the number of links in the mechanisms. The solution to this problem is to use learning based methods, where the pattern is learned instead of storing all of the information. This forms one of the basis of the proposed approach. This work has been published in ASME Journal of Computing and Information Science in Engineering, 2019 [71].

Chapter 4

Generative Models for Mechanism Synthesis

4.1 Introduction

In our quest to develop this framework, it is critical that the framework has the ability to understand salient aspects of the linkage parameters and create diverse design concepts for a given task. This chapter presents our step in this direction by formulating a generative model of linkage parameters.

A generative model is based on generative learning principal, which does not just passively observe the events it experiences, but constructs its own perceptions about them. For example, training a generative model for coupler curves of four-bar linkage formulates an understanding of the kind of curves a four-bar linkage can or cannot generate. This understanding is useful in various tasks such as input denoising, modification or imputation. Here, input imputation is defined as the process of adding the missing information in the input which is necessary for the solver to process. Generative models encapsulate the salient information about the observed data which is essential for tasks involving recognition, representation and computational creativity. In this work, we have used VAE [72] as our generative modeling framework. The parameters of generative models are much less in number than that of the data it is trained on. Thus, the model is forced to capture salient attributes and their variation to generate the data similar to it. This encapsulation of salient features is utilized in tasks that require an understanding of the data. In our case, we use it in providing the user a high-level control on manipulating different aspects

of input data and to manage input uncertainties.

This chapter is organized as follows. Section 4.2 reviews neural network architectures used in the research. Section 4.3 presents the theoretical formulation of VAE.

4.2 Review on Neural Network Architectures

4.2.1 Convolutional Neural Networks

Capturing patterns in the images is a well-researched topic; see Forsyth and Ponce [73]. The computer vision literature has witnessed a vast amount of research ranging from classical computer vision approaches to deep convolutional networks. Classical computer vision approaches use hand-crafted feature extractors for capturing spatial correlations, whereas deep convolution networks learn these feature extractors from the training data.

Convolutional Neural Networks (CNNs) are constructed to capture the spatial structure of the input [74]. CNNs consist of a set of learnable filters. In the 2D case, these filters can be represented by 2D matrices, whose coefficients are updated at each step of the training process. At the time of inference, each CNN filter is convolved around the image. A high score is reported for a strong pattern match between the filter and input. The high score gets passed on to the next layer through max pool operation.

In simple words, the output of each layer in CNN is an agglomeration of scores for various learned filters used during the convolution. This ability enables CNN to learn low-level spatial features like edges in the initial layers. As the layers go deep, the composition of simple features constitutes highly complex spatial structures depending upon the training data.

Convolution Operation Convolutional layer use convolution operation between two entities. In the continuous case, the convolution of two functions f and g

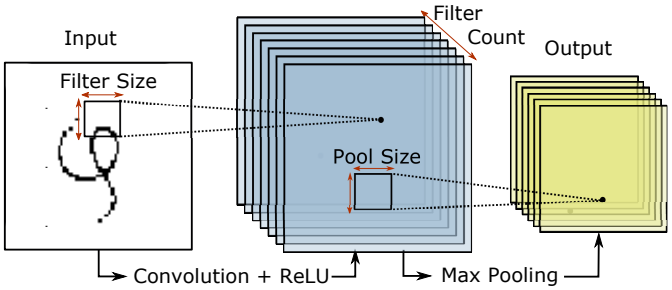

Figure 4.1: A filter is a 2D matrix whose coefficients are the learnable parameters of the convolution network. During convolution, each of such filter it is convolved on the input followed by nonlinear ReLU activation resulting in k such 2D matrices as shown by the middle entity. Next, the output is max pooled to formulate the output to be passed on to the next layer.

is given by,

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau. \tag{4.1}$$

In the discrete case, it is replaced by the sum

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(n)g(n-m) = \sum_{m=-\infty}^{\infty} f(n-m)g(m) \tag{4.2}$$

Here, g is called a kernel function. In 2D discrete case, g can be represented via a 2D matrix which has support on $\{(-M,-N),\ldots,(M,N)\}$. Then, convolution is given by

$$(f * g)(x,y) = \sum_{m=-M}^{M}\sum_{n=-N}^{N} f(x-n,y-m)g(m,n). \tag{4.3}$$

Equation (4.3) has a very simple geometric interpretation as shown in Fig. 4.1. A 2D filter of size $(f,\ f)$ is slid along the image with stride s. Before each sliding action, a sum is conducted and stored into an output tensor at a location governed by number of strides. The final outcome of such operations is a 2D matrix. This entire process is repeated for k number of filters resulting into k number of 2D matrices as shown in the figure. In CNN, the learning parameters are the coefficients of kernel matrices. These coefficients are updated via the Backpropagation algorithm in a stochastic gradient descent method of optimization. For more details; please see see [75], [76].

Table 4.1: Recognition Network Architecture

Layer	Filter Count	Filter Size	Stride	Output Shape	Activation
Convolution 1	32	(11, 11)	(1,1)	(64, 64, 32)	ReLU
Max Pooling 1	-	(2,2)	(2,2)	(32, 32, 32)	-
Convolution 2	64	(5, 5)	(1,1)	(32, 32, 32)	ReLU
Max Pooling 2	-	(2,2)	(2,2)	(16, 16, 64)	-
Convolution 3	128	(3, 3)	(1,1)	(16, 16, 128)	ReLU
Max Pooling 3	-	(2,2)	(2,2)	(8, 8, 128)	-
Flatten 1	-	-	-	8192	-
Fully Connected	-	-	-	100	-
Split	-	-	-	(2, 50)	(-, exponential)

4.2.2 CNN-VAE Architecture for Image Representation of Coupler Curves

The input to the encoder is a 64×64 image. This input is fed to three stacks of convolutional-MaxPooling layers. The output of the convolutional layer is first passed through ReLU activation before it can be max pooled. Max pooling layer passes the highest activation in the filter window to the next layer. The hyper-parameters for each layer are given in Table 4.1. After three such layers, the output is flattened into a single vector, which then is connected to a single fully connected layer which outputs two vectors representing μ and $\log \sigma$ of 50 dimensions each. Vector z is obtained by sampling a multivariate Gaussian distribution with mean μ and standard deviation σ. This vector is passed to the generative model as an input.

The generative model architecture is composed of transpose-convolution layers to finally output a tensor similar to that of image. The architecture and hyper-parameters are given in Table 4.2.

4.3 Theory of Variational Auto Encoders (VAE)

VAE [72] is a neural network architecture that learns to approximate the true distribution of an observed data x. In this work, different models of VAE are trained to learn different observed data, thus x can represent different quantities for different

Table 4.2: Generative Network Architecture

Layer	Filter Count	Filter Size	Stride	Output Shape	Activation
Fully Connected	-	-	-	1024	ReLU
Reshape	-	-	-	(8, 8, 16)	-
Transpose Convolution 1	128	(3, 3)	(2,2)	(16, 16, 128)	ReLU
Transpose Convolution 2	64	(5, 5)	(2,2)	(32, 32, 64)	ReLU
Transpose Convolution 3	32	(11, 11)	(2,2)	(64, 64, 32)	ReLU
Transpose Convolution 3	1	(3, 3)	(1,1)	(64, 64, 1)	Sigmoid

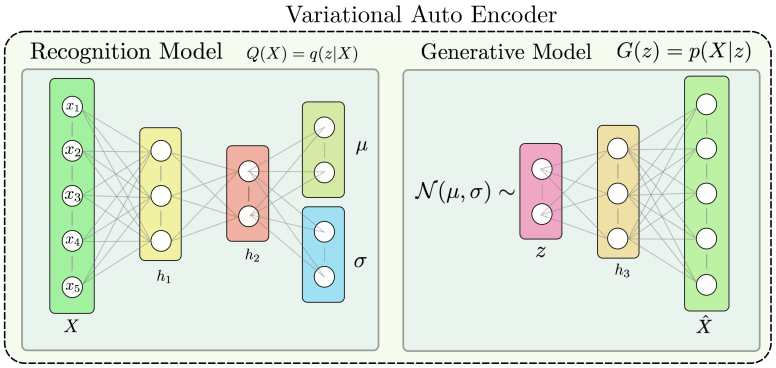

Figure 4.2: Recognition Model encodes the observed data X into probabilistic latent coding z of dimension much smaller than x. In this case, we assume a multivariate Gaussian distribution for z. Generative Model takes samples from this distribution to generate output $\hat{x}$.

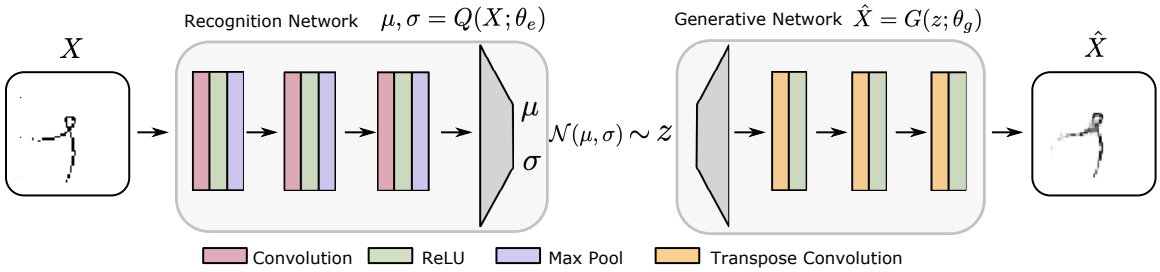

Figure 4.3: CNN-VAE Architecture for Image Representation of Coupler Curves. For encoder, each hidden layer consists of convolution, ReLU and max pooling operation. For decoder each hidden layer consists of transposed-convolution and relu operation.

VAEs. Depending upon the quantity that x represents, x can have different dimensions. For example, if a VAE is trained on the coupler path dataset, then x represents coupler paths and will be denoted by $X_{\text{path vector}}$[1] or $X_{\text{path image}}$[2] . Whereas, if VAE is trained on coupler motions or four-bar linkage parameters, then x would represent coupler motions ($X_{\text{motion vector}}$) and four-bar linkages (X_{fourbar}), respectively. x can be a set of points or tensor representing images.

Figure 4.2 shows a representative architecture of VAE. In this architecture, the Recognition Model encodes the data into the probability distribution of latent variables, while the Generative Model is responsible for generating new data or reproducing trained data. In the middle of this figure, there is feature space encoded by the variable z, which seeks to capture the essence of the input data. However, as opposed to Auto Encoder architecture in which z is a discrete variable, in the VAE, the z is determined via a probability density function.

Let us assume that the d dimensional data x is highly structured and occupies a much smaller k dimensional space. We know that a k dimensional unit Gaussian distribution can be mapped into any k-dimensional distribution through a nonlinear mapping. In other words, it can be said that the data x is generated by some natural process that maps a k dimensional variable z to d-dimensional variable x. We try to mimic this process with an unknown parametric generative model based on hidden variable z, given that probability distribution of z is unit Gaussian, .

$$pr(z) = \mathcal{N}(0, 1), \tag{4.4}$$

$$x = G(z; \theta_g), \tag{4.5}$$

where θ_g are weight parameters of the neural network that acts as the generative model. The variable z is called the latent variable, which contains salient information of the observed variable x. We would like to infer salient attributes z based on

[1] When the path is represented as a sequence of points
[2] When the path is represented as images

observed x, which can be expressed by conditional probability $pr(z|x)$

$$pr(z|x) = \frac{pr(x|z)pr(z)}{pr(x)}, \tag{4.6}$$

where the abbreviation $pr(A)$ represents probability of variable A. Unfortunately, computing probability of x (i.e., $pr(x)$) is usually intractable. As it involves computing the integral $\int pr(x|z)pr(z)dz$. However, we can apply variational inference [77] to estimate the joint probability distribution $pr(z|x)$. We approximate $pr(z|x)$ by a tractable distribution $q(z|x)$, which we define such that it can be computed by a neural network Q.

$$\mu, \sigma = Q(x; \theta_e), \tag{4.7}$$

$$q(z|x) = \mathcal{N}(\mu, \sigma) \tag{4.8}$$

Here, $\mathcal{N}(\mu, \sigma)$ is a multivariate Gaussian distribution function with mean μ and variance σ. Now, we want to find parameters of Recognition Model $Q(x)$ that predict the distribution $q(z|x)$ such that it is very similar to $pr(z|x)$. Then, we can use it to perform approximate inference of the intractable distribution.

The objective is to find parameters θ_g, θ_e of $G(z)$ and $Q(x)$ respectively such that our model generates samples as close as true observed distribution and distribution $q(z|x)$ is as close as true distribution $p(z)$. This is achieved by training the neural network models for maximizing the lower bound of the marginal likelihood ($\mathcal{L}_{\text{ELBO}}$), which is given by,

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{Q(z|x^i;\theta_e)}[(\log(p(x^i|z))) - D_{KL}(Q(z|x^i;\theta_e)||p(z))] \tag{4.9}$$

Here, the first term on RHS represents reconstruction likelihood and the second term is called Kullback-Leibler divergence (KL divergence) [78] which ensures that our learned distribution $Q(z|x;\theta_e)$ is similar to the true prior distribution $p(z)$. Since we assume that $p(z)$ is a Gaussian distribution, the lower bound of marginal likelihood becomes,

$$\mathcal{L}_{\text{ELBO}}{}^{(x^i)} = -(\hat{x}^i - x^i)^2 - (\sum_i^k {\sigma_i}^2 + {\mu_i}^2 - \log(\sigma_i) - 1) \tag{4.10}$$

The training objective is given by,

$$\arg\min_{\theta_e,\theta_g}(-\mathcal{L}_{\text{ELBO}}) \tag{4.11}$$

For further details please see [72]. The training objecting maximizes the lower bound of marginal likelihood

Once, entire VAE is trained, the recognition model and generative model can be used separately or together depending upon the application. In what follows, we describe the details of the Recognition and the Generator Models.

4.4 Recognition Model

The architecture shown in Fig. 4.2 consists of a recognition model, which is an ANN. The inputs are passed through dropout [79], which randomly skips a connection between input and the first hidden layer with a probability of 0.1. This small amount of uncertainty in the input helps in learning robust patterns present in the input. The Recognition model is comprised of hidden layers, which finally produce two z_{dim} dimensional vectors representing mean and variance of latent attribute z. The hidden layers can be convolutional layers or fully connected layers depending upon the nature of observed data. Convolutional layers are often the preferred choice when dealing with images. This is due to the nature of convolution operation which connects only the neighboring neurons to capture the local pattern.

The Output of the Recognition model is a multivariate probability distribution of the latent variable, which captures the salient attributes of the data. Random samples drawn from this distribution are passed to the Generator network. In the case of training via the Back-Propagation algorithm, gradients are passed from generator to recognition model by the means of Reparameterization trick [72]. Recognition model effectively captures the approximate posterior inference ($pr(z|x)$ of the input data and thus can be used for tasks such as recognition, denoising, representation and visualization purposes.

4.5 Generator Model

Parameters of this neural network are learned to map a latent vector to a reconstruction, which would exhibit similar latent attributes if passed through the recognition network. Thus, the generative model can generate data whose probability distribution is similar to that of training data. The architecture of this model starts with an input layer that receives the latent vector and passes through single or multiple layers of neurons culminating into the original size of the observed data. The middle layers can be deconvolutional or fully connected layers that upscale the input they receive from the previous layer.

4.6 Variants of VAE

Denoising VAE VAE networks can be used to remove noise from input data. This is done by adding noise to the input data while training. However, the generator is forced to produce original noise-free samples by defining the reconstruction loss between the reconstructed image and the original noise-free image. This forces the recognition network to encode robust features from the data.

Conditional VAE Until now, we have seen VAE with unsupervised learning architectures, i.e., which requires unlabeled data for training. C-VAE is trained to learn the conditional distribution of an observed variable with respect to an explicitly observed property (or, label) y. This is achieved by small modifications in the aforementioned VAE architecture. Instead of only providing with observed data, the recognition network accepts concatenated input of x and y. Moreover, the generator also receives a concatenated input of z and y. This grants C-VAE additional information for the variational inference task. This can be used to generate samples that are highly associated with the given y at the time of generation.

Chapter 5

Computational Creativity and Task Conditioning

5.1 Introduction

The task in this dissertation is defined as the input a solver consumes in kinematic synthesis. For motion generation, the task is a set of poses[1] that describe the intended motion. In the case of path generation, the task is a set of points describing the path. Since the task is devoid of knowledge about the outcome of solvers, there is no way of knowing if the task is ill-posed. The task conditioning is the process in which the task is modified to make it more conducive for the solvers. In addition, the task conditioning can also include informed modification for satisfying the desired context. This chapter presents the details on how such informed modifications are achieved using a data-driven strategy. Section 5.2 presents the details of the process. Section 5.3 presents data preparation and representation for motion and path tasks. Section 5.7 presents examples of feasibility conditioning of tasks. Section 5.9 presents how a context function can be learned in a data-driven way.

5.2 Task Conditioning Overview

To such modifications, it is imperative to possess knowledge about the properties of the task. This knowledge is expressed as a prior probability distribution, often simply called the **prior**. This work uses Variational Auto-Encoders (VAE) to learn this **prior** by learning from a set of real coupler trajectories of randomly constructed

[1] combination of positon and orientation

linkages. Thus, the recognition model of a VAE is trained to compute a maximum a posterior probability (MAP) estimate of latent features for the given input given by,

$$\mu, \sigma = Q(X_{task}), \tag{5.1}$$

$$z \sim \mathcal{N}(\mu, \sigma). \tag{5.2}$$

Here, z is a sample from the recognized latent distribution. This distribution can be thought of as a distribution of recognized properties of coupler curves that were observed in the task by recognition model.

Then, the generative model of VAE generates few samples from approximate marginal likelihood distribution of recognized latent features given by,

$$\hat{X} \sim G(\mathcal{N}(\mu, \sigma)). \tag{5.3}$$

Here, $\hat{X}$ is a sample from a distribution computed by approximate marginal inference for a given input task X_{task} and is given by,

$$\hat{X} \sim pr(X|X_{task}) \approx \int pr(X|z)pr(z|X_{task})dz. \tag{5.4}$$

Intuitively, it represents a modified task after applying a **prior** function which was learned in a data-driven way. VAE computes a distribution ($pr(\hat{X}|X)$ of such modified tasks, which are then inputted to synthesis solvers. These modified tasks are termed as **Variational Inputs**.

5.3 Representation of Tasks

This section presents the details of the normalization and representation of tasks in various formats. In this work, we showcase the conditioning of tasks for path and motion generation problems.

First, consider a database of coupler paths formed by planar linkages. Each coupler path is an ordered collection of m points sampled uniformly throughout the entire rotation of the crank, where each point has x and y coordinate. Thus, each coupler

curve $X_{\text{path vector}}$ is $\{x_j, y_j\}_{j=1}^{m}$ and each coupler motion $X_{\text{motion vector}}$ is $\{x_j, y_j, \theta_j\}_{j=1}^{m}$ for motion data. Both of the task representations are first normalized before feeding to VAE.

5.3.1 Data Normalization

First, each coupler trajectory is normalized with respect to scale and position. This is done by subtracting the means $(\bar{x}, \bar{y})$ and dividing by the root mean squared variance in Cartesian X- and Y-directions. Next, the trajectory is rotated such that its principal component axes are aligned with the coordinate axes. The principal component axes are the eigenvectors of the covariance matrix of a point cloud data that defines the trajectory. The covariance matrix C of a point cloud of m 2D points $\{x_j, y_j\}_{j=1}^{m}$ is given by,

$$C = \begin{bmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{bmatrix}, \tag{5.5}$$

$$C_{xx} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(x_i - \bar{x}), \tag{5.6}$$

$$C_{yy} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y})(y_i - \bar{y}), \tag{5.7}$$

$$C_{xy} = C_{yx} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y}). \tag{5.8}$$

Figure 5.1 depicts the normalization procedure for two couple paths.

Another common approach is to treat this sequence of points $\{x_j, y_j\}_{j=1}^{m}$ as a one dimensional temporal signal with parameter t given by,

$$\text{CC} = \{x(t), y(t)\}$$

This formulation allows for applying various signal processing techniques like Fourier Analysis. While this type of modeling has many advantages which are evident by the success in path generation [6],[5], it fails to capture spatial correlations that are not implicit in temporal correlations.

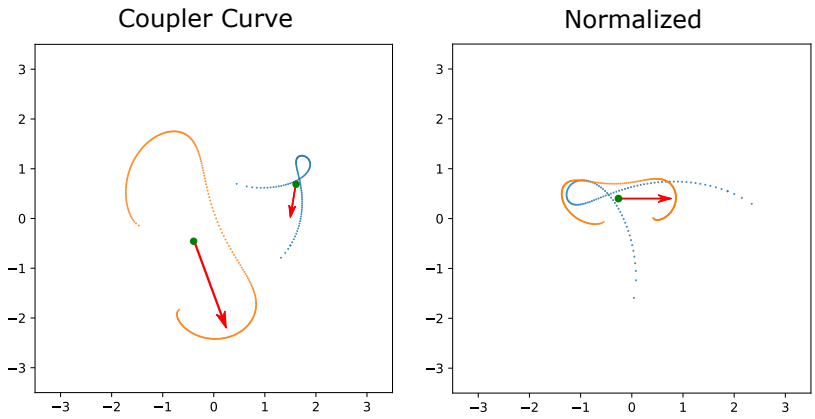

Figure 5.1: Each curve is normalized for position, scale and orientation.

5.3.2 Representing Task as an Image

The proposed image-based representation allows us to capture those correlations in exchange for losing temporal correlations between the input sequences. However, it can be argued that the ordering is embedded in the spatial distribution of pixels since unlike typical computer vision applications, such as facial recognition, which have a true 2D set of pixels for images, a curve is fundamentally a one-dimensional entity only. In addition, if the task does not need a specific ordering of points, the image-based approach allows for capturing all possible input orders in a unified way. An image represents a probabilistic coupler curve, where the intensity of a pixel represents the likelihood of containing a coupler curve point. First, each coupler curve is normalized with respect to scale, position, and orientation.

The normalized planar curve is now discretized into a 2D image. The two-dimensional Cartesian space is linearly discretized into $S-1$ partitions along each axis, where S corresponds to a side of the image. The partitions along x-axis is done as follows: x coordinates that lie between $-\infty$ to $-x_{lim}$ are binned into a the first partition. The next partition includes x ranging from $(-x_{lim}, -x_{lim} + x_{step}]$, and so on up to the last partition $(X_{lim}, +\infty)$. Here, x_{lim} and x_{step} are discretization parameters, which control the range and resolution respectively. Figure 5.2 depicts the discretization of a normalized curve. It can be noted that the extreme coordinates

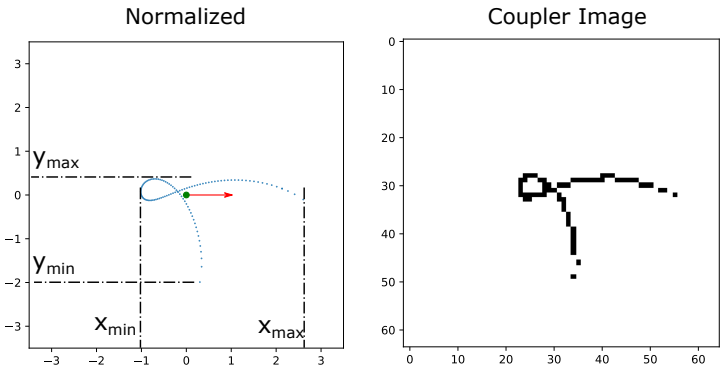

Figure 5.2: The extreme coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$ of a normalized coupler curve are shown. If a point lies in Cartesian span of pixel, it is given 1.0 probability.

play an important role in deciding resolution and range of pixels.

To estimate x_{lim}, a statistical study is conducted on a dataset of 6,818 linkages comprising of four-bar (2,188), Stephenson I (2,754) and Stephenson III (1,876) six-bar linkages. The coupler curve for each linkage is stored as a sequence of points sampled per one-degree change in the crank angle. The number of points in coupler curves is different for different mechanisms. Figure 5.3 shows the histogram of extreme coordinates corresponding to the normalized coupler curves from the database. It can be seen that more than 99% of linkages are contained by a square with a side 7 and center at the origin. Thus, x_{lim} is set as $7/2 = 3.5$. For an image with $(S \times S)$ pixels, x_{step} is given by,

$$x_{step} = \frac{2x_{lim}}{S-1}.$$

Here, each pixel of an image represents a probability of containing at least one point of the coupler curve. In this work, we choose $S = 64$, which provides sufficient resolution to capture the variation in coupler curves. Figure 5.2 depicts the discretization of a normalized coupler curve into an image. It should be noted that the discretization process causes some loss of information. However, in many cases, the input is inherently vague. Thus, it acts as a buffer and prevents the further process from over-emphasizing the numerical precision. One can also notice that image-based representation removes the order of points in the curve. However, it grants spatial

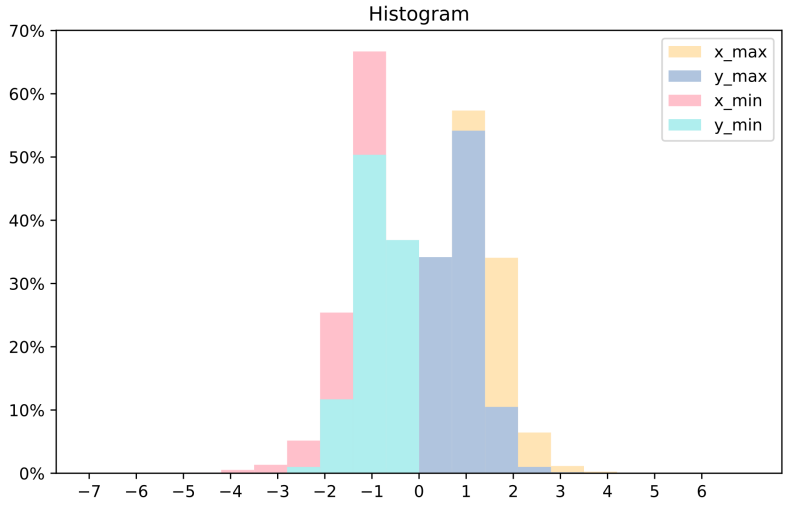

Figure 5.3: A histogram of extreme coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$ of coupler curves. It can be seen that less than 1% of the normalized coupler curves have a point with an absolute coordinate larger than 3.5 units.

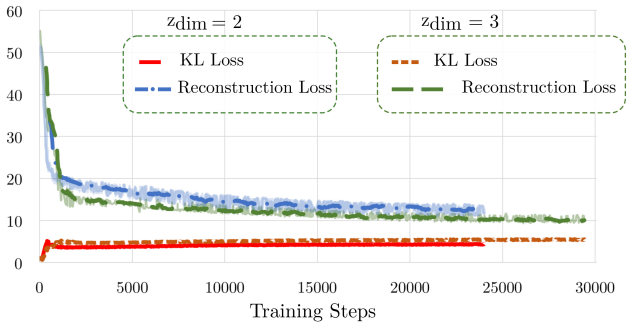

Figure 5.4: Reconstruction and KL Divergence losses for two architectures with z dimension 2 and 3. It can be seen that higher z-dimension enables capturing more variation in the database, which results in lower reconstruction losses.

correlations between neighboring points, which helps in capturing interesting spatial patterns.

5.4 VAE Training

Various datasets of linkages are used to train various architectures of VAE with varying numbers of latent dimensional size. A new linkage is added to the dataset if it is sufficiently dissimilar from all the linkages previously added to the dataset. Here, the measure of dissimilarity between two linkages is given by a sum of $L2$ norms between the normalized trajectories traced by all moving points of two linkages. Table 5.1 presents the details on various datasets used for training.

Figure 5.4 depicts training losses for two such architectures with latent dimension (i.e., the dimension of z vector) 2 and 3. As we can see in Fig. 5.4, the model with a 3-dimensional z vector space achieves lower reconstruction error with slightly higher KL divergence loss. This result is expected because with an increase in latent size, the room to capture variation in the data increases. However, the increase in the latent dimension also increases the complexity in visualization, interpretation, and manipulation of latent attributes in recognition tasks. It is interesting to notice that the two losses somewhat compete with each other in the training process. The reconstruction loss pushes the model to capture the diversity in the dataset, by forcing

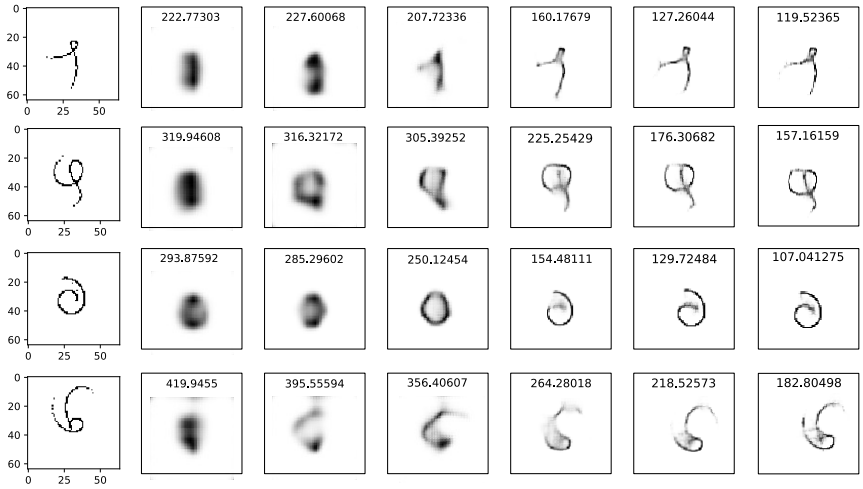

Figure 5.5: The training progress is reflected in the reconstruction quality of test coupler curve images. The corresponding variational lower bound $L_{ElBo}^{X^i}$ is displayed above each reconstructed image. It can be seen that the probability assignment by VAE for each pixel improves over the training.

the generator to be able to reconstruct every type of data in the dataset. Whereas, KL divergence forces the latent space to occupy a restrictive distribution and thereby demanding coherence in the generation.

Figure 5.5 shows improvement in reconstructions of coupler image samples from the test set as the training progresses. It can be seen that the pixels with a higher probability of containing a coupler point are assigned a higher probability and vice versa. Once the network is trained sufficiently, the encoder learns to capture spatial correlations and returns a probability distribution in latent feature space. Each sample from this distribution has a high probability of being a valid coupler image. This feature is used to obtain feasible representations of raw input and is demonstrated in Section 5.7.

5.5 Recognition and Generation of Coupler Trajectories

For training or inference, each data point $X_{\text{path vector}}$ is passed through a recognition model that computes the parameters for multivariate Gaussian distribution of latent vector. In inference, the aim is to recognize the salient features of the input

Table 5.1: Datasets used for Training VAE

Data Type	Size
Four-bar Linkage (Grashof Only)	480
Four-bar Linkages	2188
Slider-Crank Linkages (Grashof Only)	466
Six-bar Stephenson IIIa Linkage (Grashof Only)	937
Six-bar Stephenson IIIa Linkage	3902

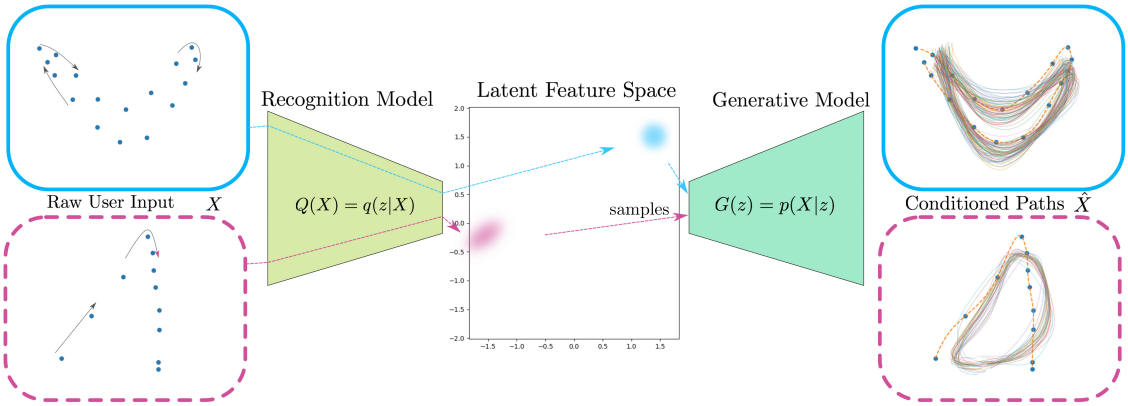

Figure 5.6: The raw user input X is passed through a recognition network, which captures the salient information in the form of multivariate distribution of latent features. Random samples from this distribution are fed to the generator to generate paths with a high likelihood of producing good solutions. Moreover, users can manipulate the sample location in latent space, which gives them a low-dimensional and higher level of control on modifying the shape of the path.

point cloud. To showcase the inference functionality, let us consider two-point sequences that are obtained by manually drawing points on an interface. Each of the sequences represents an approximate target path. Now we would like to infer their salient features and generate plausible coupler paths having similar salient features. We take a trained VAE with two-dimensional $z_{\text{path vector}}$ vector trained on closed coupler paths (VAE-z2-path). The two-point sequences, one at a time is passed through a recognition network, which predicts a 2-D Gaussian distribution of latent features for each of the input as shown in Fig. 5.6. Now for each case, random samples drawn from the distribution are passed to the generator network of the VAE, it generates the samples with closed paths that resemble the original input path as shown in Fig. 5.6.

5.6 Interactive Shape Modification with VAE

Let us assume a scenario where the user needs to specify a closed-loop target path. Since the synthesis of linkages is chaotic, it is always fruitful to condition the inputs such that the probability of finding good linkages is maximized. Moreover, it is desirable to have a higher level of control on the overall shape of the target path. We use VAE to accomplish both of the above tasks. First raw user input $X_{\text{path vector}}$ is passed through recognition model as shown in Fig 5.6. The recognition model captures the shape user intends to draw and returns it in the form latent feature distribution $q(z|X)$ as shown in Fig. 5.6. The VAE generator takes a sample feature vector z_s from $q(z|X)$ and generates a path $\hat{X}$. It should be noted that $\hat{X}$ is a sample from the distribution $p(X|z)$ and has more probability of being a path drawn from a four-bar. As we can see in Fig 5.6, generated paths follow the user-defined points, but also resemble paths generated by Grashof four-bar linkages. Moreover, the user can select or modify the z_s interactively based on the variation of its corresponding generated path. Hence, the user has a higher level control on the shape of the path which improves the experience rather than modifying the input path point by point.

This can be achieved by creating a user interface where the user can see the effects of changing feature vector z in all possible directions. Figure 5.7 presents samples created by the generator for different latent vectors. Here, the user can visualize where the given curve lies, and change its shape by moving across the plane in 2-D. This approach is amenable if the dimensionality of feature vector z is kept below 4].

5.6.1 Type Synthesis via Recognition

Several VAEs with different architectures are trained on the dataset that comprises of coupler paths, motions from four-bars, slider-cranks and Stephenson type six-bars. Since the linkages above have a different topology, their coupler motions should possess some features that are affected by it. Figure 5.8 depicts 2-D embedding of closed coupler paths from Four-Bar (all revolute joints), Slider-Crank and Stephenson six-bar mechanisms. It can be clearly seen that six-bar coupler curves cover a wider variety in shape compared to four-bar linkages. Variety covered by slider-crank linkages is the lowest and is mostly overlapped by a four-bar linkage. The reasoning behind this observation can be given by the fact the slider-crank linkages are a special case of 4R linkages when the length of the rocker link and its fixed pivot approach infinity. A classifier trained on such data can predict the probabilities of a given task being fulfilled by the corresponding linkage type.

5.6.2 Example User-ML Interaction

In this section, we showcase the application of VAE to solve a path generation problem using a motion generation problem solver. As we have stated earlier, the ML Intermediary can take crude input from the user and provide all the necessary information required by available solver with computational subtleties. These tasks require VAE to capture primitive information provided by raw user input and generate a set of plausible coupler motions. It is important to note that since the orientation information required by the motion generation solver is not provided by the user, it

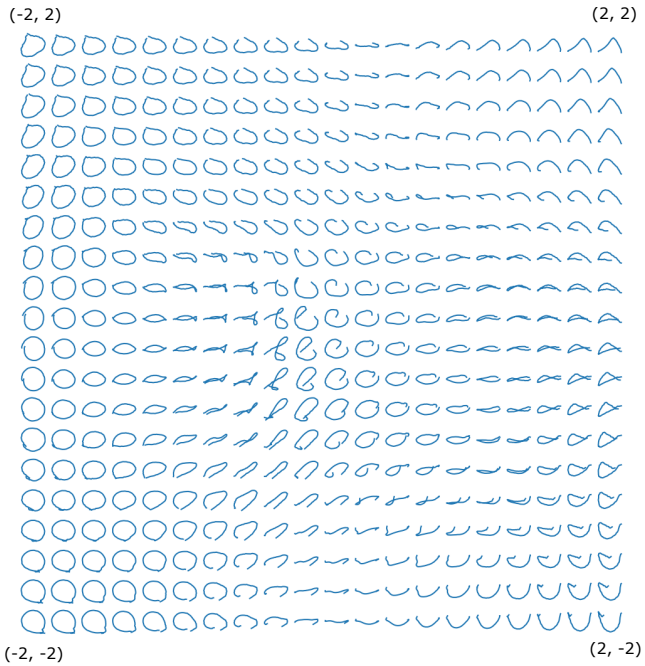

Figure 5.7: Samples generated from a generator from a two-dimensional latent vector which is a vertex in the two-dimensional uniform grid with limits shown in corners. It can be seen that samples generated from neighboring vertices are highly correlated, which gives away the indication that recognition network learns the salient information about the shape of coupler curves.

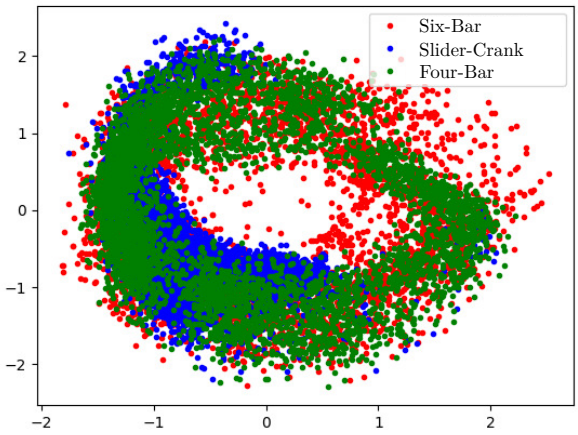

Figure 5.8: Visualization of 2-D feature embedding obtained by training a VAE on closed coupler paths of various planar linkages. The variety in features by means of spread over the space directly relates to the variety in the shape of respective linkage type.

should be guessed by the ML Intermediary. This is done by using a C-VAE that is trained to generate coupler motions $X_{\text{motion vector}}$ data given its path $X_{\text{path vector}}$. The architecture and training scheme for the C-VAE follows the discussion presented in Chapter 4.6. We use our computational methods [1],[2] for solving the motion generation problem in real-time. The solutions returned by the solver are fed to the linkage recognition model of a VAE trained on entire four-bar linkages. This linkage recognition model computes a compact feature representation of each linkage on which we apply the K-means[80] clustering algorithm. The details of linkage recognition and clustering are presented in section 5.11. It should be noted that the feature distributions of linkages produced by a recognition model are in ten-dimensional space. Thus, the depiction of the latent distribution of linkages and following clusters as shown in Fig. 5.9 are only for illustration purposes. The cluster centers are distinct solution concepts. Figure 5.9 depicts the entire procedure from start to finish. The solutions obtained using the proposed approach yields diverse mechanisms with various linkage topologies. To juxtapose our approach with classical approaches, Fig. 5.9 also depicts the result obtained using a classical path synthesis method [3] based on Fourier Descriptors. It can be seen that the classical result is similar to one of the concept solutions obtained using variational synthesis.

5.7 Feasibility Conditioning and Input Feedback on Image-based Task

Raw user input is defined as an image with a different distribution of pixel intensities than the ones the VAE has learned while training. If such an input image is fed to VAE, the recognition network tries to find the learned patterns in the input. The patterns with high correlation with learned patterns will receive a high score that will be passed through max-pooling layers and eventually reflect in the obtained latent distribution. If samples from that distribution are passed through the generative model, it will produce coupler curve-like images that share similar

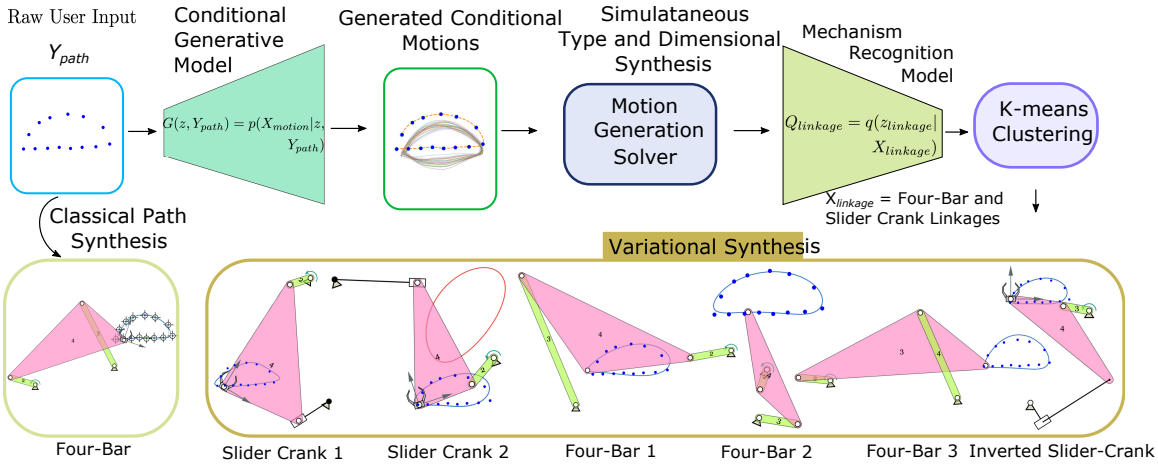

Figure 5.9: C-VAE generates coupler motions corresponding to the path input. The orientation information of generated motion is not shown for cleaner illustrations. The generated motions are fed as inputs for the motion generation problem. The solutions solved by [1] are passed through a linkage recognition model which results predicts the latent distribution for each solution. This mixture of distributions is clustered to form distinctive concept distributions. The four-bar linkage in the bottom left corner depicts the optimal path synthesis solution using [3].

spatial attributes. Figure 5.10 shows examples where VAE takes in raw inputs and computes feasible inputs that are used for downstream tasks for path generation. It can be seen that VAE shifts the probability assignments on the pixels of the raw input. Figure 5.10 highlights this effect on various raw inputs. This is used to provide feedback on the user input. Figure 5.11 depicts the outcome of VAE construction for two different inputs. The input shown on the top can be closely approximated by a four-bar mechanism, whereas the other input is highly unlikely for a four-bar or six-bar to interpolate. Since the VAE is trained on coupler curves of four-bar and six-bar linkages, it can reconstruct the original input with high accuracy. On the other hand, second input is highly modified by VAE to change it into a more conducive input. The difference between the modification done by VAE is reflected in the percentage change in the histogram of pixel intensities. For the first image, almost 70% of the pixels retain their original 100% intensity, whereas that number is a mere 17% for the second input.

5.8 Latent Space Exploration

During training, VAE learns to encode X in a prior distribution which in our case is a multivariate Gaussian. Latent vector z corresponding to X is distributed such that the reconstructions $G(z)$ share similar spatial features. Figure 5.12 presents an example where a known coupler image $X_{\text{path image}}$ is passed through recognition model of VAE with ten-dimensional latent space to obtain its mean latent embedding z_μ. Then the latent embedding is shifted along each axisymmetric direction and the corresponding latent vector is reconstructed via generative model. It can be seen in Fig. 5.12 that the neighboring reconstructions not only share similar spatial features they smoothly morph into other feasible coupler images. This property can be exploited to make higher-level changes to X_{task}.

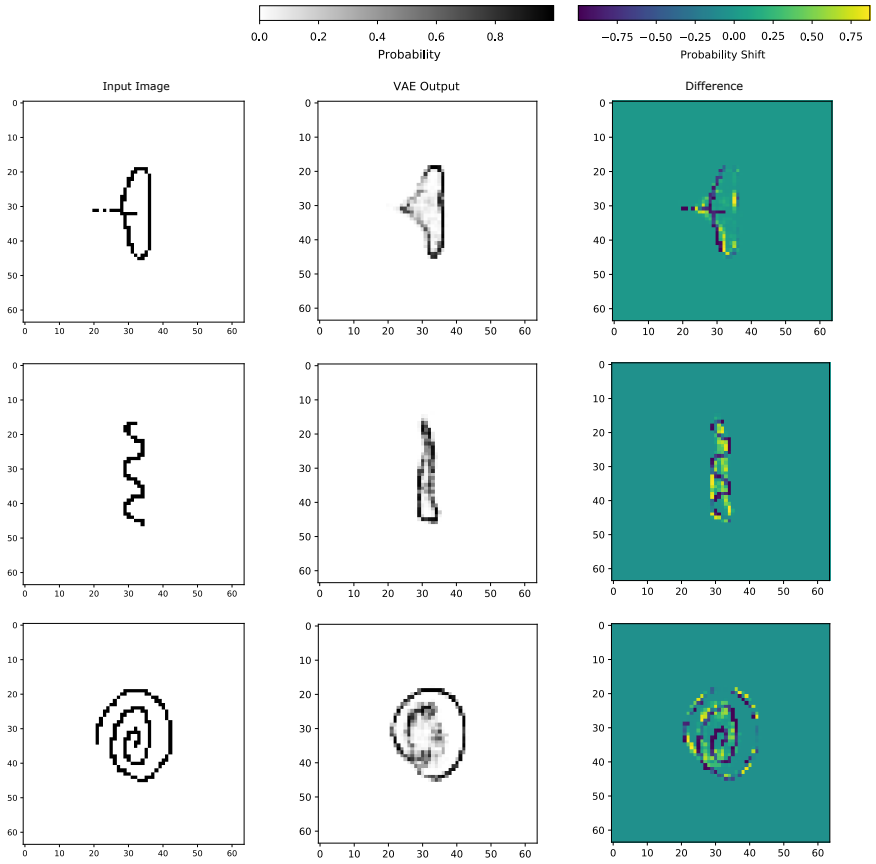

Figure 5.10: Raw input image is passed through VAE which provides a feasible input image. The shift in the probabilities is highlighted in the rightmost images. The shift shown in dark indicates that the portion of raw input is highly unlikely for linkage from the dataset to interpolate. This provides visual and intuitive feedback to the user on the input.

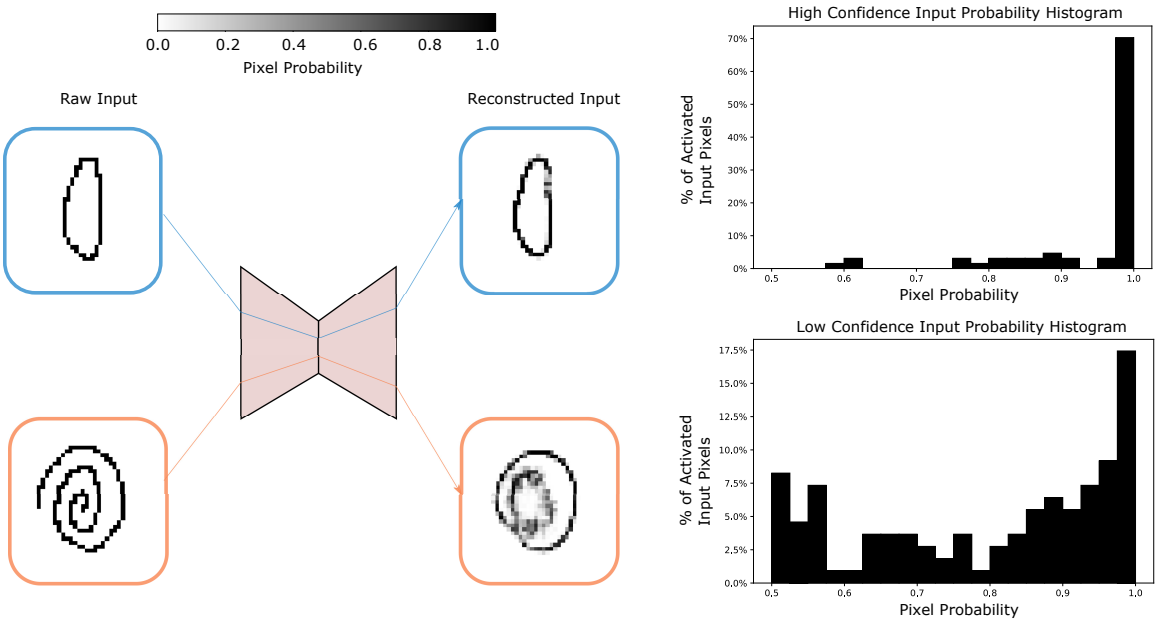

Figure 5.11: The raw input on the top is close to the true distribution of coupler curves, thus VAE retains almost 70% of the highest intensities as depicted by the histogram on the top. For the second input, VAE overrides user assignments on the input pixels by a large number. This is due to the highly spiral nature of the input. It can be also seen in Fig 5.10 that the inner portion of the input is highly penalized.

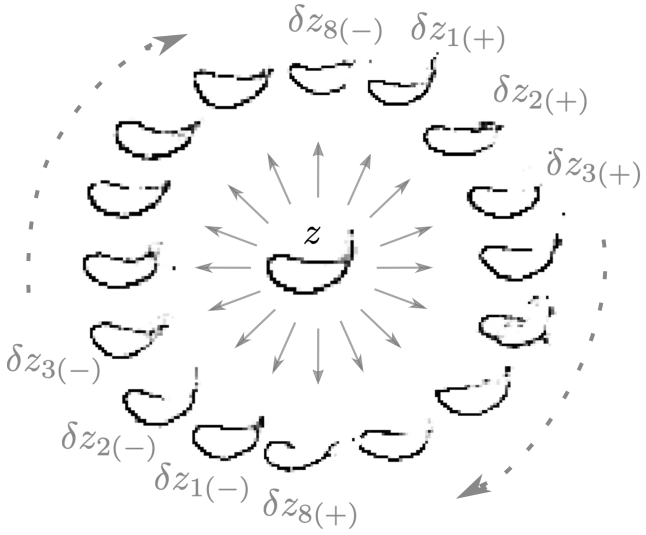

Figure 5.12: Latent exploration in axisymmetric directions is depicted. The original latent vector z is translated in axisymmetric directions. The arrow towards $\delta z_{1(+}$ indicates z is translated in positive direction along $z1$ axis in ten-dimensional space.

5.8.1 Informed Latent Exploration

Latent space allows us to navigate on a manifold of feasible coupler curve images. We can further exploit this by performing an informed exploration. Let us consider an example where the objective is to design a re-configurable linkage that satisfies a family of coupler curves. This family of coupler curves can be represented by a curve in latent space. VAE can be used to find a family of curves that morph from one curve to another by interpolation between the two latent curves. Figure 5.13 show a few examples of such interpolations. Although it should be noted that not all possible curves in latent space may correspond to feasible families of coupler curves. This is due to the possibility that the latent space may contain some void spaces as depicted by Fig 5.14. This happens if certain data samples are drastically different from others.

5.9 Context Conditioning

Until now, we have seen that the latent space z of a VAE trained on real data X can be used to reconstruct samples $\hat{X}$ that have a high similarity to X. This section presents a strategy to explore the latent space in a more effectively to generate a samples $\hat{X}_{\text{context}}$ that maximize a fitness function $f_{\text{context}}(X)$. The algorithm for context conditioning is given in Algorithm 3,

It can be argued that if there exist a continuous function $f_{\text{context}}(X)$ then one can simple perform the search of X that maximizes $f_{\text{context}}(X)$. However, the distribution of real X data is non-continuous and occupies a much smaller dimension in its high dimensional domain. Thus, a direct optimization w.r.t. X can get lost into large infeasible regions of X. In addition, searching in ow dimensional z space is more efficient and the results are more semantically similar X to the original task X_{task}. It should be noted that the objective is not to cause significant changes to the original task X_{task}.

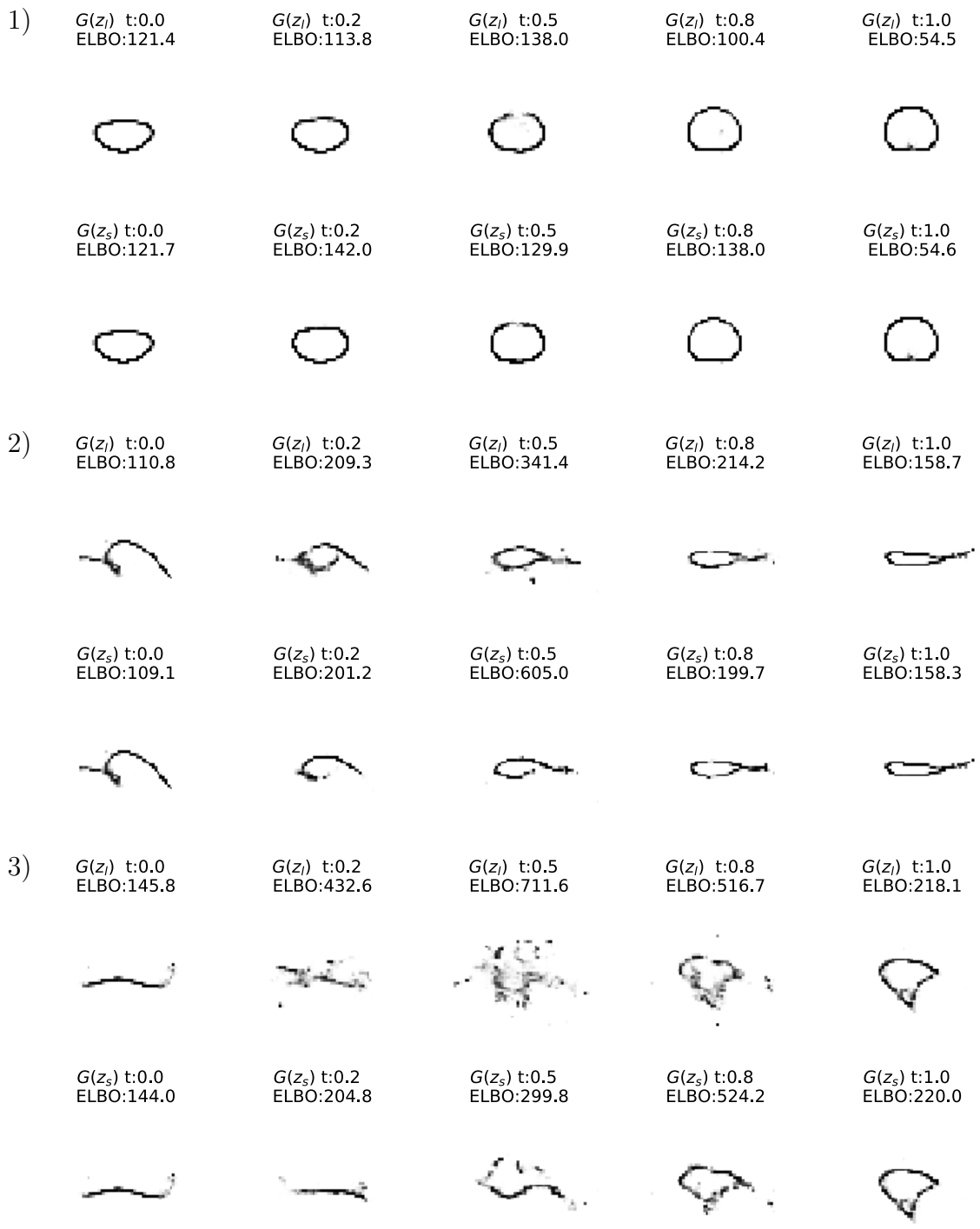

Figure 5.13: Examples of linear and spherical interpolation between a pair of known coupler images. $G(z_l))$ and $G(z_s))$ denote reconstructions for linear and spherical interpolations respectively. It can seen that in some cases, the interpolation passes through infeasible region indicated by higher $\mathcal{L}_{\text{ELBO}}$.

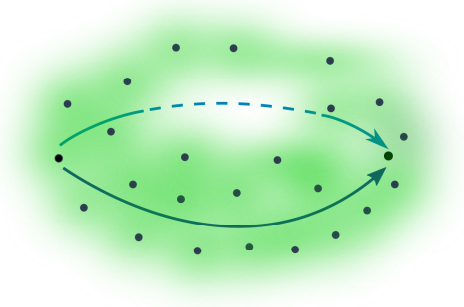

Figure 5.14: Illustration of gaps in latent spaces that lead to higher reconstruction errors if a sample is drawn from those regions. Two interpolation strategies are shown, one of which passes through the infeasible region.

Algorithm 3: Contextual Latent Exploration

Input : X_{task}, $f(X) = f_{\text{context}}(X)$

Output: $\hat{X}_{\text{context}}$

$\mu, \sigma = Q(X_{task})(\text{Input})$; ▷ task recognition

$z_{\text{init}} = \mathcal{N}(\mu, \sigma)$ ▷ Initial starting point

$z = z_{\text{init}}$

$\frac{\delta f(\hat{X})}{\delta z} = \frac{\delta f(\hat{X})}{\delta \hat{X}} \frac{\delta \hat{X}}{\delta z} = \frac{\delta f(\hat{X})}{\delta \hat{X}} \frac{\delta G(z)}{\delta z}$ ▷ $\frac{\delta f(\hat{X})}{\delta \hat{X}}$ and $\frac{\delta G(z)}{\delta z}$ are analytically known

Maximize f(X) w.r.t. z ▷ Using Stochastic Backpropagation

using gradients $\frac{\delta f(\hat{X})}{\delta z}$

Subject to,

$||z_{new} - z_{init}|| < \text{Z}_{\text{Threshold}}$,

$\mathcal{L}_{\text{ELBO}}(G(z)) < \text{ELBO}_{\text{Threshold}}$ ▷ Suitable thresholds should be chosen

return $G(z)$

5.9.1 Context as Desired Fixed Pivot Regions

In the case of one degree-of-freedom linkages, there exists a definite relationship between fixed pivots and the coupler curve of the linkage. In the case of four-bar linkage, three linkages exist that can trace the exact same curve. The set of such linkages are called cognates. The number of cognates can be different for different topologies.

In this case, we use the data-driven approach to learn the relationship between coupler curves and the corresponding fixed pivot locations by training a neural network. Instead of training a regression model that predicts the exact fixed pivot locations, we train a set of classifier models that predicts the probability of a discretized region containing a fixed pivot. This is shown in Fig. 5.15.

The 2D space for fixed pivot prediction is discretized into $k \times k$ pixels, where k is the number of divisions along x or y axis. Here we take $k = 7$. The objective for classifier (FP) is to learn a set of functions $FP_{i,j}(X_{64\times64})_{i=1,j=1}^{i=7,j=7}$, where each function $FP_{i,j}$ predicts the probability of the region covered by pixel at i, j. Since one coupler curve can be traced by many linkages (three for four-bar), $FP(X)$ should predict multiple possibilities in terms of a probability distribution.

For $i = 7, j = 7$, f comprises of total 49 functions. For the input X shown above, it should predict at least two possibilities. The below figure displays the expected output for X. A classifier is trained that predicts a probability for each of the 49 pixels. Figure 5.16 presents classification result of a sample input.

We use this classifier as the context function to make small informed changes in a task image depicted in Fig. 5.17. $FP_{target}(X)$ in Fig. 5.17 depicts a binary mask. This mask is multiplied with the output of the classifier. Summation of the above product is treated as the fitness function in context conditioning given by,

$$f_{\text{context}}(X) = \sum_{i=1,j=1}^{i=7,j=7} FP(X) \cdot FP_{\text{target}}(X) \tag{5.9}$$

Then the procedure presented in Algorithm 3 is followed. This procedure con-

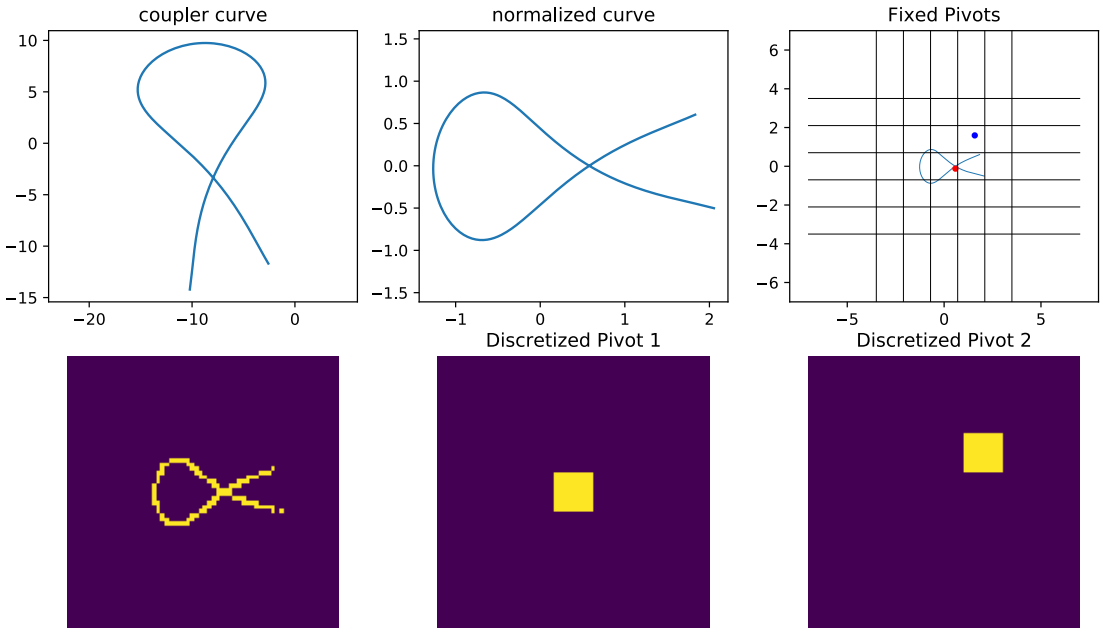

Figure 5.15: The fixed pivots for the linkage are transformed according to the normalization of the coupler curve. The location of each fixed pivot is discretized to create a one-hot vector of 49 dimensions, depicted in an image form.

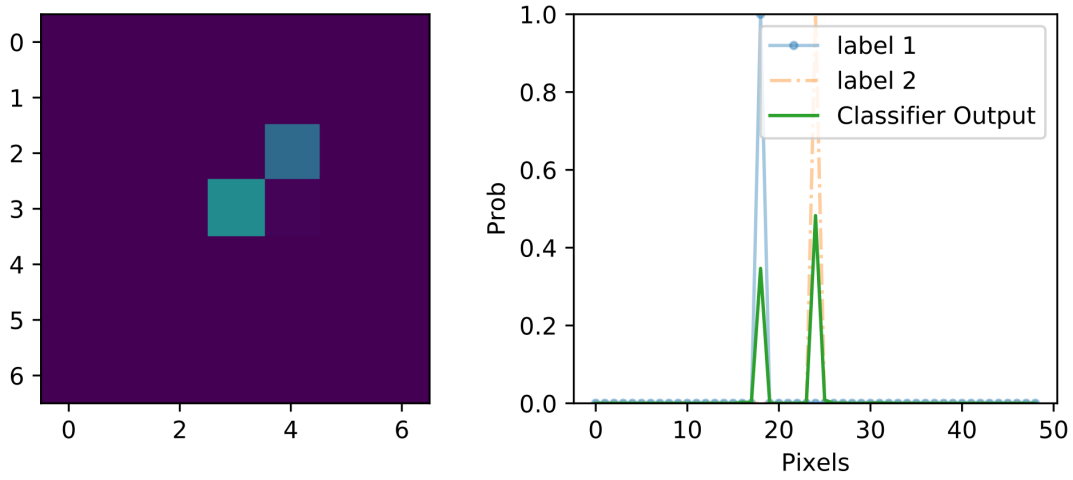

Figure 5.16: Classifier predicts a probability for each of the 49 pixels. The probability indicates the likelihood that the region spanned by that pixel contains a fixed pivot. The pixels where the fixed pivot lie are given a significantly high probability than the rest of the pixels.

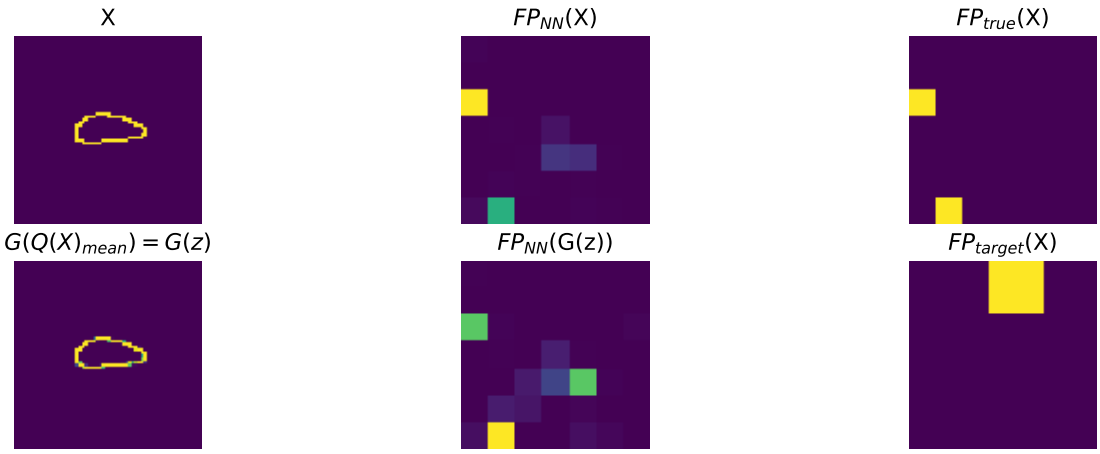

Figure 5.17: The approach start with a linkage with coupler curve image X and fixed pivots at $FP_{true}(X)$. A trained classifier $FP_{NN}(X)$ accurately predicts the true location of fixed pivots. The X is reconstructed using a trained generative model which will be used in context conditioning. The image titled $FP_{target}(X)$ highlights a portion of classifier which will be subjected to maximization in the optimization routine given in Algorithm 3.

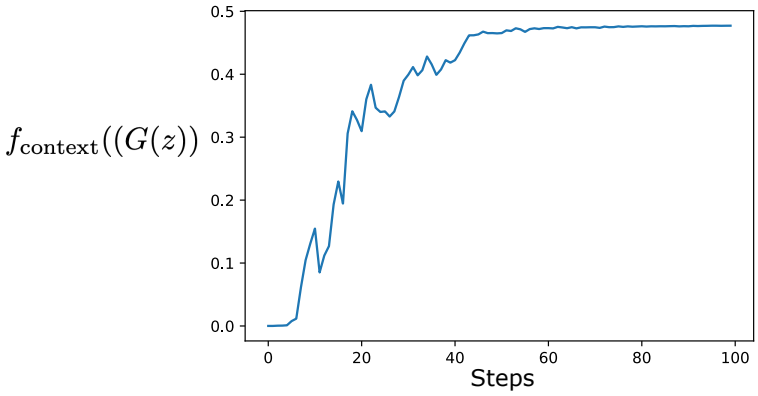

Figure 5.18: Value of objective function over 100 steps of gradient based stochastic optimization.

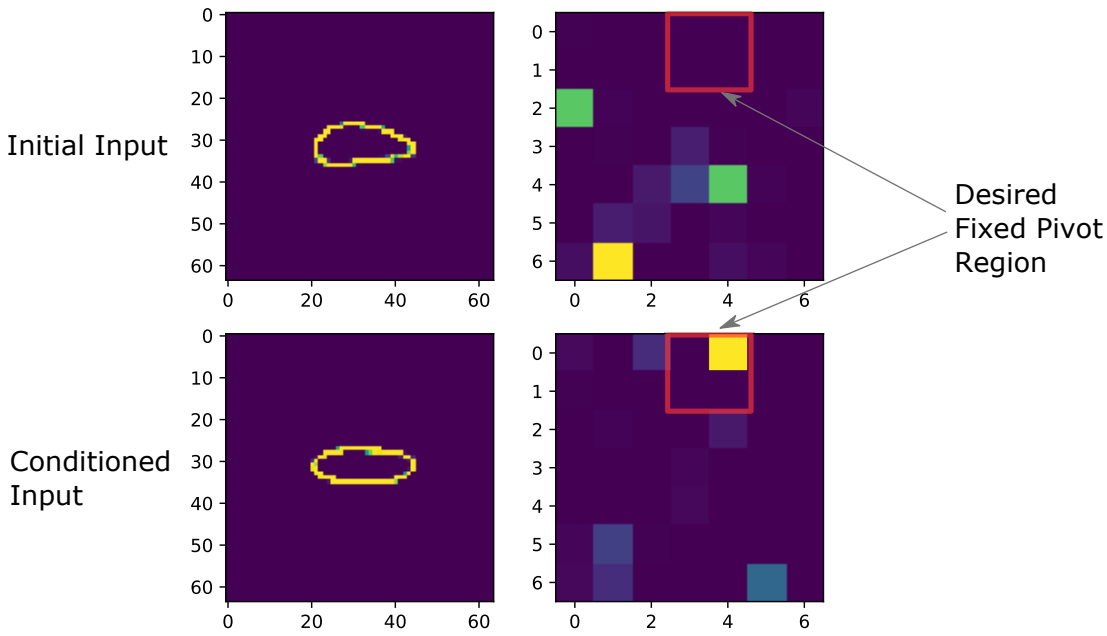

Figure 5.19: Initial and final coupler curve images with their corresponding fixed pivot region predictions. It can be seen that only subtle changes in the coupler curve are needed to satisfy the desired fixed pivot predictions.

ducts an informed exploration in the latent space that maximizes the fitness function. Figure 5.18 depicts the increase in value of objective function as the optimization progress. Figure 5.19 depicts the initial and final coupler curve images. It can be seen that only subtle changes in the coupler curve are needed to satisfy the desired fixed pivot predictions. It should be noted that the output of context conditioning is not guaranteed to produce the solutions with fixed pivots lie in the desired region. However, it is an input that has a higher probability (indicated by trained classifier) of finding solutions with fixed pivots in the prescribed regions. In this work, we have developed a synthesis algorithm that provides theoretical guarantees to satisfy such practical constraints in the case of motion generation for four-bar linkage. The approach presented in this section, however, is more versatile and can incorporate any data-driven context and is invariant to linkage topology.

5.10 Nearest Latent Neighbors for Variational Path Synthesis

In Section 5.7, it was shown that the recognition network captures the spatial correlations that highly correspond to that of coupler curve images. To take advantage of this fact, the database is remapped into a 50-dimensional space where each mechanism is represented by the latent vector obtained using the recognition model. At the time of synthesis, raw input is passed through the recognition model to find a distribution of possible latent vectors. K-Nearest neighbors to the mean of this distribution are returned as potential solutions. Figure 5.20 presents some of the solutions obtained for the nearest neighbor query on raw user input.

5.11 Linkage Clustering

Consider a set of M linkages represented by set of vectors, $\{X_{linkage_i}\}_{i=1}^{M}$. Now, it is useful to find K representative linkages that cover the variety of obtained solutions, where K is a user-selected parameter. To find such K representative linkages, we

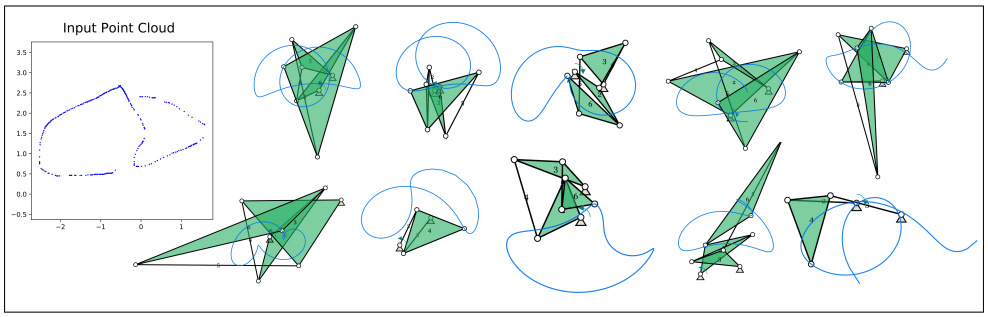

Figure 5.20: The linkages corresponding to the K nearest neighbor in latent space for the given raw input are depicted. It can be seen that the linkages exhibit a wide variety in terms of their coupler path; thereby capturing input uncertainty.

perform K-means[80] Clustering on the set of M linkages, where $K << M$. Before performing clustering, we first pass the set through recognition module to obtain a compressed representation for the set $\{(\mu_{linkage_i}, \sigma_{linkage_i})\}_{i=1}^{M}$ given by,

$$\mu_{linkage_i}, \sigma_{linkage_i} = Q_{linkage}(X_{linkage_i}; \theta_e). \tag{5.10}$$

Where $Q_{linkage}$ is the recognition model of the VAE trained on the corresponding linkage dataset.

Now, for computing the pairwise distance between two linkages $X_{linkage_i}$ and $X_{linkage_j}$, we take the $L2$-norm between their corresponding mean latent vectors μ_i and μ_j. Using the $L2$-norm on latent representations instead of $L2$-norm on the original vectors has shown to yield better clustering for high dimensional data[69]. This results in the identification of K clusters and along with the corresponding cluster centers. Figure 1.2 shows cluster centers for the recognition and clustering task for a set of 100 four-bar linkages with $K = 4$.

Chapter 6

Conditional Generation of Linkages

6.1 Introduction

In this chapter we present an alternate End-to-End deep neural network architecture for Variational Synthesis of linkages. This End-to-End architecture is a Conditional-VAE (C-VAE), which approximates the conditional distribution of linkage parameters with respect to coupler trajectory distribution. The outcome is a probability distribution of kinematic linkages for an unknown coupler path or motion. This framework functions as a bridge between the current state of the art theoretical and computational kinematic methods and machine learning to enable designers to create practical mechanism design solutions.

This chapter presents a generative way of learning End-to-End synthesis for planar linkages using C-VAE. A case study of End-to-End Variational Synthesis using pure Deep Learning is presented in Section 6.3.1.

6.2 Conditional Variational Auto Encoders (C-VAE)

C-Variational Auto Encoders (C-VAE) [72] is a neural network architecture that learns to approximate conditional distribution of an observed data X given an observed property Y. Figure 6.1 shows a general architecture of a C-VAE. In this figure, X and Y are concatenated and fed to Recognition Model as the input. There are two hidden layers h_1 and h_2 in the Recognition Model, while the Generative Model has one hidden layer h_3. In the middle, there is feature space encoded by the

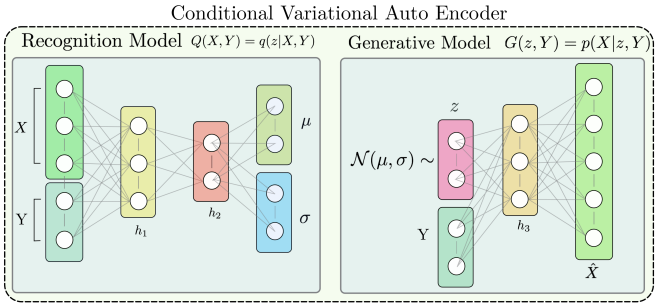

Figure 6.1: Recognition Model encodes the observed data X and observed property Y into probabilistic latent coding z of dimension much smaller than X. In this case, we assume a multivariate Gaussian distribution for z. Generative Model takes samples from this distribution and combines it with Y to generate output $\hat{X}$

variable z, which seeks to capture the salient features of the input data in a compact latent space. Given observed variables X and Y, Recognition Model computes a approximate probability distribution $q(z|X,Y)$ of the latent variable z as following:

$$\mu, \sigma = Q(X, Y; \theta_e), \tag{6.1}$$

$$q(z|X) = \mathcal{N}(\mu, \sigma). \tag{6.2}$$

Here, $\mathcal{N}(\mu, \sigma)$ is a multivariate Gaussian distribution function with mean μ and variance σ. The generative model is a function which is trained to maximize likelihood samples $\hat{X}$ by taking a sample from $z \in q(z|X,Y)$ and concatenating it with Y. Thus, X is given by,

$$\hat{X} = G(z, Y; \theta_g), \tag{6.3}$$

In this paper, we assume the prior probability distribution of $p(z)$ as a multivariate Gaussian with mean 0 and unit variance. The training task is to find parameters θ_g, θ_e of $G(z)$ and $Q(X)$, respectively that maximize an objective, which is a combination of generating maximum likelihood samples and obtaining posterior distribution of z (i.e., $q(z|X,Y)$) close to the true distribution $p(z)$. This is achieved by training the neural network models for minimizing the loss given by,

$$L = (\hat{X} - X)^2 + \sum_{j}^{z_{dim}} KL(q_j(z|X,Y)||p(z)) \tag{6.4}$$

Here, the first term represents reconstruction likelihood and the second term is called Kullback-Leibler divergence (KL divergence) [78] which is the measure of divergence in between the learned distribution $q(z|X,Y)$ and true prior distribution $p(z)$. Note that z_{dim} is the dimension of latent space. KL divergence, when $p(z)$ is a Gaussian distribution with zero mean and unit variance, is given by,

$$KL = \sum_{i}^{z_{dim}} {\sigma_i}^2 + {\mu_i}^2 - \log(\sigma_i) - 1, \tag{6.5}$$

where μ and σ are given by eq. (6.2); for further details please see [72]. The reconstruction likelihood imposes a penalty for not being able to reconstruct the original data, while the KL divergence term penalizes creation of an excessive number of clusters in the feature space.

Once the entire C-VAE is trained, the generative model can be used separately to perform kinematic synthesis. Parameters of this neural network are learned to map a concatenating vector comprising of latent space and label Y to a reconstruction, which would exhibit similar latent attributes z if passed through the recognition network. The architecture of this model starts with an input layer which receives the concatenated vector and passes through single or multiple layers of neurons culminating into the original size of the observed data. The middle layers are fully connected layers which upscale the input they receive from the previous layer. We apply Leaky ReLU activation function [81] on the output of each hidden layer. Leaky ReLu is given by,

$$ReLU(x) = max(\alpha x, x), \tag{6.6}$$

where α is a small constant, which we take to be 0.001 based on common machine learning practice.

6.2.1 State of the linkage

C-VAE discussed in section 4.6 requires a tuple (X,Y) to train, where X is an observed variable (in this case the entire linkage) and Y is an observed property

or condition (in this case the coupler curve). In theory, this formulation should work for any such tuple which has a strong correlation. We note that simply using the dimensional parameters of the mechanism, which describe link length and the fixed pivot locations of a mechanism would not be a suitable choice for the vector X as they would merely describe a set of unrelated and discrete parameters with no possible meaning associated with them. This demonstrates that divorcing kinematic knowledge from the ML will not give us meaningful answers. Instead, we formulate the observed variable X from the linkage parameters such that it should contain the information of the entire simulation of linkage in its current configuration. First, we orient and scale a linkage such that one of its fixed links has magnitude 1 and is parallel to the Cartesian x-axis. We uniformly sample locations of all the points of interest for m crank orientations sampled uniformly throughout the possible range. Next, we represent these locations in polar coordinates with origin at fixed pivot corresponding to the crank. Then, these coordinates are stacked together for all of the m orientations. In the case of four-bar linkage, we have three points of interest P_1, P_2, P_3 as shown in Fig. 6.2. Thus, the state tensor for four-bar linkage is given by,

$$X_{state} = \{r_{P1}, \theta_{P1}, r_{P2}, \theta_{P2}, r_{P3}, \theta_{P3}\}_{i=1}^{m}, \tag{6.7}$$

where r_{Pj} and θ_{Pj} are the radial and angular coordinates of point P_j.

We flatten this tensor to form a 600-dimensional vector $X_{fourbar}$. The dimension of state vector $X_{linkage}$ of linkage is equal to $2 \times m \times PoI$, where PoI is the number of points of interest for that linkage.

In the case of a Stephenson six-bar linkage, we have six points of interest $P_1, P_2, P_3, P_4, P_5, P_6$ as shown in Fig. 6.3. Thus, the state tensor X is given by,

$$X = \{r_{P1}, \theta_{P1}, r_{P2}, \theta_{P2}, r_{P3}, \theta_{P3}, r_{P4}, \theta_{P4}, r_{P5}, \theta_{P5}, r_{P6}, \theta_{P6}\}_{i=1}^{m}, \tag{6.8}$$

where r_{Pj} and θ_{Pj} are the radial and angular coordinates of point P_j. It should be noted that a tuple $(X = \{r_{P1}, \theta_{P1}, r_{P2}, \theta_{P2}, r_{P3}, \theta_{P3}, r_{P4}, \theta_{P4}, r_{P5}, \theta_{P5}, r_{P6}, \theta_{P6}\}_{i=1}^{m},)$

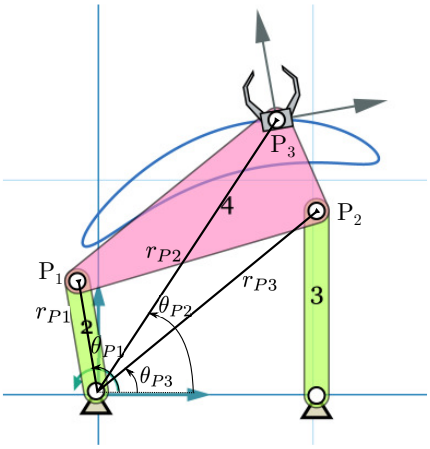

Figure 6.2: A four-bar linkage with the fixed link of unit magnitude and co-linear with X-axis. The polar coordinates of points P_1, P_2, and P_3 stacked together for m crank orientations constitute the state representation of the four-bar.

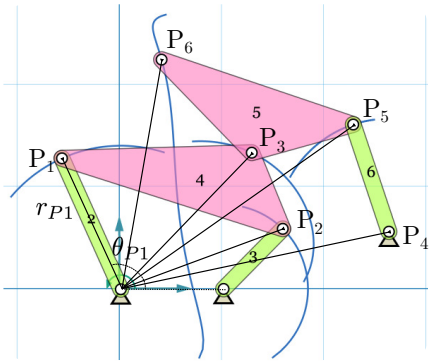

Figure 6.3: A Stephenson six-bar linkage with the fixed link of unit magnitude and co-linear with X-axis. The polar coordinates of points $\{P_i\}_{i=1}^{6}$ stacked together for m crank orientations constitute the state representation of the six-bar

Table 6.1: VAE and C-VAE Models : Architectures (FB=Four-bar, SC=Slider-Crank, SB=Six-Bar)

(X)	X Dim	Name	Enc Arch.	(z) dim	Dec Arch.	Y
FB M.	300	C-VAE-M3	(30)	3	(30)	6 Pts on P.
FB, SC, SB M.	300	C-VAE-M3	(30)	3	(30)	10 Pts on P.
FB Linkages	600	C-VAE-LF10	(300, 100)	10	(100, 300)	P.
SB Linkages	1200	C-VAE-LS15	(600, 300)	15	(300, 600)	P.

for any of the orientations can be used to construct the original mechanism back. X has dimensions $[m, 2 \times PoI]$, where PoI is the number of points of interest, whereas, Y is taken as the corresponding coupler path, which has dimension $2m$.

6.2.2 Training C-VAE for Mechanism Synthesis

We flatten X and Y to form vectors of dimensions $m \times 2PoI$ and $2m$, respectively. In order to train C-VAE, we pass a batch of X and Y to the network and compute gradients and losses. The training losses of C-VAE for four-bar and six-bar mechanisms are depicted in Fig. 6.4. It can be seen in the Fig. 6.4 that C-VAE for four-bar takes lesser time to train with better training accuracy. This observation is supported by the fact the six-bar linkages have higher complexities in the distributions of the parameters.

The model architectures and their training results are tabulated in Tables 6.1 and 6.2 respectively. In Table 6.1 where P. and M. signify path and motion, respectively. This Table also shows the details of each architecture, such as the number of neurons in each layer, the number of hidden layers.

Table 6.2: VAE and C-VAE Models : Training Losses

Name	Rec Loss	KL Loss
C-VAE-M3	7.21	2.32
C-VAE-M3	10.13	3.42
C-VAE-LF10	11.04	13.20
C-VAE-LS15	13.02	17.34

Figure 6.4: Reconstruction and KL Divergence losses for C-VAE-FB and C-VAE-SB. The architectural details of these models are given in Table 6.1.

6.3 End to End Variational Synthesis of Planar Linkages

C-VAE is trained to map the probability distribution of linkages to the shape of their corresponding coupler paths. C-VAE-10 takes a 200-dimensional vector Y and a sample from Gaussian distribution as an input and returns 600-dimensional vector $\hat{X}_{linkage}$ as output. From this 600 dimensional vector, we take the average location of each point of interest and construct the mechanism. This ensures that each generated sample results in a valid linkage. There is no guarantee that generated linkage should resemble the path Y, but C-VAE is trained to maximize that likelihood. Figures 6.5 and 6.6 depict some examples generated by C-VAE when a unseen coupler curve is passed as Y along with a multivariate Gaussian with zero mean and unit variance as z in Eq. (6.3). It is interesting to see the variations in the linkage parameters generated by C-VAE.

Combining this model with VAE for coupler trajectory can yield useful planar linkages with variations, making it an End-to-End deep learning model for the variational synthesis of planar linkages. Further investigation is necessary to make an accurate comparison of this End-to-End deep learning model with classical methods.

6.3.1 Case Study

Now, we use this trained model to generate conceptual designs for the prescribed gait rehabilitation path. C-VAE generates a conditional distribution of linkages for a given path. A hundred samples from this distribution were taken out of which forty of the samples possessed similar coupler curves as the task curve. From those samples, eleven linkages were selected by visual inspection and displayed in Fig. 6.7 along with the prescribed path Y.

It should be noted that by avoiding to formulate the problem as minimizing the fitting error between the task path and the coupler curve of the mechanisms, we are able to generate a multitude of mechanisms with desired structure. Figure 6.8 shows

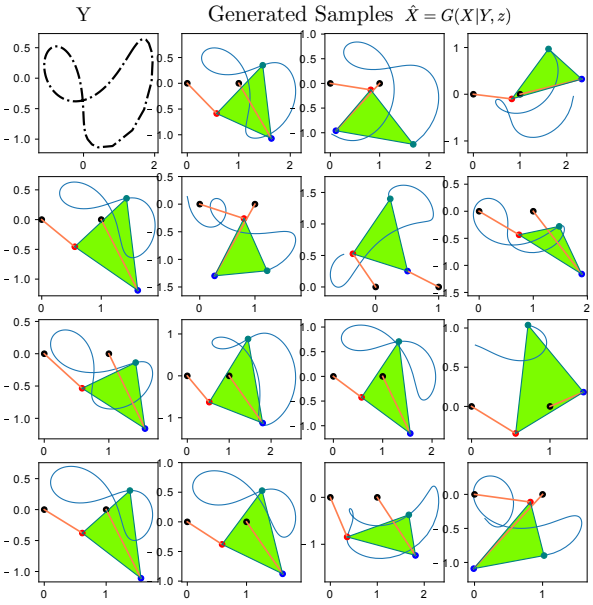

Figure 6.5: Sample linkages generated by C-VAE-LF10 when it is supplied with the conditional coupler curve Y and 10 dimensional Gaussian multivariate z. Architecture of this C-VAE is presented in Table 6.1.

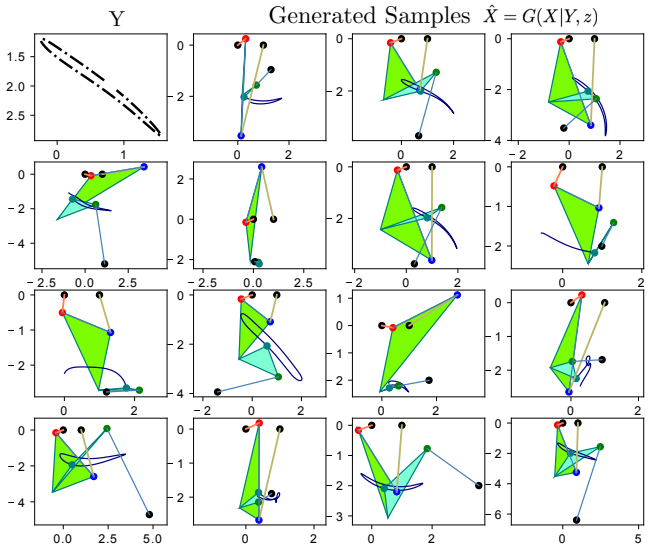

Figure 6.6: Stephenson Six-bars generated by C-VAE-LS15 (see Table 6.1) conditioned for coupler curve Y and 15 dimensional Gaussian multivariate z.

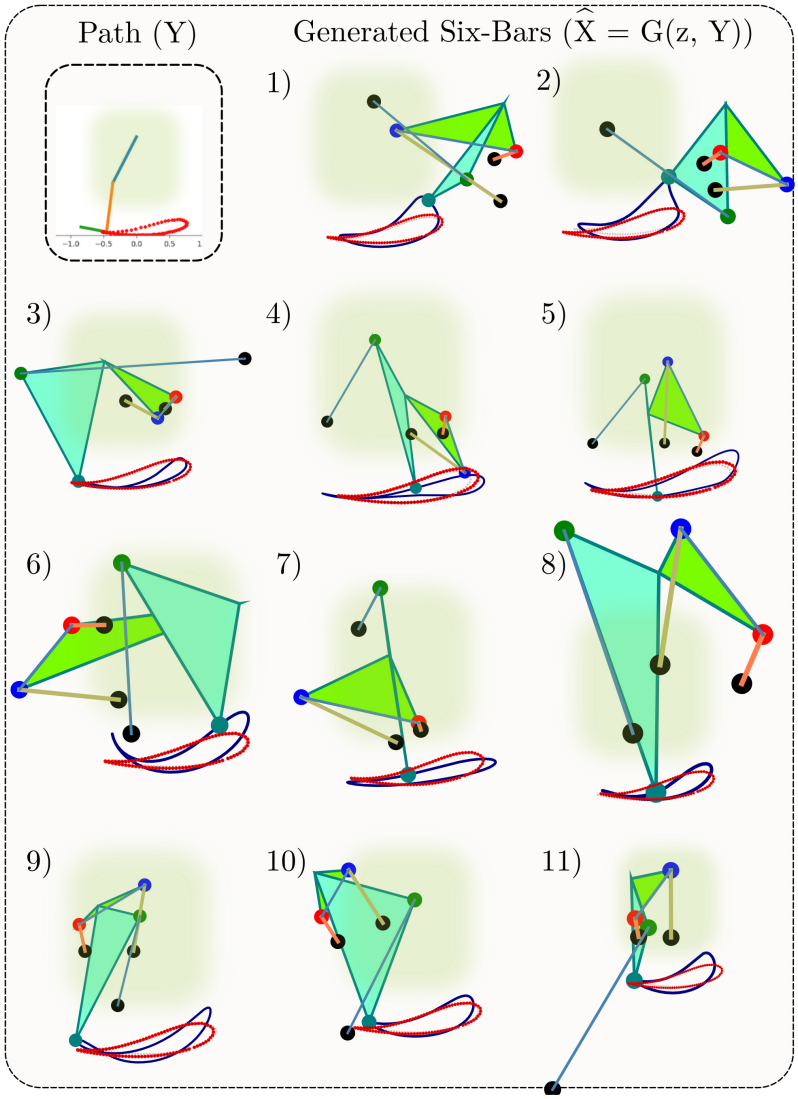

Figure 6.7: Stephenson Six-bars generated by C-VAE-LS15 (see Table 6.1) conditioned for coupler curve Y and 15 dimensional Gaussian multivariate z. The mechanisms having pivots in desired region are selected for the next step

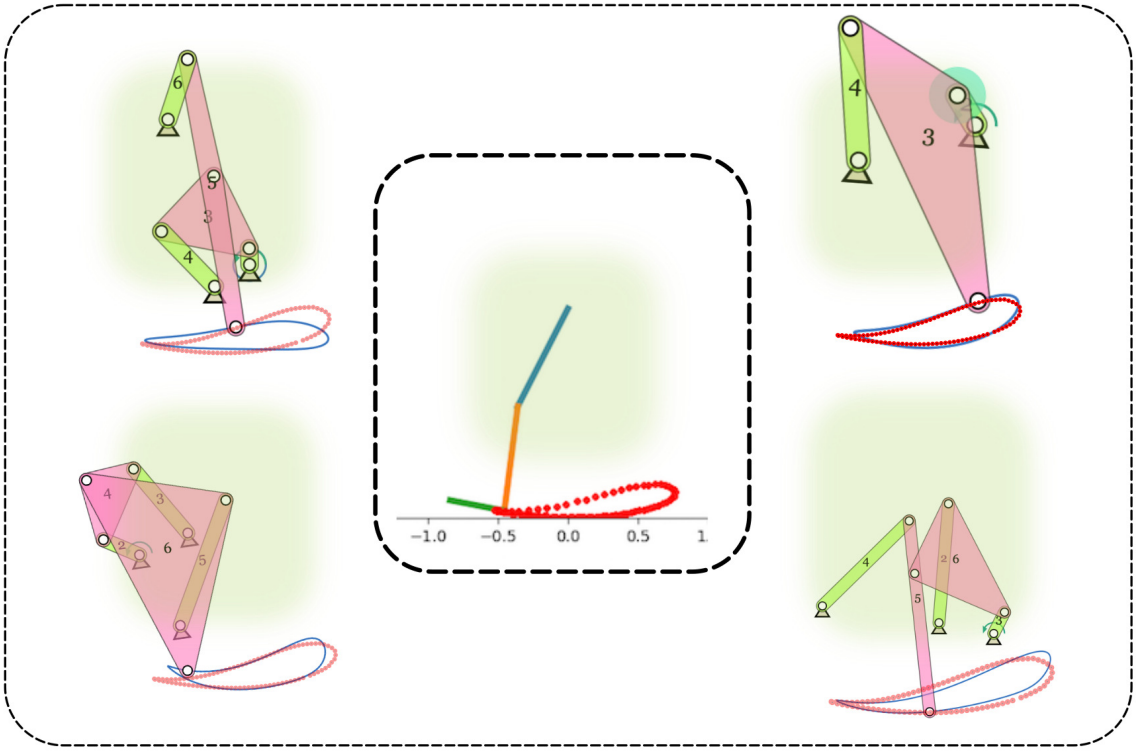

Figure 6.8: Collection of Feasible Six-Bar Mechanisms with desirable properties. It can be seen that their fixed pivots are in the desired shaded region

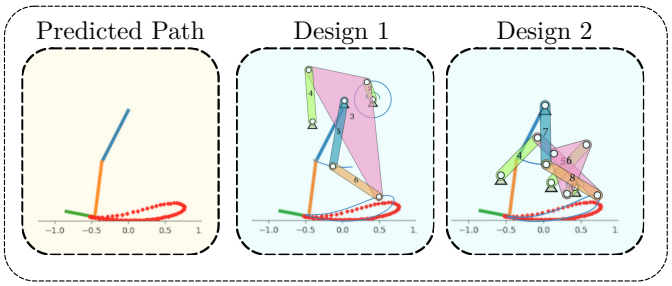

Figure 6.9: Final Linkage Concept Solutions

a few such feasible mechanisms. It can be seen that some of these mechanisms may have a worse fitting to the task path, but are better suited for this application as their fix pivots lie inside the desired region. Here, the number of solutions obtained is only limited by the sampling of the latent space; however, not every sample would produce a unique or useful linkage.

The concepts obtained in Fig. 6.8 can be further fine-tuned in detailed design phase. Along with Six-bars, C-VAE-FB model (see Table 6.1) was also used to find out the suitable four-bar linkages. The Fig. 6.9 shows one chosen solution concept along with an additional 2R link, which resembles the thigh and lower leg of the person. It can be seen that fixed pivots and link ratios have appropriate proportions with respect to the lower limb of the person. Using the C-VAE approach, we are able to generate a large sample of mechanisms that can satisfy the design requirement.

Chapter 7

Conclusion

The primary contribution of the dissertation is in the development of a data-driven computational framework that combines deep generative models with mechanism synthesis algorithms. Deep learning was used to learn the meaningful representations of linkage parameters and used in a novel way to enhance the users' design experience. This approach derives from the existing kinematic knowledge to create a new framework for mechanism synthesis, which solves problems that have had no good theoretical underpinning, such as defect-free generation, conditioning of the input, and contextual concept generation.

This is achieved by learning the probability distribution of various linkage parameters and their interdependence to perform useful inference tasks. The inference capabilities of the generative model are used to intelligently modify the synthesis task to enhance the prolificacy and robustness of precision-point based synthesis algorithms. We define this approach, where the input uncertainty is intelligently managed to generate a distribution of solutions, as **Variational Synthesis of Mechanisms**

In addition to the general framework, this paper also presents a novel image-based approach for path generation, which is particularly amenable to mechanism synthesis when the input from mechanism designers is deliberately imprecise or inherently uncertain due to the nature of the problem. It models the input curve as a probability distribution of image pixels and employs a probabilistic generative model to capture the inherent uncertainty in the input. In addition, it gives feedback on the

input quality and provides corrections for a more conducive input. The image representation allows for capturing local spatial correlations, which plays an important role in finding a variety of solutions with similar semantics as the input curve. The purpose is to obtain a diverse set of acceptable solutions instead of finding a single optimal solution for an inherently uncertain input.

The approach is independent of linkage topology. The approach is general enough to be extended to spatial linkages for which there are even fewer synthesis methods available. In addition, an alternate synthesis approach using End-to-End deep learning models is presented which captures the conditional distribution of linkage parameters and the task.

Bibliography

[1] 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.

[2] Deshpande, S. and Purwar, A., 2017, "A Task-driven Approach to Optimal Synthesis of Planar Four-bar Linkages for Extended Burmester Problem", ASME Journal of Mechanisms and Robotics, **9(6)**, p. 061005, 10.1115/1.4037801.

[3] 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 Journal of Computing and Information Science in Engineering, **19(1)**, p. 014501.

[4] Chase, T. and Mirth, J., 1993, "Circuits and Branches of Single-Degree-of-Freedom Planar Linkages.", ASME Journal of Mechanical Design, **115(2)**, p. 223–230.

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

[6] Wu, J., Ge, Q. J., Gao, F., and Guo, W. Z., 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-Transactions of the Asme, **3(3)**.

[7] Li, X., Wu, J., and Ge, Q. J., 2016, "A Fourier Descriptor-Based Approach to Design Space Decomposition for Planar Motion Approximation", ASME Journal of Mechanisms and Robotics, **8(6)**, pp. 064501–064501–5, 10.1115/1.4033528.

[8] Vasiliu, A. and Yannou, B., 2001, "Dimensional synthesis of planar mechanisms using neural networks: application to path generator linkages", Mechanism and Machine Theory, **36(2)**, pp. 299–310.

[9] Khan, N., Ullah, I., and Al-Grafi, M., 2015, "Dimensional synthesis of mechanical linkages using artificial neural networks and Fourier descriptors", Mechanical Sciences, **6(1)**, pp. 29–34.

[10] 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", Mechanism and Machine Theory, **44(6)**, pp. 1132–1143.

[11] 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 Journal of Computing and Information Science in Engineering, **12**, pp. 041003–041003.

[12] Zhao, P., Li, X., Purwar, A., and Ge, Q. J., 2016, "A Task-Driven Unified Synthesis of Planar Four-Bar and Six-Bar Linkages With R- and P-Joints for Five-Position Realization", ASME Journal of Mechanisms and Robotics, **8(6)**, pp. 061003–061003–8, 10.1115/1.4033434.

[13] McCarthy, J., 2017, "Want a patent? Try a Six-bar linkage", URL http://mechanicaldesign101.com/2016/07/26/want-a-patent-try-a-six-bar-linkage/.

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

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

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

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

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

[19] Lohse, P., 2013, Getriebesynthese: Bewegungsabläufe ebener Koppelmechanismen, Springer-Verlag.

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

[21] Erdman, A. G. and Sandor, G. N., 1991, Mechanism Design: Analysis and Synthesis, volume 1, Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.

[22] Purwar, A., Deshpande, S., and Ge, Q. J., 2017, "MotionGen: Interactive Design and Editing of Planar Four-Bar Motions via a Unified Framework for Generating Pose- and Geometric-Constraints", ASME Journal of Mechanisms and Robotics, 10.1115/1.4035899.

[23] Blaschke, W., 1911, "Euklidische Kinematik und nichteuklidische Geometrie", Zeitschr. Math. Phys., **60**, pp. 61–91 and 203–204.

[24] Grünwald, J., 1911, "Ein Abbildungsprinzip, welches die ebene Geometrie und Kinematik mit der raumlichen Geometrie verknupft"", Sitzber. Ak. Wiss. Wien, **120**, pp. 677–741.

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

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

[27] Ravani, B. and Roth, B., 1983, "Motion Synthesis Using Kinematic Mappings", Journal of Mechanisms Transmissions and Automation in Design-Transactions of the Asme, **105(3)**, pp. 460–467.

[28] Ravani, B. and Roth, B., 1984, "Mappings of spatial kinematics", Journal of Mechanisms Transmissions and Automation in Design-Transactions of the ASME, **106(3)**, pp. 341–347.

[29] Bodduluri, R. M. C. and McCarthy, J. M., 1992, "Finite position synthesis using image curve of a spherical four-bar motion", ASME J. of Mechanical Design, **114(1)**.

[30] Bodduluri, R., 1990, "Design and Planned Movement of Multi-Degree of Freedom Spatial Mechanisms", Ph.d. dissertation.

[31] Ge, Q. J. and Larochelle, P., 1999, "Algebraic motion approximation with NURBS motions and its application to spherical mechanism synthesis", ASME Journal of Mechanical Design, **121(4)**, pp. 529–532, article.

[32] Larochelle, P., 1994, "Design of cooperating robots and spatial mechanisms", PhD Dissertation University of California, Irvine.

[33] Larochelle, P., 1996, "Synthesis of planar RR dyads by constraint manifold projection", ASME Design Engineering Technical Conferences, ASME, Irvine, CA.

[34] Husty, M. L., Pfurner, M., Schrocker, H.-P., and Brunnthaler, K., 2007, "Algebraic methods in mechanism analysis and synthesis", Robotica, **25**, pp. 661–675.

[35] Hayes, M. J. D., Luu, T., and Chang, X.-W., 2004, Kinematic Mapping Application to Approximate Type and Dimension Synthesis of Planar Mechanisms, 9th Advances in Robotic Kinematics, Kluwer Academic Publishers, Dordrecht, The Netherlands.

[36] Hayes, M. J. D. and Rusu, S. R., 2011, "Quadric surface fitting applications to approximate dimensional Synthesis", 13th World Congress in Mechanism and Machine Science, Guanajuato, Mexico.

[37] Wu, J., Purwar, A., and Ge, Q. J., 2010, "Interactive Dimensional Synthesis and Motion Design of Planar 6R Single-Loop Closed Chains via Constraint Manifold Modification", ASME Journal of Mechanisms and Robotics, **2**, p. 31012(8 pages).

[38] Purwar, A. and Gupta, A., 2011, "Visual Synthesis of RRR- and RPR-legged Planar Parallel Manipulators using Constraint Manifold Geometry", ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering, ASME, Washington, DC, paper No. DETC2011-48830.

[39] Schröcker, H.-P., Husty, M. L., and McCarthy, J. M., 2007, "Kinematic Mapping Based Assembly Mode Evaluation of Planar Four-Bar Mechanisms", ASME Journal of Mechanical Design, **129(9)**, pp. 924–929.

[40] Burmester, L., 1886, Lehrbuch der Kinematik, Verlag Von Arthur Felix, Leipzig, Germany.

[41] Holte, J. E., Chase, T. R., and Erdman, A. G., 2000, "Mixed exact-approximate position synthesis of planar mechanisms", Journal of Mechanical Design, **122(3)**, pp. 278–286.

[42] Bawab, S., Sabada, S., Srinivasan, U., Kinzel, G. L., and Waldron, K. J., 1997, "Automatic synthesis of crank driven four-bar mechanisms for two, three, or four-position motion generation", Journal of Mechanical Design, **119(2)**, pp. 225–231.

[43] Venkataraman, S., Kinzel, G., and Waldron, K., 1992, "Optimal Synthesis of Four-Bar Linkages for Four-Position Rigid Body Guidance With Selective Tolerance Specifications", 1992 ASME Mechanisms Conference, Scottsdale, AZ, volume DE-vol. 46, p. 651–659.

[44] Larochelle, P., 2015, "Synthesis of Planar Mechanisms for Pick and Place Tasks With Guiding Positions", ASME Journal of Mechanisms and Robotics, **7(3)**.

[45] Lin, S., Liu, J., and Zhang, Y., 2015, "Planar Guidance Mechanism Synthesis based on Pole Curve Transformation", Proceedings of the 14th IFToMM World Congress, pp. 661–666.

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

[47] Bourrelle, J., Chen, C., Caro, S., and Angeles, J., 2007, "Graphical user interface to solve the burmester problem", IFToMM World Congress, pp. 1–8.

[48] Mariappan, J. and Krishnamurty, S., 1996, "A generalized exact gradient method for mechanism synthesis", Mechanism and Machine Theory, **31(4)**, pp. 413–421.

[49] Vallejo, J., Aviles, R., Hernandez, A., and Amezua, E., 1995, "Nonlinear Optimization of Planar Linkages for Kinematic Syntheses", Mechanism and Machine Theory, **30(4)**, pp. 501–518.

[50] Yao, J. and Angeles, J., 2000, "Computation of all optimum dyads in the approximate synthesis of planar linkages for rigid-body guidance", Mechanism and Machine Theory, **35(8)**, pp. 1065–1078.

[51] Kramer, S. N. and Sandor, G. N., 1975, "Selective Precision Synthesis - General Method of Optimization for Planar Mechanisms", Journal of Engineering for Industry-Transactions of the Asme, **97(2)**, pp. 689–701.

[52] Gogate, G. R. and Matekar, S. B., 2012, "Optimum synthesis of motion generating four-bar mechanisms using alternate error functions", Mechanism and Machine Theory, **54**, pp. 41–61.

[53] Cabrera, J. A., Simon, A., and Prado, M., 2002, "Optimal synthesis of mechanisms with genetic algorithms", Mechanism and Machine Theory, **37(10)**, pp. 1165–1177.

[54] Hegedüs, G., Schicho, J., and Schrocker, H.-P., 2015, "Four-Pose Synthesis of Angle-Symmetric 6R Linkages", ASME Journal of Mechanisms and Robotics, **7(4)**, pp. 041006–041006–7, 10.1115/1.4029186.

[55] Boyd, S. and Vandenberghe, L., 2004, Convex optimization, Cambridge university press.

[56] Wolfram Research, I., 2016, Mathematica, Wolfram Research, Inc., Champaign, Illinois.

[57] Sardashti, A., Daniali, H. M., and Varedi, S. M., 2013, "Optimal free-defect synthesis of four-bar linkage with joint clearance using PSO algorithm", Meccanica, **48(7)**, pp. 1681–1693, 200io Times Cited:11 Cited References Count:42.

[58] Ebrahimi, S. and Payvandy, P., 2015, "Efficient constrained synthesis of path generating four-bar mechanisms based on the heuristic optimization algorithms", Mechanism and Machine Theory, **85**, pp. 189–204.

[59] Bulatovic, R. R., Dordevic, S. R., and Dordevic, V. S., 2013, "Cuckoo Search algorithm: A metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage", Mechanism and Machine Theory, **61**, pp. 1–13.

[60] Buskiewicz, J., Starosta, R., and Walczak, T., 2009, "On the application of the curve curvature in path synthesis", Mechanism and Machine Theory, **44(6)**, pp. 1223–1239.

[61] Mcgarva, J. R., 1994, "Rapid Search and Selection of Path Generating Mechanisms from a Library", Mechanism and Machine Theory, **29(2)**, pp. 223–235.

[62] Wandling Sr, G. R., 2000, "Synthesis of mechanisms for function, path, and motion generation using invariant characterization, storage and search methods", Ph. d. thesis.

[63] 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.

[64] Chu, J. K. and Sun, J. W., 2010, "A New Approach to Dimension Synthesis of Spatial Four-Bar Linkage Through Numerical Atlas Method", Journal of Mechanisms and Robotics-Transactions of the Asme, **2(4)**.

[65] 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.

[66] Lewis, J. P., 1995, Fast normalized cross-correlation, volume 10 of Vision interface.

[67] Marimont, R. B. and Shapiro, M. B., 1979, "Nearest Neighbor Searches and the Curse of Dimensionality", Journal of the Institute of Mathematics and Its Applications, **24(1)**, pp. 59–70.

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

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

[70] Ward, J. H., 1963, "Hierarchical Grouping to Optimize an Objective Function", Journal of the American Statistical Association, **58(301)**, pp. 236–244.

[71] Deshpande, S. and Purwar, A., 2019, "A Machine Learning Approach to Kinematic Synthesis of Defect-Free Planar Four-Bar Linkages", ASME Journal of Computing and Information Science in Engineering, **19(2)**, pp. 021004–021004–10, 10.1115/1.4042325.

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

[73] Forsyth, D. A. and Ponce, J., 2002, Computer vision: a modern approach, Prentice Hall Professional Technical Reference.

[74] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, "Imagenet classification with deep convolutional neural networks", Advances in neural information processing systems, pp. 1097–1105.

[75] Bottou, L., 2010, "Large-scale machine learning with stochastic gradient descent", Proceedings of COMPSTAT'2010, Springer, pp. 177–186.

[76] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, "Learning representations by back-propagating errors", nature, **323(6088)**, pp. 533–536.

[77] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D., 2017, "Variational inference: A review for statisticians", Journal of the American Statistical Association, **112(518)**, pp. 859–877.

[78] Kullback, S. and Leibler, R. A., 1951, "On Information and Sufficiency", The Annals of Mathematical Statistics, **22(1)**, pp. 79–86, doi:10.1214/aoms/1177729694, URL https://doi.org/10.1214/aoms/1177729694.

[79] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research, **15**, pp. 1929–1958, URL http://jmlr.org/papers/v15/srivastava14a.html.

[80] Lloyd, S., 1982, "Least squares quantization in PCM", IEEE transactions on information theory, **28(2)**, pp. 129–137.

[81] Radford, A., Metz, L., and Chintala, S., 2016, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", Computing Research Repository, **abs/1511.06434**.