Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature_____

Date_____

Enumeration, Structural and Dimensional Synthesis of Robotics Hands: Theory and Implementation

By

Ali Tamimi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

College of Engineering, Idaho State University

December, 2015

COMMITTEE APPROVAL

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of ALI TAMIMI find it satisfactory and recommend that it be accepted.

_____Major Advisor

Dr. Alba Perez Garcia

_____ Member

Dr. Marco Schoen

_____ Graduate Faculty Representative

Dr. Steve Chiu

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Alba Perez, for her continuous support during my Masters study, for her patience, motivation and cooperation. Without her guidance and support my success would have been impossible. My thanks are due to my committee Dr. Marco Schoen and Dr. Steve Chiu for their help.

I would like to thank my colleagues at the Measurement and Control Engineering Research Center whose companionship was always a source of inspiration for me.

Let me also include my sister, Atousa, and my friends, Payandeh Ekrami and Hossein Panahi, whose sheer presence in my life has been a source of joy and strength. I must mention two great people, my parents, who have shaped me to be the individual that I am and thank for their love and support throughout my life.

Contents

List of Figures vii
List of Tables viii
Abstract ix
1 Introduction 1
1.1 Thesis Goals 1
1.2 Literature Review 1
1.3 Organization of the Thesis 3
2 Kinematics Background 4
2.1 Finite Displacement 5
2.1.1 Translation 5
2.1.2 Rotation 6
2.1.3 Spatial Displacement 7
2.2 Screw Displacement 8
2.3 Dual Quaternions 9
2.4 Kinematic Synthesis 10
3 Basic Concepts of Graph Theory 12
3.1 Introduction: 12
3.2 Graph 12
3.2.1 Degree of a vertex 12
3.2.2 Paths and cycles 13
3.2.3 Connected Graph and Subgraph 13
3.2.4 Directed Graph 15
3.2.5 Complete graph 16
3.2.6 Graph Isomorphism 16
3.3 Tree 18
3.3.1 Definitions 18
3.3.2 Weighted Tree 19
3.4 Representing a graph using Adjacency Matrix 20
4 Enumeration, Structural and Dimensional Synthesis of Robotic Hands 23
4.1 Introduction 23

4.1.1 Task Definition 23
4.1.2 Type Synthesis 24
4.1.3 Solvability 24
4.1.4 Dimensional Synthesis 24
4.2 Tree Topologies 24
4.3 Type Synthesis and Enumeration 27
4.3.1 Candidate Topology Search 28
4.3.2 Type Synthesis Enumeration 29
4.4 Solvability 32
4.4.1 Examples of Solvability 34
4.5 Dimensional Synthesis 37
4.5.1 Automatic forward Kinematic Equations 37
4.5.2 Exact Synthesis 38
4.5.3 Multiple Velocity Synthesis or Constrained-motion synthesis 39
5 Software 42
5.1 Kinematic Solver 42
5.2 Overall Software Architecture 42
6 Design Example 45
7 Conclusion 49
8 References 50

List of Figures

FIGURE 2-1 A RIGID BODY 4
FIGURE 2-2 A RIGID DISPLACEMENT 5
FIGURE 2-3(A) SHOWS PURE TRANSLATION AND (B) SHOWS PURE TRANSLATION. 6
FIGURE 2-4 REPRESENTATION OF SCREW AXIS 8
FIGURE 3-1 GRAPH 12
FIGURE 3-2 DISCONNECTED GRAPH 14
FIGURE 3-3 A GRAPH AND ONE OF ITS SUBGRAPH 14
FIGURE 3-4 A DIRECTED GRAPH 15
FIGURE 3-5 A COMPLETE GRAPH WITH 5 VERTICES 16
FIGURE 3-6 GRAPH1 ISOMORPHISM 17
FIGURE 3-7 GRAPH2 ISOMORPHISM 17
FIGURE 3-8 A TREE 18
FIGURE 3-9 A WEIGHTED GRAPH 20
FIGURE 3-10 ADJACENCY MATRIX FOR GRAPH 21
FIGURE 3-11ADJACENCY MATRIX FOR DIRECTED GRAPH 22
FIGURE 4-1 A MOTION TASK FOR A FOUR-FINGERED HAND, OBTAINED USING HUMAN HAND MOTION CAPTURE 23
FIGURE 4-2 A FIVE-FINGERED, TWO-PALM HAND TOPOLOGY. A) INDICATES THE NUMBERING OF THE EDGES AND B) INDICATES THE NUMBER OF JOINTS FOR EACH EDGE. 26
FIGURE 4-3 A FIVE-FINGERED, TWO-PALM HAND KINEMATIC SKETCH. 26
FIGURE 4-4 KINEMATIC SKETCH OF ALL NON-ISOMORPHIC CANDIDATE TOPOLOGIES WITH TWO FINGERTIPS AND SOLVABLE FOR THREE PRECISION POSITIONS. FROM THE LEFT TO RIGHT: 0-(2R, 2R), R-(2R, R) AND 2R- (R ,R). 32
FIGURE 4-5 THE SUBTREES WHICH NEED TO BE CHECKED FOR SOLVABILITY 33
FIGURE 4-6 TREE A) BEFORE AND B) AFTER CHANGING THE ROOT 33
FIGURE 4-7 TREE TOPOLOGY AND KINEMATIC SKETCH OF EXAMPLE 1. 35
FIGURE 4-8 TREE TOPOLOGY AND KINEMATIC SKETCH OF EXAMPLE 2. 36
FIGURE 4-9 TREE TOPOLOGY AND KINEMATIC SKETCH OF EXAMPLE 3. 36
FIGURE 4-10 TREE TOPOLOGY AND KINEMATIC SKETCH OF EXAMPLE 4. 37
FIGURE 4-11TOPOLOGY AND KINEMATIC SKETCH FOR 2-(2, 2) HAND 41
FIGURE 5-1 SOFTWARE ARCHITECTURE 43
FIGURE 6-1 GRAPH AND KINEMATIC SKETCH OF SELECTED TOPOLOGY 46
FIGURE 6-2 THE POSITION TASK. COLORS CORRESPOND TO EACH POSITION OF ALL THREE FINGERTIPS. 47
FIGURE 6-3 HAND DESIGN USING THE TOPOLOGY 2-(1-(2,2),2) 48
FIGURE 6-4 2-(1-(2,2),2) HAND DESIGN FOR THE SPECIFIC POSITION 48

List of Tables

TABLE 4-1 TYPE SYNTHESIS RESULTS FOR SELECTED INPUTS 29
TABLE 4-2 EXAMPLES OF TYPE SYNTHESIS 31
TABLE 4-3 EXAMPLES OF SOLVABILITY 35
TABLE 6-1 SOLVABLE TOPOLOGIES FOR 5 POSITIONS, 3 FINGERTIPS 45
TABLE 6-2 SOLVABLE TOPOLOGIES FOR 5 POSITIONS, 4 AND 5 FINGERTIPS 45
TABLE 6-3 DIMENSIONAL SYNTHESIS SOLVER RESULT 47

Abstract

Designing robotic hands for specific tasks could help in the creation of optimized end-effectors for grasping and manipulation. However, the systematic design of robotic hands for a simultaneous task of all fingertips presents many challenges. In this work the algorithms and implementation of an overall synthesis process is presented, which could be a first step towards a complete design tool for robotic end-effectors.

Type synthesis for a given task and number of fingers, solvability and dimensional synthesis for arbitrary topologies are developed and implemented. The resulting solver is a powerful tool that can aid in the creation of innovative robotic hands with arbitrary number of fingers and palms. Several examples of type synthesis, solvability calculations and dimensional synthesis are presented.

An innovative design process for designing robotic hands for specific task is illustrated in this application. Consideration of velocities, acceleration and force at finger tips is required to evaluate grasping and manipulation abilities in order to have a complete design.

1 Introduction

1.1 Thesis Goals

The goal of this thesis is development of algorithms for type synthesis, solvability and dimensional synthesis and apply them for different robotic hand topology. The work will help to find robotic hand topology for defined task, check the solvability of hand ,and apply dimensional synthesis for selected hand.

In order to achieve this goal, first the algorithm is defined for finding solvability of robotics hand topology. Then this algorithm is applied as a part of type synthesis algorithm to find robotics hand topology for defined tasks. Finally, dimensional synthesis is applied for finding the needed positions.

In this research LUA, a programing language in Linux Environment and C++ is used to reach the goal. The program has three main parts including type synthesis part which shows all possible hand topology can be used for defined task, solvability part which finds that the topology is solvable or not, dimensional synthesis part which applied mathematical and genetic algorithm solvers to solve forward kinematic equations.

1.2 Literature Review

Robotic hands are mechanical linkages where a common set of links spans a number of serial chains. Among the variety of robotic end-effectors, those generally defined as robotic hands are considered suited not only for grasping, but also for some dexterous manipulation.

When considering applications in robotic grasping and manipulation of grasped objects, many aspects of robotics have to converge, including sensing, identification, learning and planning, to name a few, and notable results are being obtained in these fields. One field that has not attracted so much attention is the systematic methodology for the physical embodiment of the

robotic end effector; however, its effect on the successful completion of the task may be considerable.

The design of end-effector robotic tools has focused on three different strategies [1], which yield very different designs: anthropomorphism, designing for grasping tasks, and designing for dexterous manipulation. The anthropomorphic hands are constrained in their design, but they are considered a straightforward solution for human environment and human manipulation task mapping [2], [3]. Of the other two design strategies, hands oriented to grasping tasks are usually simpler or under actuated; new efforts are being devoted to obtain under actuated or simple hands with some degrees of dexterity [4], [5]. Hands for in-hand manipulation tend to be more complex, especially if a wide range of manipulation actions are targeted, however most of them are anthropomorphic in design. For a current review on design efforts of anthropomorphic hands, see [6]. More recently, the design process for robotic hands has started receiving some attention [7], [8], [9].

A task-based, systematic design process, needs to consider both the enumeration of topologies, or structural synthesis, and the dimensioning of the selected topologies, the dimensional synthesis, followed by a stage of detailed design and implementation.

The literature in type or structural synthesis is vast, especially for linkages with closed loops, which present bigger challenges in their classification. Type or structural synthesis is based on subgroups of motion, following [10] and on the use of screw theory, such as [11], among others [12], combined with graph theory for the enumeration and classification. Most of the current methods are based on defining subgroups of motion or subspaces of potential velocities for the system. A task-based approach for the structural synthesis needs to take into account the shape of the desired workspace, or kinematic task. Pucheta [13] applied a graph theory-based method and precision position method consecutively for planar linkages and for multiple kinematic tasks. Results on structural synthesis of hands based on mobility while grasping an object has been studied in [14] and [15], and more recently in [16].

For the dimensional synthesis stage, most of the research has focused on the design of individual, under actuated fingers; see [17] and [18]. The first tool for the systematic dimensional synthesis of complete multi-fingered robotic hands for given manipulation tasks, up to the authors'

knowledge, was developed by Simo-Serra et al. [19]. This tool allows to design multi-fingered robotic hands with a set of common wrist joints and a palm branching in different number of fingers, see also [20], and [21] for its theoretical development.

In this work, a complete design methodology is presented for arbitrary robotic hands. This includes topology enumeration and the corresponding arrays defining the topology, structural synthesis for an input task, and dimensional synthesis for hands that can present several splitting stages. Theoretical aspects, algorithmic implementation and computational aspects are included. The aim is to integrate these in a design tool to help in the creation of robotic hands tailored to specific applications.

1.3 Organization of the Thesis

The thesis is organized as follows. Chapter 1 discusses the theme of the thesis and the literature review. Chapter 2 provides the mathematical background in kinematics and synthesis. Chapter 3 provides the graph theory background. Chapter 4 focuses on enumeration, structural and dimensional synthesis of robotic hands. In Chapter 5, the software is explained. Chapter 6 focuses on comprehensive example. Chapter 7 provides conclusions and future works.

2 Kinematics Background

In this chapter, the kinematic background needed for synthesis process is reviewed. As it is commonly accepted, "Kinematics is defined as the study of the motion, regardless of the force causing it and caused by it" [22]. The study of motion includes the study of derivatives of it which are mainly velocity and acceleration [22]. The subject is modeled as rigid body which is "a set of particles such that the distance between them remains fixed" [22]. In other words, in addition of concepts such as position, velocity and acceleration, the concepts including orientation, angular velocity and angular acceleration can be defined. For defining a position of rigid body three positions are needed and each of them includes three parameters. It means that totally nine parameters are needed. Since the distances between the positions must remain constant, these three positions are dependent to each other [22]. As a result of these constraints, only six parameters are needed to define. Figure 2-1 [22] shows a rigid body which three positions are defined on that. The six needed parameters are given by d and [R].

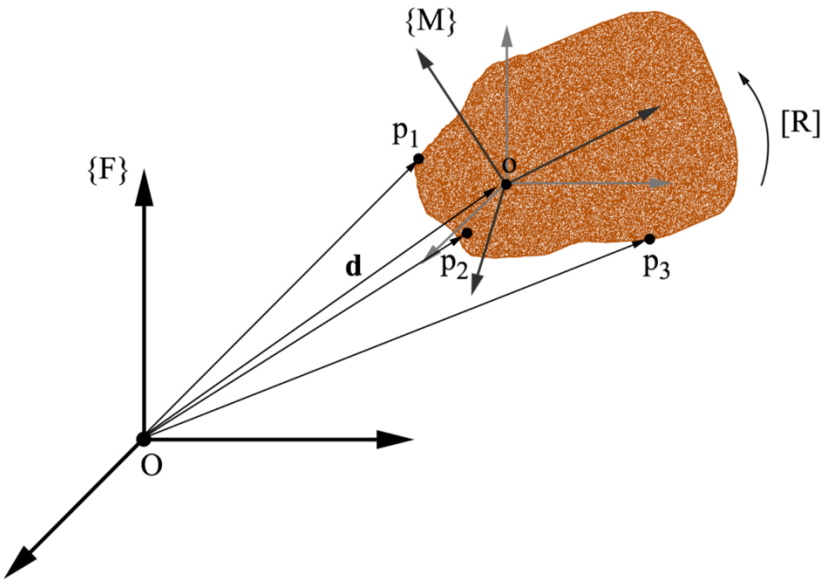


Figure 2-1 a rigid body

2.1 Finite Displacement

In order to define the movement of a rigid body, a coordinate frame is attached to the rigid body. The coordinates of points of the rigid body is fixed in the attached coordinate frame. In Figure 2-2 [22], there are two different coordinates respect to frame F and frame M. A finite motion is called a displacement [22]. A displacement can be a translation, a rotation, or both of them.

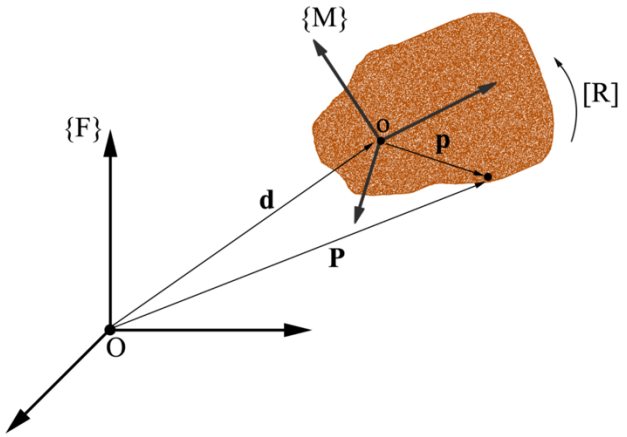


Figure 2-2 a rigid displacement

2.1.1 Translation

Translation is given by moving all of the points of a rigid body with the same amount in the same direction [22]. Equation 2.2 shows the representation of the translation. The translation vector **t** is applied to initial position **x** of the body in the space and results in new position which is called **X**.

$$\mathbf{X} = \mathbf{x} + \mathbf{t} \qquad (2.2)$$

where,

X is a coordination after the translation

x is a coordination before the translation

t is the translation vector including direction and magnitude.

2.1.2 Rotation

Rotation is a type of displacement in which a subspace of points can remain fixed while the rest of the points move by different amounts in different directions according to their location with respect to fix points [22]. Equation 2.3 shows representation of the rotation. The rotation matrix [R] is applied to initial position **x** of the body in the space and results in new position which is called **X**.

$$\mathbf{X} = [R]\mathbf{x} \qquad (2.3)$$

where,

X is a coordination after the rotation

x is a coordination before the rotation

R can be a rotation about x, y or z axes, or it can be a product of two or more.

Figure 2-3 [22] a) shows a translation and b) shows a rotation

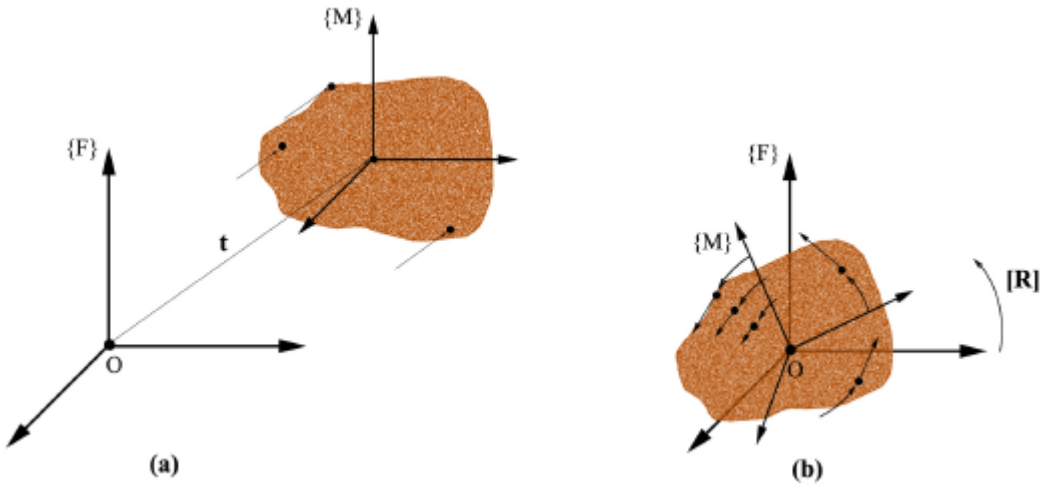


Figure 2-3(a) shows pure translation and (b) shows pure translation.

Equation 2.4 to 2.6 show the rotations about X, Y and Z.

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad (2.4)$$

$$[R_y] = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad (2.5)$$

$$[R_z] = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

2.1.3 Spatial Displacement

Spatial displacement is the combination of a rotation and a translation about an axis [22]. Equation 2.7 is a spatial displacement which consists of rotation of θ about X-axis and a translation.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} \quad (2.7)$$

The equation 2.7 can be written as equation 2.8

$$X = [R(\theta), d]x \quad (2.8)$$

where

X is a coordination after the displacement

x is a coordination before the displacement

[R(θ), d] is displacement matrix including translation and rotation.

Displacement matrix is expressed as equation 2.9.

$$\begin{pmatrix} X \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & \cos(\theta) & -\sin(\theta) & d_y \\ 0 & \sin(\theta) & \cos(\theta) & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \quad (2.9)$$

The above matrix is the 4×4 homogeneous transformation matrix. The fourth column is added to include the translation vector in the linear operation.

2.2 Screw Displacement

A displacement in space can be written as a screw displacement. The screw displacement is given by rotation and translation about an axes [22]. The screw axis is represented by six parameters. For defining the direction and location of the line, four parameters are needed and two other parameters are needed for defining the rotation and slide values. Figure 2-4 [22] shows the representation of screw axis.

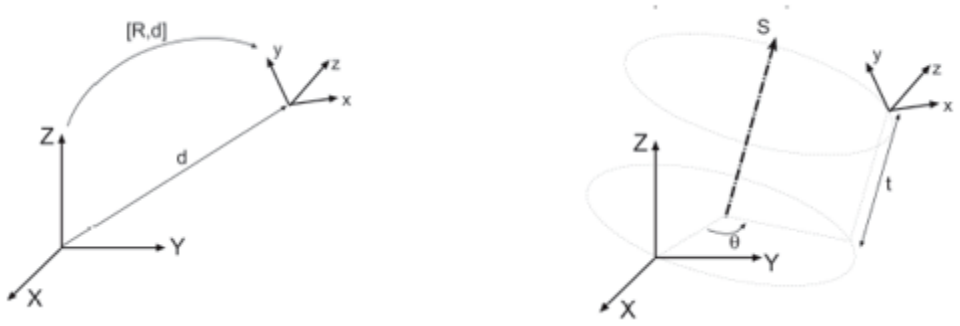


Figure 2-4 Representation of Screw axis

Equation 2.10 to 2.12 show the screw displacements for X, Y and Z.

$$X(\alpha, a) = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.10)$$

$$Y(\beta, b) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & b \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.11)$$

$$Z(\theta,d) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.12)$$

2.3 Dual Quaternions

The homogeneous matrices method has been used to represent the rigid body displacement in three dimensional space. The more efficient method in terms of computational speed, mathematical robustness and interpolation is the dual quaternions method. The main information which is applied by the dual quaternions method is the axis of the rotation and the translation vector and rotation angle. The dual quaternion for the given screw axis, *S*, is given by $S= s+ \varepsilon s^0$,

where $\varepsilon^2 = 0$ and, s is the direction of the axis and s^0 is the moment of the axis. Generally, the displacement of the dual quaternion is represented as below [22].

$$\widehat{D}\ (\hat{\theta}) = \begin{pmatrix} \sin\left(\frac{\hat{\theta}}{2}\right) S_x \\ \sin\left(\frac{\hat{\theta}}{2}\right) S_y \\ \sin\left(\frac{\hat{\theta}}{2}\right) S_z \\ \cos\ (\hat{\theta}/2) \end{pmatrix} \qquad (2.13)$$

The conjugate of the dual quaternion is

$$\widehat{D}^*(\hat{\theta}) = \begin{pmatrix} -\sin\left(\frac{\hat{\theta}}{2}\right) S_x \\ -\sin\left(\frac{\hat{\theta}}{2}\right) S_y \\ -\sin\left(\frac{\hat{\theta}}{2}\right) S_z \\ \cos\ (\hat{\theta}/2) \end{pmatrix} \qquad (2.14)$$

where, $\cos(\hat{\theta}/2) = \cos(\frac{\theta}{2}) + \varepsilon\ (-\frac{d}{2}\sin(\frac{\theta}{2}))$ and $\sin(\frac{\hat{\theta}}{2}) = \sin(\frac{\theta}{2}) + \varepsilon\ (\frac{d}{2}\cos(\frac{\theta}{2}))$

2.4 Kinematic Synthesis

From a kinematics point of view, a robot is a mechanical system that can be perform a number of tasks involving movement under automatic control [22]. Using the combination of translations and rotations, a robot can move in a six dimensional space.

A mechanical system can be defined as a series of rigid links connected by joints. The degree of freedom of a robot is called mobility which is the number of required independent parameters to specify the positions of a rigid body relative to a base frame. The degree of the freedom of a robot is calculated using equation 2.1 [22].

$$\mathrm{M} = 6(\mathrm{n}-1) - \sum_{i=1}^{j}(6-f_i) \qquad (2.1)$$

where,

n is the number of links in the serial chain.

j is the number of joints in the serial chain.

f_i is the degree of freedom of the joint number **i**. **i** can be any number from 1 to j.

Kinematic synthesis is the process in which an articulated system created and calculated to apply a motion. The main method to design the equations for the dimensional kinematic synthesis is setting up the forward kinematics equations of the robot.

The Forward Kinematic equation (2.15) calculate a relative displacement from initial position to final position. This displacement can be a combination of rotations and slide along the axis S.

$$\widehat{D}\left(\Delta\widehat{\Theta}\right) = cos\frac{\widehat{\Psi}}{2} + sin\frac{\widehat{\Psi}}{2}S = e^{\frac{\Delta\widehat{\theta}_1 S_1}{2}} e^{\frac{\Delta\widehat{\theta}_2 S_2}{2}} \dots\dots\dots\dots e^{\frac{\Delta\widehat{\theta}_k S_k}{2}} \quad (2.15)$$

3 Basic Concepts of Graph Theory

3.1 Introduction:

In this chapter the concepts of graph theory are presented. First, the definitions related to graphs are explained. It is followed by presenting a special type of graph which is called tree graph. Finally, the representation of graphs using adjacency matrix is explained.

3.2 Graph

The word "graph" is derived from the fact that regards it as graphical notation [23]. Each graph consists of two different sets. The first set defines the vertices of the graph and the second set consists of members that includes two members of vertices set. Those two members determine the end vertices of each edge. Each graph is represented by vertices and edges sets. For example, Figure 3-1 shows a graph (G) with the following vertices (V) and edges (E) sets which is represented as G = (V, E).

$V = \{1, 2, 3, 4, 5\}$ $E = \{\{1,2\}, \{1,5\}, \{2,3\}, \{2,5\}, \{3,4\}, \{4,5\}\}$

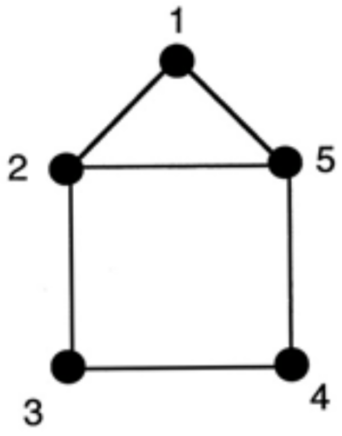


Figure 3-1 Graph

3.2.1 Degree of a vertex

The degree of a vertex or valency of a vertex [24] is the number of edges of the graph which are incident with that vertex. A vertex with degree of zero, two, and three is called isolated

vertex, binary vertex, and ternary vertex respectively. For the graph shown in Figure 3-1 the degree of vertex 1 is 2. The degree of vertices 2, 3, 4 and 5 are equal 3, 2, 2, and 3 respectively. Since each edge is incident with two vertices, the sum of degrees of vertices is equal 2 times the number of edges.

$$\sum_{v \in V} \delta(v) = 2E \qquad (3.1)$$

Where V is the set of vertices, $\delta(v)$ is the degree of the vertex v, and E is number of edges.

3.2.2 Paths and cycles

For defining path and cycle, first the adjacent vertices should be defined. Vertex *u* and vertex *v* are called adjacent if there is an edge between them. Path or walk is a sequence of vertices and edges should be beginning with a vertex and ending with a vertex [24]. In other words, a path can not begin or end with an edge. The length of a path is the number of edges which are included in the path. A path with length *n* is defined with a sequence including *n+1* vertices and *n* edges. A path also could be defined using only a sequence of vertices. A trail is a specific path in which all edges are distinct. In a path, if each vertex of the graph appears at most one time except one of them, which appears at beginning and ending the path, the path is called cycle or circuit [23]. The length of a circuit is at least 3. For the graph shown in Figure 3-1, (v (1), e ({1,2}), v (2), e ({2, 5}), v (5), e ({4, 5}), v (4)) is a path and (v (2), e ({2,3}), v (3), e ({3, 4}), v (4), e ({4, 5}), v (5), e ({2, 5}), v (2)) is a cycle.

3.2.3 Connected Graph and Subgraph

For defining a connected graph, first the connected vertices should be defined. Two vertices are connected if there is at least one path between them. They do not need to be adjacent for being connected. The graph is connected if each vertex of the graph is connected to all others vertices [23]. The degree of each vertex in a connected graph is at least one. If there is a vertex with degree zero in a graph, it means that the graph is not connected. The graph shown in Figure 3-1 is connected. Figure 3-2 [23] shows a graph that is disconnected.

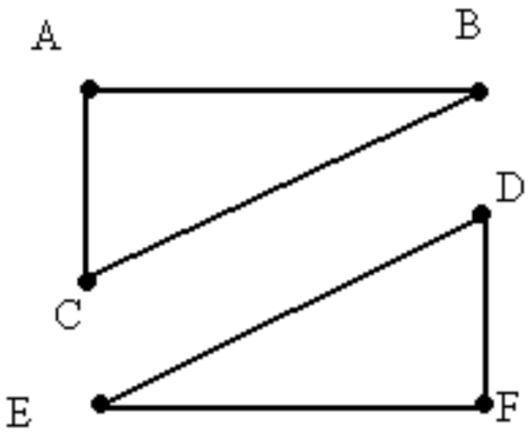


Figure 3-2 Disconnected Graph

A subgraph of a graph is a graph whose vertices form a subset of vertices of the main graph and its edges are a subset of the edges of the main graph. In other words, the subgraph is made by deleting a number of edges and vertices of a graph [23]. For building a subgraph, deleting at least one edge is necessary. But all the vertices could be saved. If a vertex is removed, all the edges which are incident to that vertex should be removed. Figure 3-3 [25] shows a graph at the left and one of its subgraphs at the right. The main graph vertex set is V= {a, b, c, d} and its edges set is E= {{a, b}, {a, d}, {b, c}, {b, d}, {c, d}}. The subgraph is made by removing vertex "d" and all the edges which are incident with vertex "d". The subgraph vertex set is V= {a, b, c} and its edges set is E= {{a, b}, {b, c}}.

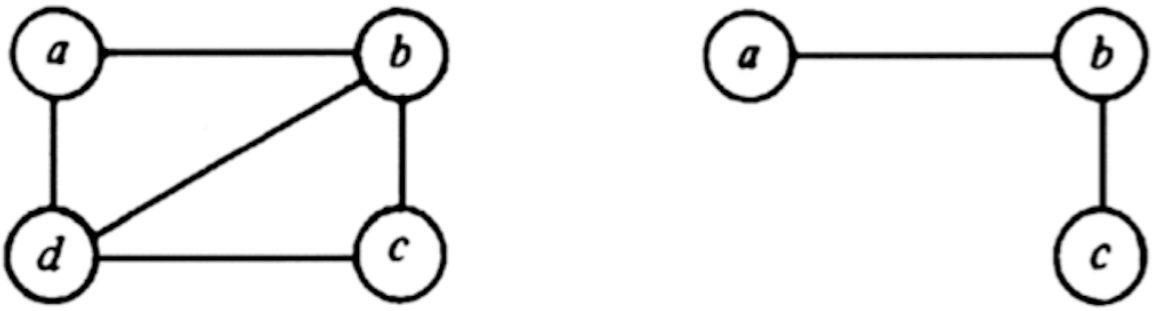


Figure 3-3 a graph and one of its subgraph

3.2.4 Directed Graph

If all the edges of a graph have direction the graph is called a directed graph. In a directed graph the order of the vertices which are the ends of an edge is important. For example, E= {{u, v}} and E= {{v, u}} are different. There are two different degrees of a vertex in directed graph. Indegree of vertex (v) is the number of edges of the graph which starts from other edges and end in vertex (v). In other words, the direction of edges is into vertex (v). Outdegree of vertex (v) is the number of edges of the graph which starts from vertex (v) and end at other vertices. In other words, the direction of edges is from vertex(v) to other vertices. If a vertex has a positive outdegree and the indegree of that is equal to zero it is called *source vertex*. If a vertex has a positive indegree and the outdegree of that is equal to zero it is called *sink vertex*. If for each vertex of graph, the indegree is equal to the outdegree, the graph is called a balanced graph. If there is only one source vertex in a graph, it is called root vertex. Figure 3-4 [25] shows a directed graph. There are four vertices and five edges in this graph (V= {1, 2, 3, 4,} and E = {{1, 2}, {1, 3}, {1, 4}, {2, 3}, {4, 3}}). Vertex 1 has indegree equal to 0 and outdegree equal to 3. It means that vertex 1 is a source vertex and since it is the only source vertex in graph it is also root. Vertices 2 and 4 has indegree and outdegree equal 1. Vertex 3 has a indegree equals 3 and outdegree equals 0. It means that vertex 4 is a sink vertex.

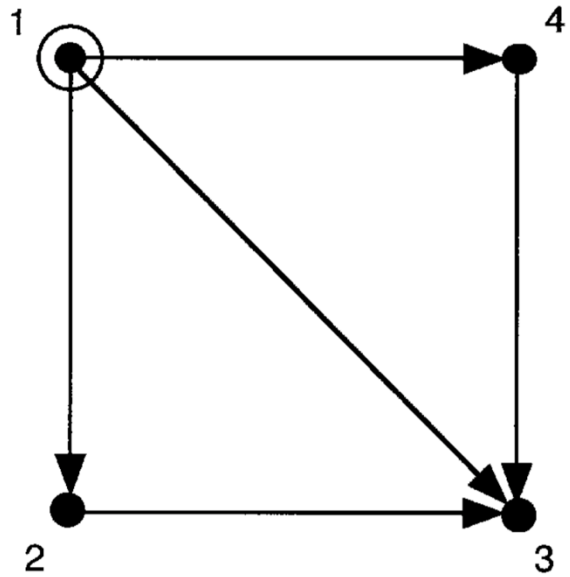


Figure 3-4 a directed graph

3.2.5 Complete graph

If every vertex in a graph is connected to all other vertices of the graph by one edge, the graph is called a complete graph [25]. A complete graph with n vertices has *n(n-1)/2* edges. The symbol for a complete graph with n vertices is K_n. Figure 3-5 [25] shows a K_5 graph. This graph has 10 edges.

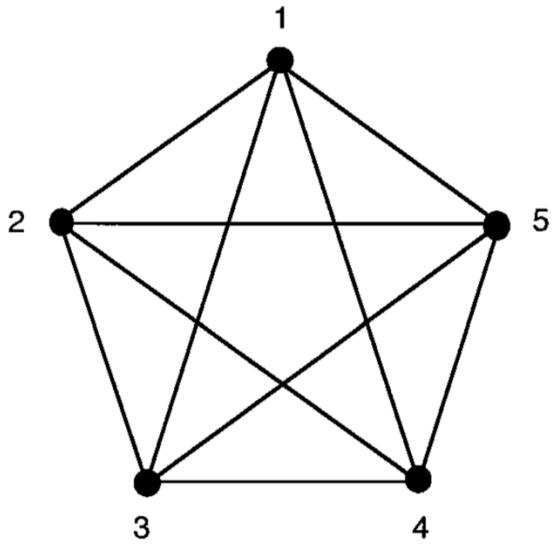


Figure 3-5 a complete graph with 5 vertices

3.2.6 Graph Isomorphism

Consider two graphs H and G which have the same number of edges and vertices. If there is a one-to-one relation between the edges and vertices of H and G, the graphs are called isomorphic [25]. In other words, each vertex in graph G corresponds to a vertex in graph H and each edge in graph G corresponds to an edge in graph H, there is an isomorphism between them. The graphs shown in Figure 3-6 [25] and 3-7 [25] are isomorphic. Both graphs have 6 vertices and 9 edges. The vertices correspond to each other as follow:

Vertex number 1 in Figure 3-6 corresponds to vertex number 1 in Figure 3-7. Vertex number 2 in Figure 3-6 corresponds to vertex number 4 in Figure 3-7. Vertex number 3 in Figure

3-6 corresponds to vertex number 2 in Figure 3-7. Vertex number 4 in Figure 3-6 correspond to vertex number 5 in figure 3-7. Vertex number 5 in Figure 3-6 corresponds to vertex number 3 in Figure 3-7. Vertex number 6 in figure 3-6 corresponds to vertex number 6 in Figure 3-7.

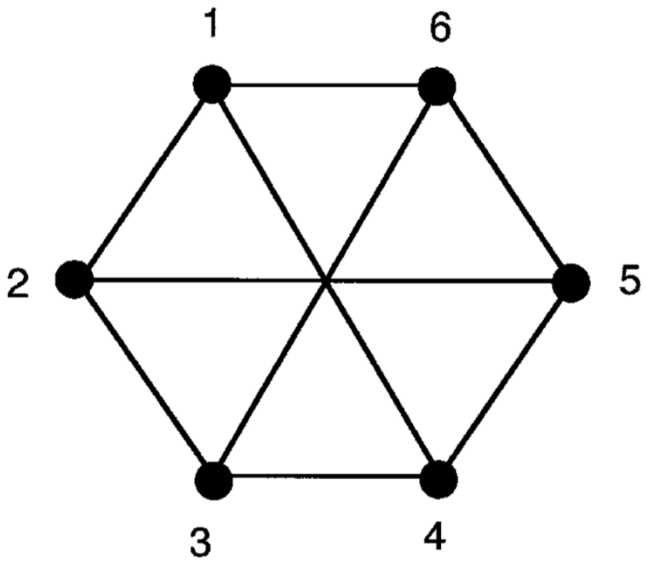


Figure 3-6 graph1 isomorphism

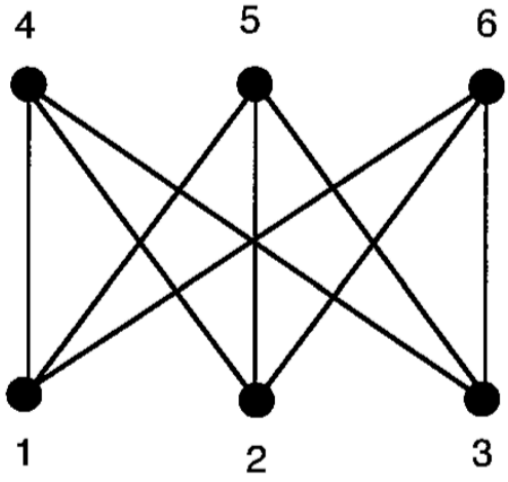


Figure 3-7 graph2 isomorphism

3.3 Tree

A connected graph with no cycles is called a tree [25]. A tree has following properties:

- There is exactly one path between any two vertices of the tree.
- A tree with v vertices has *v-1* edges. In other words, the number of vertices in the tree is exactly 1 more than the number of edges.
- If two non-adjacent vertices in the tree are connected to each other, the output is a graph with one cycle.

3.3.1 Definitions

There are two types of vertices in a tree. A leaf is a vertex with degree of one and a vertex whose degree is at least equal to two is called internal vertex [26]. If the tree has a root, root vertex can be a degree 1 vertex and it is not counted as a leaf.

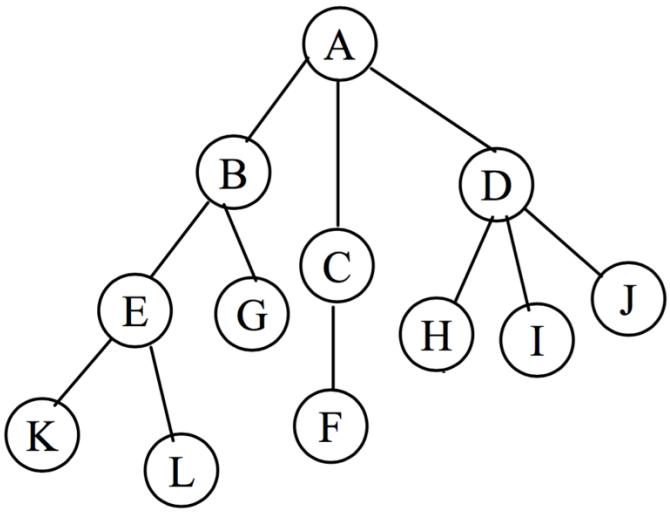


Figure 3-8 A tree

The tree shown in Figure 3-8 [26] has 12 vertices. Vertex A is the root of the tree. Vertices B, C, D, and E are internal vertices and vertices K, L, G, F, H, I, and J are leaves.

Level of a vertex: level of a vertex is related to the length of the path between the root and each vertex. The level of the root vertex is equal to one. Those vertices that have a path with length of one with the root are in level two and so on. In the graph shown in Figure 3-8 [26], vertex A is in level one. Vertices B, C, and D are in level two. Vertices E, G, F, H, I, J are in level three. Finally, the vertices K and L are in level four.

Parent: each vertex except the root has a parent. The parent for vertex in level *n* is a vertex in level *n-1* which has a path with length of one to that vertex. In the tree shown in Figure 3-8 [26], the parent of vertices B, C, and D is vertex A. Parent of E and G is B. Parent of F is C. Parent of H, I, and J is D. Finally, Parent of K and L is E. Vertex A is the root vertex and does not have a parent.

3.3.2 Weighted Tree

There is a tag (number) for each edge of the tree in a weighted tree, which shows the weight of the edge [27]. Sometimes a weighted graph is called a labeled graph. The applications of weighted graphs are in different areas such as Computer Network, Optimization, etc. One of the most popular problems which use weighted graphs is shortest-path problems such as the travelling sales man problem [27]. Figure 3-9 shows a weighted graph. Figure 3-9 [27] shows a weighted tree which is used for uniform cost search application.

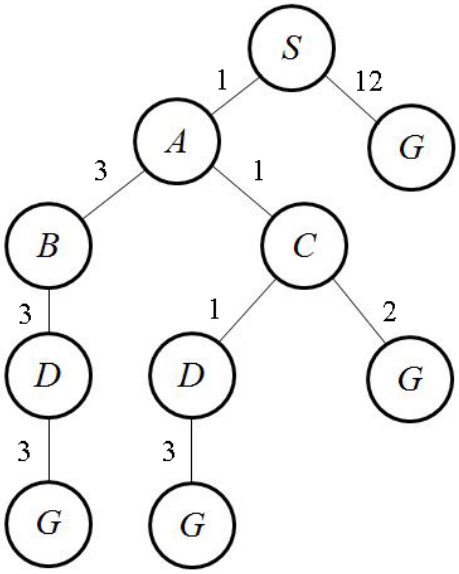


Figure 3-9 A weighted graph

3.4 Representing a graph using Adjacency Matrix

There are two different popular methods to represent the graphs [28]. The first way is representing graphs using the adjacency matrix and the second way is using a linked list. In this section, representing a graph using the adjacency matrix is presented.

Graph G = (V, E) has n vertices. For representing graph G using an adjacency matrix, define a $n \times n$ two dimensional array called T. If there is an edge between $v_i$and v_j , then $T[i][j] = 1$ and $T[j][i] = 1$. If there is not any edge between v_i and v_j , then $T[i][j] = 0$ and $T[j][i] = 0$. The following matrix shows the adjacency matrix of the graph shown in Figure 3-10 [28].

$$T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad (3.2)$$

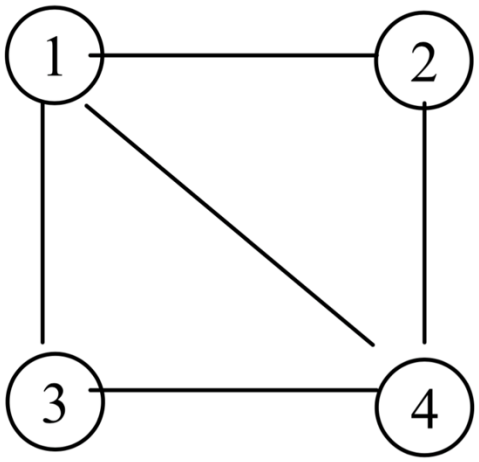


Figure 3-10 adjacency matrix for graph

If graph G = (V, E) is a directed graph and has *n* vertices, for representing graph G using the adjacency matrix define a $n \times n$ two dimensional array called T. If there is an edge between v_i and v_j which start from v_i and end in v_j then $T[i][j] = 1$. If there is not any edge between $v_i$and v_j which start from v_i and end in v_j then $T[i][j] = 0$. The following matrix shows adjacency matrix of the graph which is shown in Figure 3-10 [28].

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad (3.3)$$

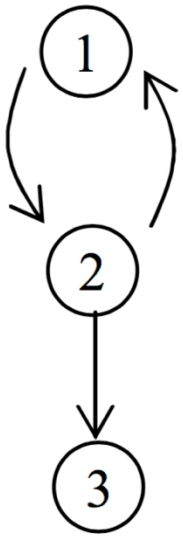


Figure 3-11adjacency matrix for directed graph

If Graph G = (V, E) is a directed weighted graph and has n vertices, for representing Graph G using the adjacency matrix define a $n \times n$ two dimensional array called T. If there is an edge between v_i and v_j which start from v_i and end in v_j then $T[i][j]$ is equal to weight of the edge. If there is not any edge between $v_i$and v_j which start from v_i and end in v_j then $T[i][j] = 0$.

4 Enumeration, Structural and Dimensional Synthesis of Robotic Hands

4.1 Introduction

Kinematic synthesis, the process of creating a mechanical system for a given motion task, can be used in order to select and size a topology as a candidate hand design. The synthesis process for robotic hands has four main steps that are detailed below: task definition, type or structural synthesis, solvability calculations, and dimensional synthesis.

After this process we obtain a set of joints, their connectivity, and their relative position along a chain. Further steps of ranking, optimization and detailed design will be necessary to implement the candidates into functioning hands.

4.1.1 Task Definition

The task is the desired motion of the elements of the hand whose interaction with the environment is of interest. For a multi-fingered hand, a simultaneous motion of all fingertips or surface contacts, which could be any limb of the hand, is to be defined. For each fingertip or contact limb, a set of positions are defined as location plus orientation. Figure 4-1 shows a trajectory task for a hand with four fingertips.

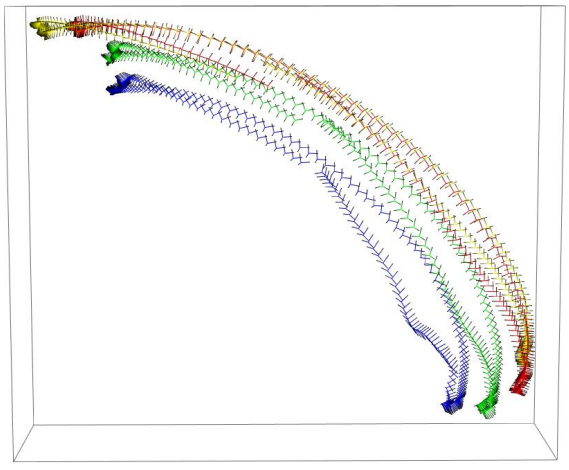


Figure 4-1 A motion task for a four-fingered hand, obtained using human hand motion capture

4.1.2 Type Synthesis

Type synthesis, or structural synthesis, is the enumeration, selection and ranking of the kinematic chain topologies to be used as candidate designs. In the case of a robotic hand, it implies the selection or calculation of the number of fingers, the number of joints at the wrist, the number of splits or branchings, and the number of joints for the serial chain making each branch, as well as the type of joints to be used. This could be an a-priori selection by the designer or it could be calculated based on the task. Having an automatic type synthesis stage helps the designer exploring the sometimes vast field of possible solutions, and identify trends in the candidate topologies.

4.1.3 Solvability

In the case of simultaneous tasks of all fingertips, solvability is defined as the ability of all combination of fingers to perform their relative tasks, and it is a condition that needs to be checked in order to be able to do the dimensional synthesis. It consists of checking what is the maximum number of positions that do not overconstrain each root-to-end-effector(s) subgraph, and ensuring that each subgraph is less constrained than the overall graph. In this calculation, the subgraphs obtained by moving the root to each end-effector need to be included to account for relative motion between fingertips.

4.1.4 Dimensional Synthesis

In the dimensional synthesis stage, the position of the joint axes are to be calculated, for the selected solvable topology and for the desired kinematic task. There are many techniques to state and solve the dimensional synthesis equations; regardless of the formulation, the output is the position of the joint axes at a reference configuration. This output is equivalent to the set of parameters defining the relative location and orientation between adjacent joints. The kinematic solution can be then used for the detailed design of the hand.

4.2 Tree Topologies

A tree topology for a kinematic chain has a set of common joints spanning several chains, possibly in several stages, and ending in multiple end-effectors [29]. A branch of the hand is defined as a serial chain connecting the root node to one of the end-effectors,

and a palm is a link that is ternary or above. The tree topology is represented as rooted a tree graph; the approach of Tsai [23] is followed, with the root vertex being fixed with respect to a reference system.

A multi-fingered hand is defined as a kinematic chain with several common joints - the wrist, which is a fundamental part of the hand manipulation- spanning several branches, possibly in several stages. At the end of each branch are the end- effectors, the fingertips. They are the main elements whose motion or contact with the environment is being defined by the task; this can be generalized to consider other intermediate vertices of the topology. Open hands, that is, hands not holding an object, are represented as kinematic chains with a tree or hybrid topology. For our synthesis formulation, the internal loops in the hand structure are removed using a reduction process [21], to obtain a tree topology with intermediate links that are ternary or above.

Tree topologies are denoted as SC − (B_1,B_2,...,B_b), where SC is a serial kinematic chain representing the initial common joints and the dash indicates a branching or splitting, with the branches adjacent to SC contained in the parenthesis, each branch B_i characterized by its type and number of joints. Figure 4-2 shows the compacted and possibly reduced graph for a 2R−(2R, R−(3R,3R,3R),2R), or 2−(2,1−(3,3,3),2) chain if we drop the R in the case of all revolute joints. This hand has three branches, one of them branching again on three additional branches, for a total of five end-effectors or fingertips. The root vertex is indicated with a double circle. While most current robotic hands have a single splitting stage spanning several fingers, this can be generalized for greater adaptation to different applications by using hands with topologies such as the one presented in Figure 4-2.

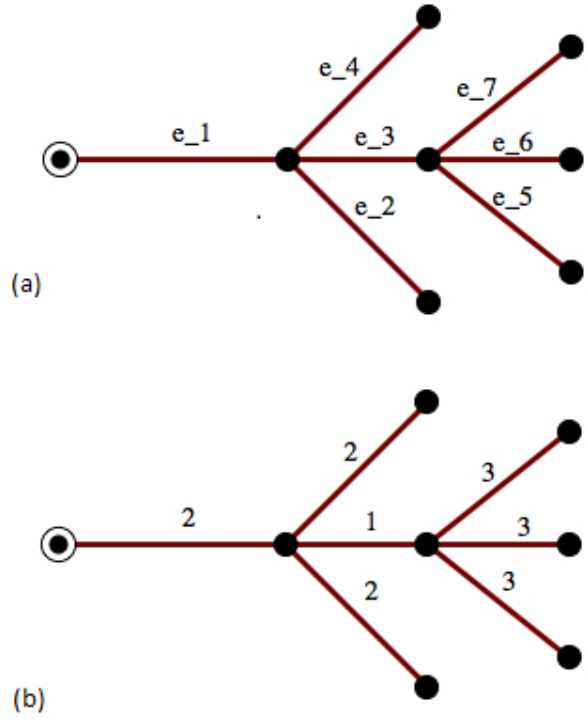


Figure 4-2 a five-fingered, two-palm hand topology. a) indicates the numbering of the edges and b) indicates the number of joints for each edge.

Figure 4-3 shows the kinematic sketch of the hand which presented in Figure 4-2.

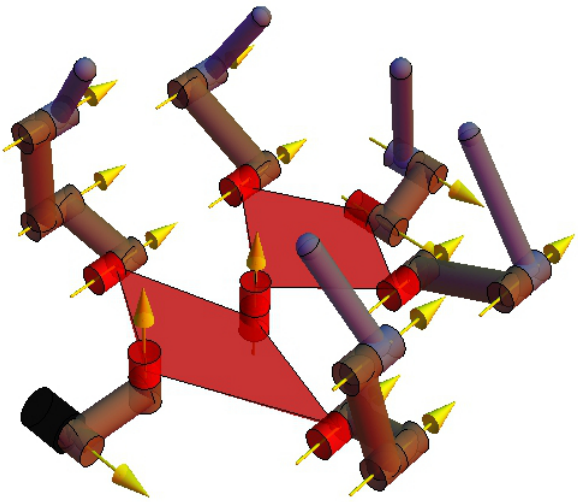


Figure 4-3 a five-fingered, two-palm hand kinematic sketch.

A tree topology is represented by two arrays, which capture incidence and adjacency properties as well as information on the edges. Assume a numbering of the graph edges to define a parent-pointer array and a joint array. The length of both arrays is equal to number of edges of the tree graph after the reduction process is applied, that is, each edge, and each entry of the arrays, will correspond to a serial chain of the robotic hand.

The parent-pointer array implements the parent-pointer representation, where each element takes the value of the previous edge, the first edge being usually the one incident at the root vertex. The edges incident at the root vertex have no parent and they take the value zero. Each element of the joint array contains the number and type of joints for each edge. If we are limited to revolute joints, then the joint array element will be equal to the number of joints for that edge. As an example, for the tree topology shown in Figure 4-2, the parent-pointer array and joint array are defined as p = {0, 1, 1, 1, 3, 3, 3} and j = {2, 2, 1, 2, 3, 3, 3} for the given numbering of the edges.

4.3 Type Synthesis and Enumeration

Given a simultaneous motion task for all fingertips, it is important to know how many, and what hand topologies are suited for the task. The number of candidate hand topologies of a certain type is usually very high and unbounded in some cases [30]. This number of suited topologies can be reduced if some additional constraints are added. At the end of the process, one or a few of these topologies will be selected for performing the additional design steps. The approach taken here is different from previous research such as [14], and it is based on free finger motion.

The conditions for considering a topology for the task are, at the least, to have the same number of end-effectors as the task and to be solvable, according to the criteria defined in [21] and [30]. The set of suited topologies can be ranked according to other criteria, such as number of edges, number of splits, and number of joints per edge, among others.

4.3.1 Candidate Topology Search

A search method and its algorithmic implementation is presented here to find solvable topologies for a defined task. The task is assumed to be a general subset in the SE(3) group of rigid motion and its derivatives, and the goal is to find all topologies that can be paired with the task for dimensional synthesis, given a set of user-defined restrictions.

User-defined inputs are the number of positions of the task *m*, the number of end-effectors, or branches, *b*, and the total number of edges of the graph *e*. Remember that every edge corresponds to a serial chain. The output is the set of topologies that meet the solvability criteria subject to these conditions.

The task-sizing formula in [21] is applied in the first place to find all possible branch topologies for the given number of end-effectors. Start by calculating following equation.

$$J = \sum_{i=1}^{e} j_i = \frac{(m-1).6.b}{m+3} \qquad (4.1)$$

where J represent the total number of joints and j_i are the joints for the serial chain corresponding to the *i-th* edge of the topology. The number of joints per edge has to be between 1 and 5 for synthesis purposes, as a serial chain of length 6 or higher does not impose any restriction on the motion.

The presented method includes three steps. First, all possible tree structures (parent-pointer arrays p) which meet the input criteria are found, for the given number of branches and number of edges. In the algorithmic implementation, the parent- pointer array is filled up starting at the root and sequentially according to the following rules:

• p (1) = 0. The first edge is the root node and has no parent.

• If *i* is not an end-effector, *p(i)* can accept any value between *p (i − 1)* to *i − 1*. The values of parent pointer array are increasing *(p(i) ≥ p(i − 1))*. This condition helps to avoid adjacent branch isomorphism.

• If *i* is an end effector, *p(i)* can accept any value between *p(i − 1)* to *e − b*, since the last *b* edges are end-effectors and cannot be parents.

Second, for each structure found in first step, construct all possible joint arrays which meet the input criteria. This implies writing all joint arrays with length equal to *e* that satisfy equation 4.1 and with entries between 1 and 5. Constructing isomorphic trees is avoided by proper index and value assignment.

Finally, after finding all possible joints arrays for each parent pointer array, check the solvability of each topology including parent pointer array and joint array. If it is solvable, add it to result as a candidate topology. This method yields all non- isomorphic trees [37] for the input parameters.

4.3.2 Type Synthesis Enumeration

Even though this search may be unbounded, reduced atlas can be created for a certain range on the number of end- effectors and precision positions.

Table 4-1 shows different values for the inputs and the number of candidate topologies that can be found. In this table, **m** is the number of task positions for each fingertip, **b** is the number of end-effectors, **e** is the number of edges of the graph. The overall number of joints is calculated under Joints and the number of different joint arrays **j** and parent-pointer arrays **p** are calculated. The candidate topologies are the solvable combinations of joint arrays and parent-pointer arrays.

Table 4-1 Type Synthesis Results for selected inputs

INPUTS			OUTPUTS			
m	b	e	Joints	j	p	Candidate Topologies
3	2	2	4	2	1	1
3	2	3	4	2	1	2
3	3	3	6	3	1	1
5	2	3	6	6	1	4
5	3	3	9	5	1	1
5	3	4	9	45	2	9

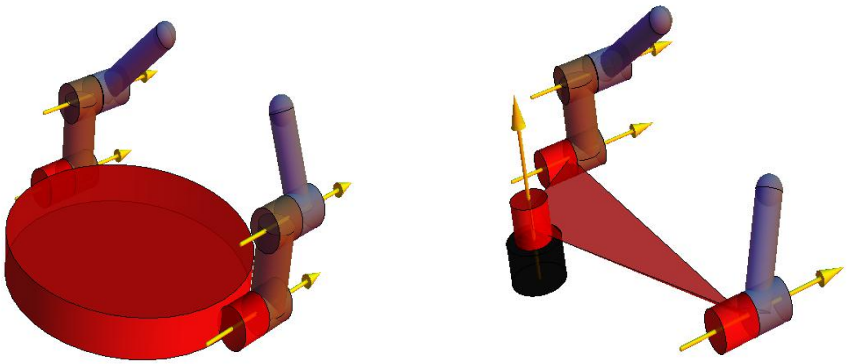
5	3	5	9	46	1	19
5	4	4	12	8	1	1
5	4	5	12	187	3	14
5	4	6	12	478	3	72
5	4	7	12	206	1	47
6	3	4	10	58	2	4
6	3	5	10	76	1	13
9	4	4	16	5	1	1
9	4	5	16	250	3	26
9	4	6	16	1442	3	237
9	4	7	16	1313	1	292
13	2	3	9	11	1	6
13	4	5	18	187	3	4
13	4	6	18	1645	3	161
13	4	7	18	2137	1	233
13	6	7	27	781	5	2
21	2	3	10	10	1	10
21	3	3	15	1	1	1
21	3	4	15	45	2	24
21	5	5	25	1	1	1
21	5	6	25	168	4	57

To illustrate the results of Table 4-1, Table 4-2 shows some of the candidate topologies that can be found using this method, where **p** denotes the parent-pointer array and **j** the joint array of the topology. Due to the high number of solvable candidate topologies, it is not possible to present them all in the table, however the final number is presented in Table 4-1 for each example.

Table 4-2 Examples of Type Synthesis

Example 1	Example 2
m=3 b=2 e=2&3 Topologies: p = (0,0) j = (2, 2) p = (0,1,1) j = (1,1,2) p = (0,1,1) j = (2,1,1)	m=5 b=4 e=6 Some selected topologies: p = (0,0,1,1,2,2) j = (1,1,2,3,2,3) p = (0,1,1,1,2,2) j = (3,1,1,3,3,1) p = (0,1,1,2,2,2) j = (3,2,1,3,2,1) p = (0,1,1,2,2,2) j = (2,2,2,2,2,2)
Example 3	Example 4
m=13 b=4 e=5 Some selected topologies: p = (0,1,1,1,1) j = (2,4,4,4,4) p = (0,1,1,1,1) j = (3,4,4,4,3) p = (0,1,1,1,1) j = (4,3,4,4,3) p = (0,1,1,1,1) j = (4,4,4,4,2)	m=21 b=5 e=6 Some selected topologies: p = (0,0,0,0,1,1) j = (2,5,5,5,4,4) p = (0,0,0,1,1,1) j = (3,5,5,2,5,5) p = (0,0,1,1,1,1) j = (4,5,3,5,3,5) p = (0,1,1,1,1,1) j = (5,5,3,4,3,5)

Figure 4-4 presents the three non-isomorphic topologies for two fingertips and three precision positions.



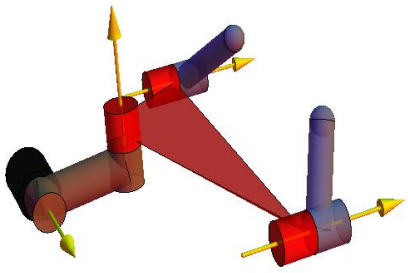


Figure 4-4 Kinematic Sketch of all non-isomorphic candidate topologies with two fingertips and solvable for three precision positions. from the left to right: 0-(2R, 2R), R-(2R, R) and 2R- (R ,R)

4.4 Solvability

A hand is defined as *solvable* when it can be designed for a meaningful simultaneous task of all the fingertips or end-effectors, that is, a positive rational task with at least two positions. Because some fingers may be overconstrained while others are underconstrained for a given topology, solvability needs to be checked systematically for all root-to-end-effector subgraphs of the hand, including those obtained when changing the root vertex to one of the end-effectors.

Equation 4.2 calculates number of positions for the exact kinematic synthesis of a tree topology. If the number of positions so obtained for the kinematic task of all subtrees is greater or equal than the number of positions for the overall tree, the tree is solvable for kinematic synthesis.

$$m = \frac{D_s^e.E - D_c^n.B}{D_{ee}^n.B - D_j^e.E} + 1 \qquad (4.2)$$

In this equation, D_s^e is the vector containing the number of structural variables for each edge, E is the vector of ones for the edges belonging to the subgraph, D_c^n is the vector of possible extra constraints for each branch, B is the vector of ones for the branches belonging to the subgraph, D_{ee}^n is the vector of degrees of freedom for the motion of each end effector, and D_j^e is the vector containing the number of joint variables for each edge. These vectors are calculated with the help of the root-to-end-effector path matrix of the graph.

The algorithmic implementation of the solvability condition has two steps, the first step is creating all possible subgraphs and their corresponding arrays, and the second step is calculating the solvability for those subgraphs.

As an example, for the tree topology shown in Figure 4-2, solvability needs to be checked for the original tree, the following root-changing trees (Figure 4-5) and all of their subtrees. Figure 4-6 shows the new parent-pointer representation assignment.

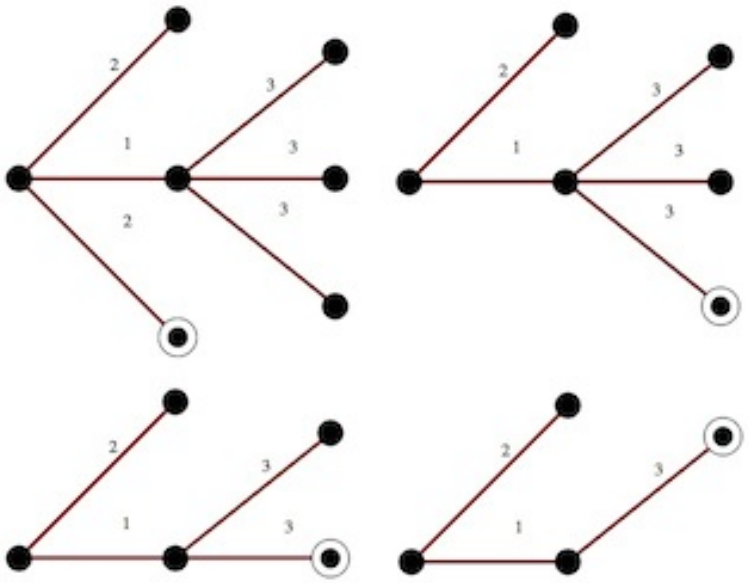


Figure 4-5 The subtrees which need to be checked for solvability

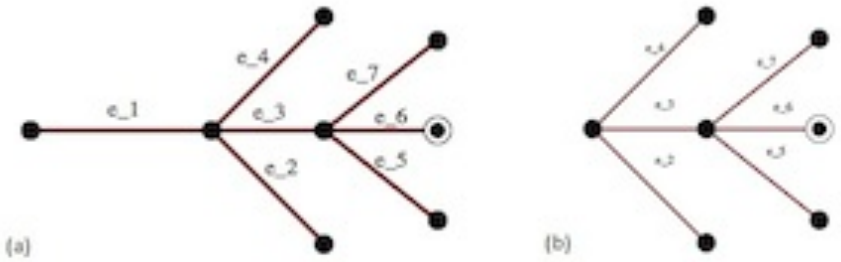


Figure 4-6 Tree a) before and b) after changing the root

The process can be itemized as follows:

a) Remove the common edge and set parent pointer as zero for those edges which are the child of the common edge. Figure 4-6 shows the tree previous root in the first tree and the tree after changing the root to next root in the second one, and further steps in the other two graphs. The parent pointer array after this step changes from ppt = {0,1,1,1,3,3,3} to ppt = {−1,0,0,0,3,3,3}. The value −1 means that the edge has been removed.

b) There is a path between the previous root and the new root. In this step, the parent-pointer value of the edges that are connected in this path is updated. In the example of Figure 4-6, the path includes edges 3 and 6. In this step, the value of parent pointer for edges 2, 4, 5, 7 is changed. The parent-pointer array after this step changes from ppt = {−1,0,0,0,3,3,3} to ppt = {−1,3,0,3,6,3,6}.

c) Finally, the parent-pointer value for the edges which are in the path is updated, by changing the value of parent pointer for edges 3 and 6. The parent-pointer array after this step changes from ppt = {−1,3,0,3,6,3,6} to ppt = {−1,3,6,3,6,0,6}.

d) After the previous step, the new tree is ready and *m* can be calculated for all the combinations of the branches, as the algorithm shows. Knowing the branch connectivity is needed for making the [B] matrix, and the tree with b branches has $2^b - 1$ combinations of branches for the original root node; when switching the root node to each end-effector, that yields $2(2^b - 1) - b$ different subtree combinations. These are defined by changing $1 \rightarrow 2^j$ to binary numbers using j digits, j = 1,...b. Finally, all the needed matrices are available for calculating m and comparing them to M.

4.4.1 Examples of Solvability

The following examples show the results of the solvability checking algorithm for some hand topologies. For the cases in which the topology is solvable, the number of positions to be used for exact kinematic synthesis is returned. If the tree is not solvable, the overconstrained subtrees are identified.

Table 4-3 Examples of Solvability

Example 1: 3R-(2R, R-(3R,3R,3R,3R)) p = (0,1,1,2,2,2) j = (3,1,2,1,1,1) Solvable m = 7	Example 2: R-(2R-(3R-(R, R),3R-(R, R)), 2R-(3R-(R, R),3R-(R, R))) p = (0,1,1,2,2,3,3,4,4,5,5, 6,6,7,7) j = (1,2,2,3,3,3,3,1,1,1,1, 1,1,1,1) Not Solvable R-(R) overconstrained
Example 3: R-(R-(2R-(R, R-(R,R)),R-(R,R)) R-(R,R)) p = (0,1,1,2,2,3,3,4,4,5,5,9,9) j = (1,1,1,2,1,1,1,1,1,1,1,1,1) Solvable m = 3	Example 4: 2R-(3R, R-(2R,2R,2R),3R p = (0,1,1,1,3,3,3) j = (2,3,1,3,2,2,2) Solvable m = 5

Figure 4-7 shows the tree topology and kinematic sketch of example 1.

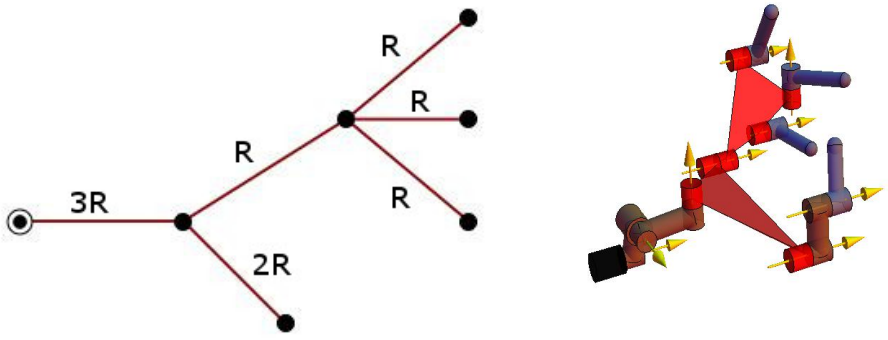


Figure 4-7 Tree topology and kinematic sketch of example 1.

Figure 4-8 shows the tree topology and kinematic sketch of example 2.

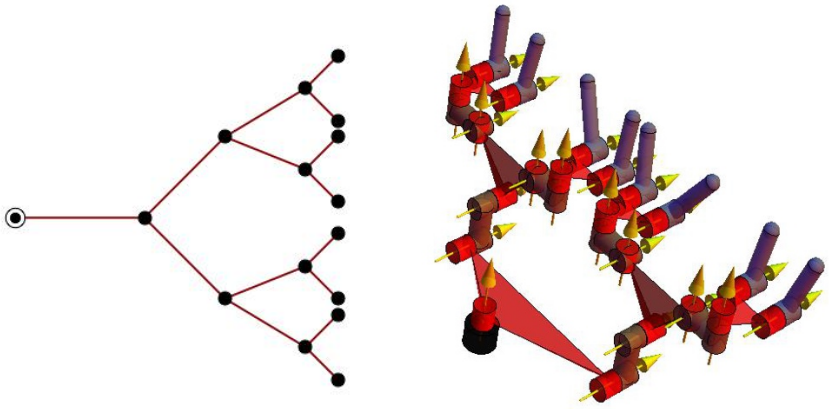


Figure 4-8 Tree topology and kinematic sketch of example 2.

Figure 4-9 shows the tree topology and kinematic sketch of example 3.

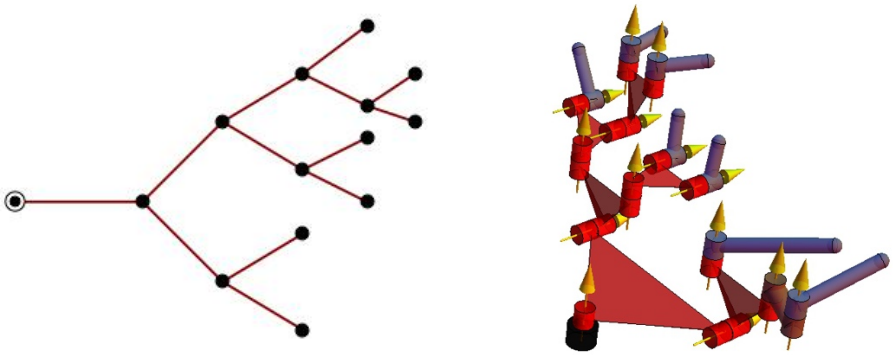


Figure 4-9 Tree topology and kinematic sketch of example 3.

Figure 4-10 shows the tree topology and kinematic sketch of example 4.

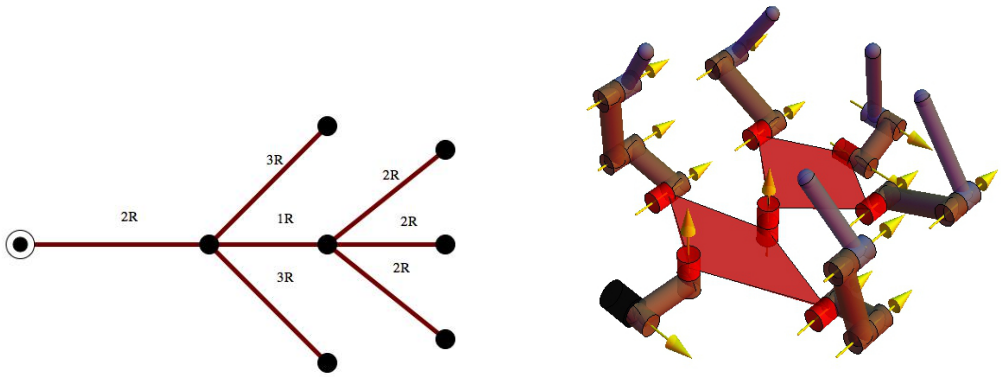


Figure 4-10 Tree topology and kinematic sketch of example 4.

4.5 Dimensional Synthesis

Kinematic dimensional synthesis is used to shape the new designs for robotic hands, able to grasp and/or manipulate in a given application. Dimensional synthesis has the candidate topology and the kinematic task as inputs. The kinematic task consists of a set of simultaneous displacements for each fingertip, as well as velocities and accelerations defined at some or all of those positions.

4.5.1 Automatic forward Kinematic Equations

For the design of robotic hands with arbitrary topologies, including multiple splitting stages, forward kinematics equations need to be automatically created from the tree topology and its associated arrays, identifying the common joints that will appear in the equations of several branches. The strategy to accomplish this is to divide the forward kinematics in serial chains -corresponding to graph edges-, branching points and end-effector points. Three types of objects are defined as outlined below:

1) Chain: a set of joint axes connected in series. There are two different types of chains, those ending on an end effectors and those ending at a branching point. This second type is common to several branches, however from the point of view of the object, they are generated equally.

2) Tip Contact Point (TCP): TCPs are created for each end-effector and then attached to the corresponding end- effector chain. In the most general case, we can attach a TCP to any link, such

as a palm link or intermediate finger link.

3) Splitter: a vertex that spans more than one edge. Splitters are identified and created, and the chains spanning from each of them are attached to the splitter. If the splitter has a predecessor, then the splitter is attached to the common serial chain.

This sequential process takes place until design equations are created for each chain from root node to end-effector node. First step is generating the end effector chains. For each end effector chain, generate and attach a TCP. For a single- branch topology (b=1), the process is done. For multi-finger topologies, Splitters are generated for each common joint and chains are attached to them, from end-effector to root. Finally, the tree is attached to a first Splitter (sp0) for topologies with no wrist, or to the serial chain of the first common edge in case of a wristed hand.

4.5.2 Exact Synthesis

Exact dimensional synthesis has been explored in [21]. The approach followed to create dimensional synthesis equations consists on equating the forward kinematics of each root-to-fingertip branch in the hand to the set of positions defined for the fingertip. Given a set of m_p task positions $\hat{P}^i_k$, k = 1 . . . m_p for each end-effector (denoted by superscript i), m_v task velocities V^i_r for each end-effector i, r = 1...m_v, and m_a task accelerations A^i_s for each end-effector i, s = 1 . . . m_a , where $m = m_p + m_v + m_a$, design equations are created. Compute the relative displacements from a selected reference position, usually position 1, and equate the relative forward kinematics to those relative positions $\hat{P}^i_{1k}$. The twist of each 1k end effector V_r^i is equated to the linear combination of twists for each joint axes, and similarly for the acceleration of the end effectors. The Plucker coordinates of the joint axes appear explicitly in the forward kinematics when these are computed as the product of exponentials for relative displacements, and linearly in the velocity and acceleration equations. For a hand with b fingertips, this yields b sets of equations 4.3 that are to be solved simultaneously,

$$
\begin{aligned}
&\hat{P}^i_{1k} = \prod_{j\in\{B_i\}} e^{\frac{\Delta\hat{\theta}^k_j}{2}\mathsf{S}_j},\\
&\mathsf{V}^i_t = \sum_{j\in\{B_i\}} \mathsf{S}^t_j\dot{\theta}^t_j,\\
&\mathsf{A}^i_r = \sum_{j\in\{B_i\}} \mathsf{S}^r_j\ddot{\theta}^r_j + \sum_{j,h\in\{B_i\}} \dot{\theta}^r_j\dot{\theta}^r_h[\mathsf{S}^r_j,\mathsf{S}^r_h],\\
&\qquad i=1,\ldots,b; \quad k=2,\ldots,m_p; \quad t\in\{T_i\}; \quad r\in\{R_i\},
\end{aligned}
\tag{4.3}
$$

where the number of end-effectors, or branches as root-to- fingertip chains, is indicated by b, m_p is the number of exact positions, and $\{B_i\}$ is the set of ordered indices of the joints belonging to branch i, which can be obtained from the graph matrices. The set of ordered indices $\{T_i\}$ and $\{R_i\}$ correspond to positions where twists and accelerations have been defined, for each branch i. Notice that some of the joints will be common to several branches. The joint axes at the reference configuration are denoted as S_j, and the joint axes at the configuration given by position k are denoted as S^k_j .

This yields a total of $6(m_p - 1 + m_v + m_a)b$ independent equations to be simultaneously solved. The method has been applied to simultaneous rigid-body motion tasks for all fingertips [19], defined by a finite set of positions, and to simultaneous fingertip tasks defined by a finite set of displacements and associated twists. For most topologies, this method yields many potential designs.

4.5.3 Multiple Velocity Synthesis or Constrained-motion synthesis

For tasks aiming to define a free trajectory for each fingertip, the definition of a finite set of positions, with a single twist vector defining the velocity for each position, and possibly a single acceleration 6D vector, gives a full characterization of the task. However, for tasks which are constrained by the contact between the fingertip and an object, the definition of the allowed subspace of velocities at each point can be used to ensure the desired behavior for some grasping actions such as finger sliding or finger rolling, for a suited hand topology. Notice that the velocities must always be defined at a given position of the end-effector.

Consider the desired angular velocity of the end-effector and linear velocity of the origin

of the end-effector frame at a given position, and calculate the fixed-frame six-dimensional twist. In this twist, the point velocity is calculated at the origin, so that it would yield the desired linear velocity at the origin of the end-effector frame.

A constrained motion given by a contact is defined, at a given position, as a subspace of wrenches, W, whose magnitudes can be as high as needed. The subspace of reciprocal twists, V , define the potential directions of allowed motion at that position.

A fingertip in contact with a surface can be kinematically modeled using one of the standard fingertip joints, see for instance [31], such as pointy fingers or soft fingers, which are defined by their degrees of freedom and friction cone if applicable. For a general case, the dimension of the subspace of reciprocal twists can be made to coincide with the mobility of the parallel mechanism formed when the hand is in contact with an object (defined by n links and j joints of f_i degrees of freedom each),

$$\dim(V) = 6(n-1) - \sum_{i=1}^{j}(6 - f_i) \qquad (4.4)$$

Using this method, a hand can be synthesized for a desired m-dimensional subspace of twists at each precision position, just by defining a set of m independent twists at that position.

As an example, consider a hand with a $2-(2, 2)$ topology, with two fingers and soft finger joints at the fingertip. Figure 4-11 shows the graph and kinematic sketch of the $2-(2, 2)$ hand. This hand has two revolute joints at the wrist and two revolute joints at each of the two fingers.

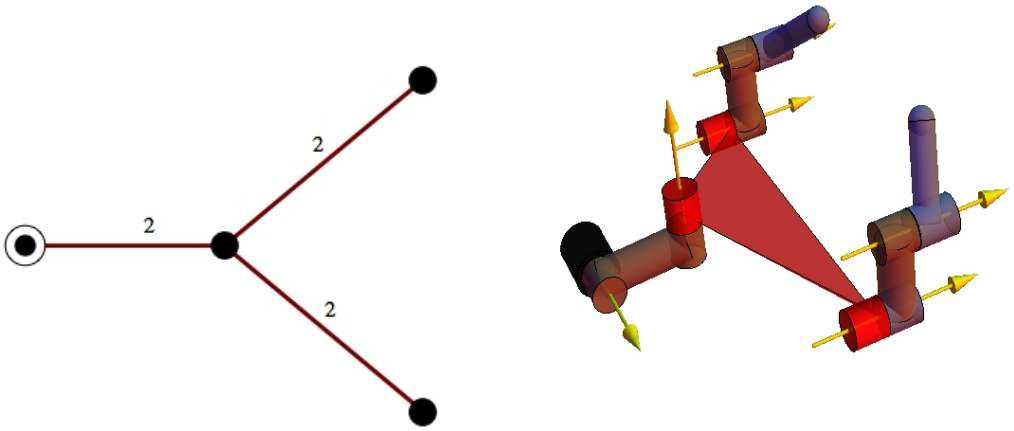


Figure 4-11Topology and Kinematic sketch for 2-(2, 2) hand

This topology is solvable for a total of m = 5 precision positions. Define a task with m_p = 3, with one position having two specified twists each (m_V = 2), and the rest of positions having no specified velocities. If this is a task in which both fingers contact an object, and assuming a general grasp, the 2 − (2, 2) hand has 4 degrees of freedom according to the general mobility formula. Two of them corresponds to the wrist rotations, while the other two are in-hand degrees of freedom. This allows us to include in the task the ability of the fingers to be compatible with a contact constraint at a given position.

5 Software

5.1 Kinematic Solver

The algorithms presented in this work have been implemented in a kinematic design software. A first version of the solver for dimensional synthesis, *ArtTreeKS* (Articulated Tree Kinematic Synthesis), was developed [19] for tree topologies with a single branching, corresponding to anthropomorphic or simple hands. A single root is sought for the system of equations using a hybrid solved, based on a Genetic Algorithm (GA) built on top of a Levenberg-Marquadt local optimizer, which minimize the average error of the dual quaternions representing the task. This numerical solver yields a single solution but allows dealing with tree topologies with a very high number of joints and fingers.

Evolutionary Algorithms have been applied in different area of Robotics such as localization [32], [33]. As all meta-heuristic algorithms, this genetic algorithm must be adjusted experimentally according to the problem being solved. Each entity in the genetic algorithm is represented as a vector of real numbers that allows simple integration with other numerical solver libraries like MINPACK [34] which is used in the Hybrid solver.

This numerical solver has been integrated in the kinematic design package. The package includes a type synthesis stage, solvability checking, the ability to synthesize new designs with arbitrary branching stages (corresponding to hands with several palms), and the ability to define a task with positions and several velocities or accelerations at a given position. This second feature is important in order to fully determine manipulation actions such as finger rolling or finger sliding, or simple dexterity without changing the grasping point.

5.2 Overall Software Architecture

The software implementation follows a three-layer architecture, which is shown in Figure 5-1, and uses the elements described below. The user interface and writing of input files is done using Lua, while the solver is programmed using C++.

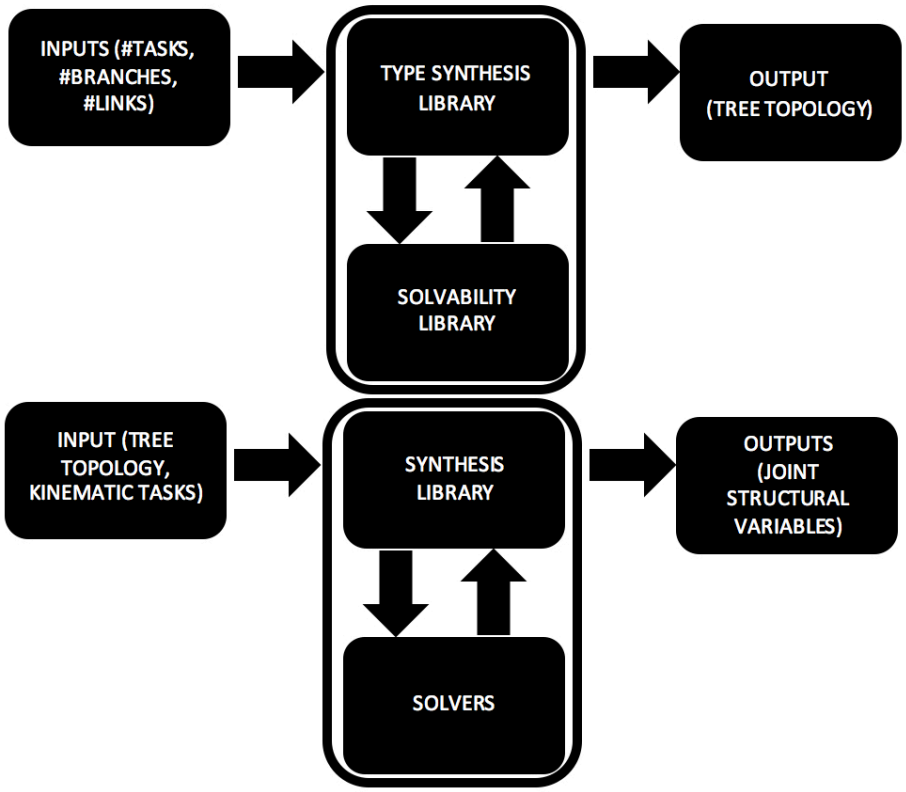


Figure 5-1 Software Architecture

- Input Files: Input information from the designer.

  · Type synthesis input: contains the number of branches, task positions and edges.

  · Dimensional synthesis input: Contains the tree topology and the values for the task positions, velocities and accelerations.

- Output files: Results of calculations that the designer can access.

  · Solvability output: output file with the results of solvability calculations for a given topology.

· Type synthesis output: output file with the atlas of solvable topologies for a given number of fingers and positions.

· Design Synthesis File: output file containing the design equations.

· Dimensional synthesis output: output file containing the result of the dimensional synthesis: Plucker coordinates of joint axes and joint variables.

- Process files: internal calculations

· Solvability Library: Lua functions to calculate tree solvability.

· Type synthesis Library: Lua file to construct all possible topologies for a set of input conditions.

· Generator File: Lua functions to check solvability, assemble forward kinematics equations and assign initial values.

· Synthesis Library: Library of functions to communicate solver and Synthesis file.

· Solvers: Genetic algorithm and Minpack C++ code to generate candidate solutions and perform minimization.

6 Design Example

As an illustration for the overall design process, let us consider a hand task that can be defined with five positions of each fingertip. For this task we can use a minimum of three fingers and a maximum of five fingers, for grasping and manipulation purposes. Three fingers may be sufficient for stable grasping but adding the extra two fingers may help in some manipulation strategies. We start the design process defining $m_p = 5$ number of task positions, three to five branches b = 3 to b = 5, and we limit the number of edges to the interval from e = 1 to e = 9 in order to have a bounded search and to limit the complexity of the design. We find, for b = 3, a total of 29 solvable non-isomorphic topologies. For b = 4 there are 134 solvable topologies, and for b = 5 we find a total of 728 solvable topologies. For b = 4 and b = 5 fingertips, we notice that the minimum number of edges for the solvable topologies is e = 4 and e = 5 respectively. In order not to complicate the design too much, we limit the search to a maximum of e = 5 number of edges. Table 6-1 shows the solutions of the type synthesis stage for 3 fingertips, and 6-2 contains the solvable topologies for 4 and 5 fingertips and up to 5 edges.

Table 6-1 Solvable Topologies for 5 positions, 3 fingertips

Fingers Edges	Topologies	Parent-pointer	Joint
b=3 e=3	1	{0, 0, 0}	{3, 3, 3}
e=4	9	{0,0,1,1} {0,1,1,1}	{1,3,2,3} {2,3,1,3} {2,3,2,2} {3,3,1,2} {1,2,3,3} {2,1,3,3} {2,2,2,3} {3,1,2,3} {3,2,2,2}
e=5	19	{0,1,1,2,2}	{1,1,2,2,3} {1,1,3,1,3} {1,1,3,2,2} {1,2,2,1,3} {1,2,2,2,2} {1,2,3,1,2} {1,3,2,1,2} {2,1,1,2,3} {2,1,2,1,3} {2,1,2,2,2} {2,1,3,1,2} {2,2,1,1,3} {2,2,1,2,2} {2,2,2,1,2} {2,3,1,1,2} {3,1,1,1,3} {3,1,1,2,2} {3,1,2,1,2} {3,2,1,1,2}

Table 6-2 Solvable Topologies for 5 positions, 4 and 5 fingertips

Fingers Edges	Topologies	Parent-pointer	Joint
b=4 e=4	1	{0,0,0,0}	{3,3,3,3}

e=5	14	{0,0,0,1,1} {0,0,1,1,1}	{1,3,3,2,3} {2,3,3,1,3} {2,3,3,2,2} {3,3,3,1,2} {1,3,2,3,3} {2,3,1,3,3} {2,3,2,2,3} {3,3,1,2,3} {3,3,2,2,2} {1,2,3,3,3} {2,1,3,3,3} {2,2,2,3,3} {3,1,2,3,3} {3,2,2,2,3}
b=5 e=5	1	{0,0,0,0,0}	{3,3,3,3,3}

The simplest solvable topology able to perform this task has parent-pointer array p = {0,0,0} and joint array j = {3, 3, 3}, that is, a 0 − (3, 3, 3) topology with three R-R-R fingers and no wrist. Some of the most complex topologies are the 3 − (2,1 − (1,2)) topology, with p = {0,1,1,2,2} and j = {3,2,1,1,2}, or the 0 − (3,2 − (2,2,3)), with p = {0,0,1,1,1} and j = {2,3,2,2,3}. Out of the 45 candidate topologies, we select for the design the topology with p = {0,1,1,2,2} and j = {2,1,2,2,2}, corresponding to the 2 − (1 − (2, 2), 2) hand, with two R joints at the wrist spanning two fingers, the first one spanning two more fingers for a total of three end-effectors. Figure 6-1 shows the topology and kinematic sketch.

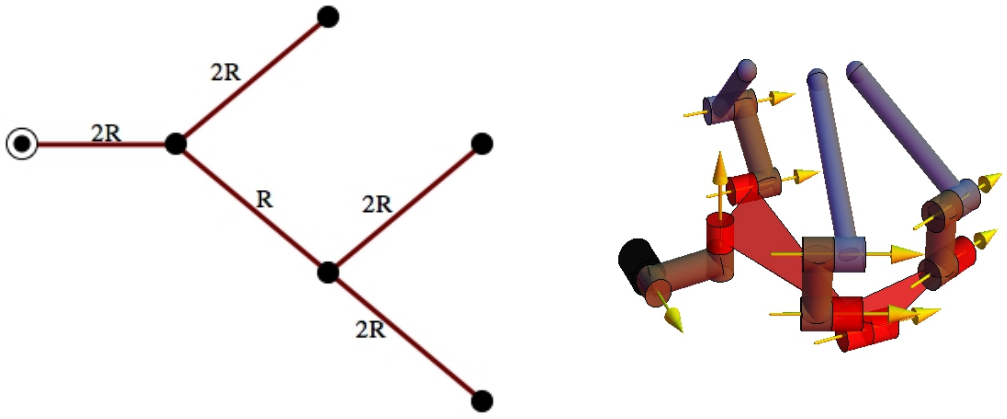


Figure 6-1 Graph and Kinematic Sketch of selected topology

Dimensional synthesis is used to shape this topology with five random finite displacements. The resulting set of equations from equation (4.3) consists of 96 highly nonlinear equations in 90 unknowns, 72 of which are independent. The results of five runs with different initial conditions are presented in Table 6-3. All five obtained solutions were feasible and different, which makes

us infer that there will be a big number of solutions for this topology.

Table 6-3 Dimensional Synthesis Solver Result

Run	Final error	Iterations	Time to solution
1	$1.15 * 10^{-13}$	1	12 sec.
2	$6.0 * 10^{-12}$	2	10 sec.
3	$1.4 * 10^{-13}$	1	5 sec.
4	$2.0 * 10^{-13}$	9	29 sec.
5	$1.0 * 10^{-13}$	1	7 sec.

The solutions obtained with the dimensional synthesis solver have been modeled using the automatic drawing procedure developed in [35] and are presented in Figure 6-3 and Figure 6-4. The positions used for this design are presented in Figure 6-2.

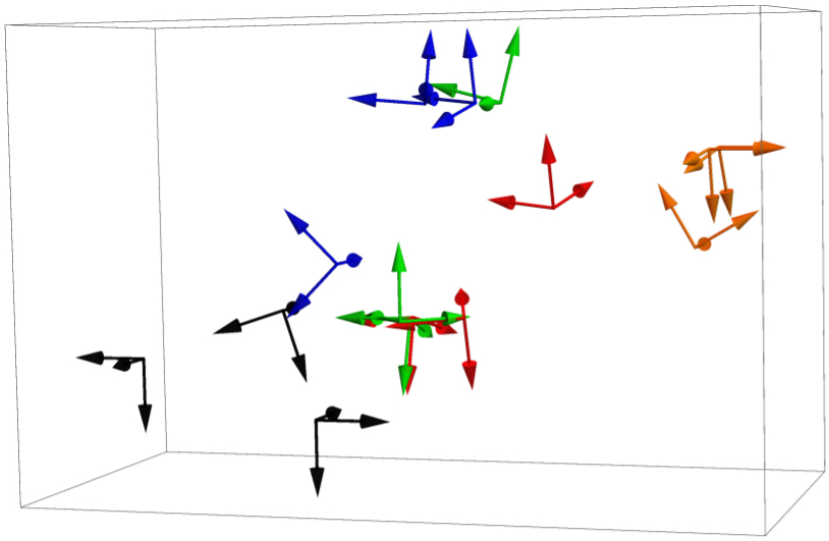


Figure 6-2 The position task. colors correspond to each position of all three fingertips.

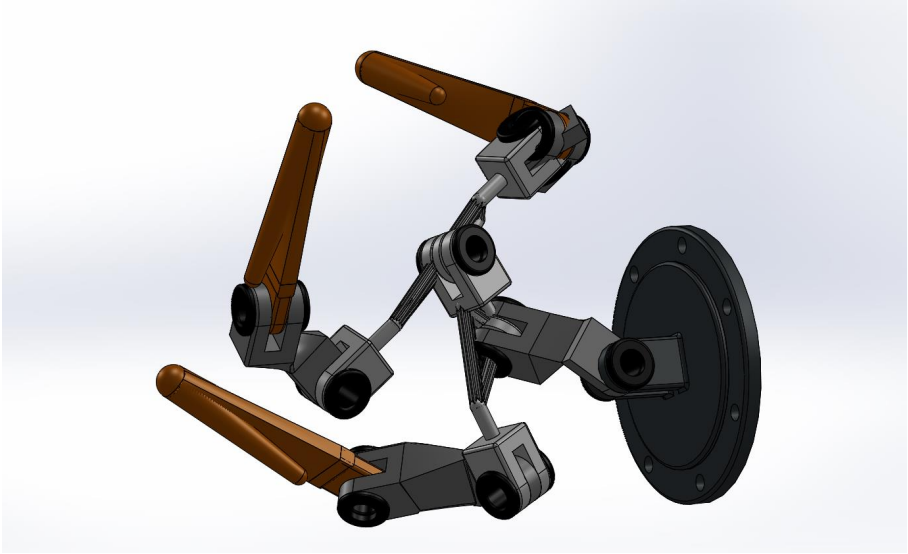


Figure 6-3 Hand design using the topology 2-(1-(2,2),2)

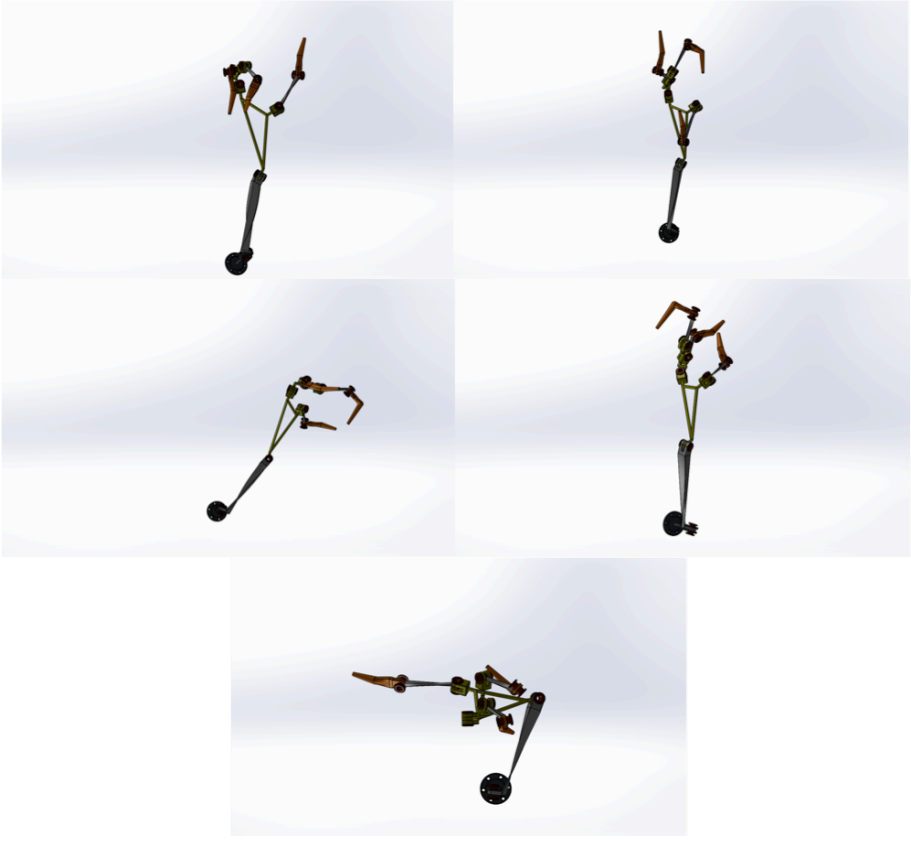


Figure 6-4 2-(1-(2,2),2) hand design for the specific position

The output of the kinematic synthesis stage is to be used as the input for a detailed design, using computer-aided tools.

7 Conclusion

This works presents the development and implementation of the different stages of kinematic design within a tool for the creation of innovative multi-fingered robotic hands. The resulting design package is able to perform type and dimensional kinematic synthesis for arbitrary tree topologies, enumerating candidate topologies and creating wristed hands with arbitrary number and type of fingers and arbitrary number and type of branchings. The solver accepts several inputs, basically a kinematic task and some limits on the desired topologies such as number of fingers or some bounds on the number of edges. The kinematic task may include finite displacements of each fingertip, and multiple velocities and accelerations for the fingertips at some of those finite positions.

Type synthesis and solvability are implemented using an enumeration technique which constructs non-isomorphic trees. The implementation of the dimensional synthesis combines the automatic construction of the tree forward kinematics with a solver consisting of a genetic algorithm and a Levenberg- Marquardt stage in order to explore the space of solutions, and shows fast convergence to a solution for each run. The current version of the solver is freely available at the project webpage. Future work will focus on generalizing some other features of the solver and on the automatic connection to subsequent stages in the design process.

The output of the design process is a *kinematic design*: a set of joint axes, defined by their Plucker coordinates at a reference configuration, and a set of joint variables and joint rates. Each kinematic design can be implemented in a final design in an unlimited number of ways, selected by the designer and constrained by additional specifications. The rationale is that a hand design tailored to an application may simplify many other aspects of the process, increasing the success of the grasping and manipulation actions.

A paper [36] has been extracted from the researches in this thesis and has been submitted in IEEE Transaction on Robotics.

8 References

[1] M.T. Mason, S.S. Srinivasa, A.S. Vazquez, and A. Ro- driguez. Generality and simple hands. *International Journal of Robotics Research*, 2010.

[2] J. Martin and M. Grossard. Design of a fully modular and back drivable dexterous hand. *The International Journal of Robotics Research*, 33(5):783–798, 2014.

[3] E. Mattar. A survey of bio-inspired robotics hands im- plementation: New directions in dexterous manipulation. *Robotics and Autonomous Systems*, 61(5), 2013.

[4] L.U. Odhner and et al. A compliant, underactuated hand for robust manipulation. *The International Journal of Robotics Research*, 33(5):736–752, 2014.

[5] M. Quigley, C. Salisbury, A.Y. Ng, and J.K. Salisbury. Mechatronic design of an integrated robotic hand. *The International Journal of Robotics Research*, 33(5):706– 720, 2014.

[6] M. Controzzi, C. Cipriani, and M.C. Carrozza. The Human Hand as an Inspiration for Robot Hand. In R. Balasubramian and V.J. Santos, editors, *Design of Artificial Hands: A Review*, volume 95 of *Springer Tracts in Advanced Robotics*, pages 219–244. Springer Switzer- land, 2014. ISBN 978-3-319-03016-6.

[7] J.E. Parada Puig, N.E. Nava Rodriguez, and M. Cecca- relli. A methodology for the design of robotic hands with multiple fingers. *Advanced Robotic Systems*, 5(2): 177–184, 2008.

[8] D.B. Sterusand and C.J. Turner. A design methodology based process for robotic gripper design. In *Proc. of the 2011 ASME International Design Engineering Technical Conferences, IDETC-CIE*, Washington D.C., USA, August 28-31, 2011.

[9] A. Ciocarlie and P. Allen. Data-driven optimization for underactuated robotic hands. In *Proc. of the 2010 International Conference on Robotics and Automation*, pages 1292–1299, Anchorage, Alaska, USA,, 2010.

[10] N. Hasanzadeh, X. He, and A. Perez-Gracia. A design implementation process for robotic hand synthesis. In *ASME Int. Design Engineering Technical Conferences*, 2015.

[11] X. Kong and C. Gosselin. Type synthesis of 3t1r 4- dof parallel manipulators based on screw theory. *IEEE Transactions on Robotics and Automation*, 20(2):181– 190, 2004.

[12] Q. Zeng and Y. Fang. Structural synthesis and analysis of serial–parallel hybrid mechanisms with spatial multi- loop kinematic chains. *Mechanism and Machine Theory*,

49:198–215, March 2012.

[13] MA. Pucheta and A. Cardona. Topological and dimen- sional synthesis of planar linkages for multiple kine- matic tasks. *Multibody System Dynamics*, 29(2):189–211, February 2013.

[14] J.J. Lee and L.W. Tsai. Structural synthesis of multi- fingered hands. *Journal of Mechanical Design*, 124:272– 276, 2002.

[15] J.K. Salisbury and B. Roth. Kinematic and force analysis of articulated mechanical hands. *Journal of Mechanisms, Transmissions and Automation in Design*, 105(1):35–41, 1983.

[16] E. Ozgur, G. Gogu, and Y. Mezouar. Structural synthesis of dexterous hands. In *Intelligent Robots and Systems (IROS) Conference*, 2014.

[17] N. Robson, J. Allington, and G.S. Soh. Development of Underactuated Mechanical Fingers Based on Anthropo- metric Data and Anthropomorphic Tasks. Buffalo, USA, 2014. ASME.

[18] M. Ceccarelli and M. Zottola. Design and simulation of an underactuated finger mecanism for larm hand. *Robotica*, 2015.

[19] E. Simo-Serra, F. Moreno-Noguer, and A. Perez-Gracia. Design of non-anthropomorphic robotic hands for anthro- pomorphic tasks. August 28-31, 2011.

[20] E. Simo-Serra, A. Perez-Gracia, H. Moon, and N. Rob- son. Design of multifingered robotic hands for finite and infinitesimal tasks using kinematic synthesis. In *Advances in Robot Kinematics*, Innsbruck, Austria, June 2012.

[21] E. Simo-Serra and A. Perez-Gracia. Kinematic synthesis using tree topologies. *Mechanism and Machine Theory*, 72 C:94–113, 2014.

[22] Dr. Alba Perez Garcia, *Kinematics of Robotics.*

[23] Lung W. Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function*. CRC Press, Boca Raton, 2001.

[24] Norman L. Biggs. *Discrete Mathematics*. Clarendon Press, Oxford, 1985.

[25] Alfred V. Aho, John E. Hpocroft, Jeffrey D. Ullman. *Data Structures and Algorithm*. Addison-Wesley, 1983.

[26] Reinhard Diestel. *Graph Theory*. Springer-Verlag, NewYork, 2000.

[27] K. D Joshi. *Foundations of Discrete Mathematics*. Wiley-Interscience,1989.

[28] Nasser Ayat, Jafar Tanha, Jeffrey D. Ullman. *Data Structures and Algorithm (in persian)*. Payamnur University, 2008.

[29] J. M. Selig. *Geometric Fundamentals of Robotics (Mono- graphs in Computer Science)*. SpringerVerlag, 2004. ISBN 0387208747.

[30] A. Makhal and A. Perez-Gracia. Solvable multi-fingered hands for exact kinematic synthesis. In *Advances in Robot Kinematics*, Ljubljiana, Slovenia, June 2014.

[31] M.T. Mason. *Mechanics of Robot Manipulation*. The MIT Press, 2001.

[32] A.Tamimi, H.Sadjadian, H.Omranpour. Mobile Robot Global Localization using Imperialist Competitive Algorithm. *3rd International Conference on Advanced Computer Theory and Engineering,* 2010.

[33] A.Tamimi, M.H.Jalilolghadr. Mobile Robot Position Tracking using Imperialist Competitive Algorithm. *3rd International Conference on Computer Engineering and Applications*, 2011.

[34] J.J. More, B.S. Garbow, and K.E. Hillstrom. User guide for minpack-1. *Argonne National Laboratory Report*, ANL-80-74, 1980.

[35] J.M. Herve ´. Analyse structurelle des me ´canismes par groupe des de ´placements. *Mechanism and Machine Theory*, 13(4):437 – 450, 1978. ISSN 0094- 114X. doi: DOI:10.1016/0094-114X(78)90017-4. URL: http://www.sciencedirect.com/science/article/pii/ 0094114X78900174.

[36] Ali Tamimi, Alba Perez-Gracia. Enumeration, structural and dimensional synthesis of robotic hands: theory and implementation. *Submitted to IEEE Transactions on Robotics*.

[37] A.D. Jaggard and J.J. Marincel. Generating tree isomor- phisms for pattern-avoiding involutions. *Ann. Comb*., 15: 437–448, 2011.