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 ____

DESIGN OF A MULTI-FINGERED ROBOTIC GRIPPER FOR AGRICULTURAL TASKS

By

Qutaiba Khalid

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

Master of Science in Measurement and Control Engineering

College of Engineering, Idaho State University

December, 2014

COMMITTEE APPROVAL

To the Graduate Faculty:

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

____ **Major Advisor**

Dr. Alba Perez Garcia

____ **Member**

Dr. Brian Williams

____ **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 co-advisors Dr. Brian Williams and Dr. Steve Chiu for their help in all matters of my life at ISU.

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. My thanks are due to Neda Hassanzadeh, Abhijit Makhal, Yimisker Yihun and last but not the least Ali Tamimi. The faculty of College of Engineering Ellen Combs for her support throughout my masters.

Let me also include my sister, my brothers and my friends whose sheer presence in my life has been a source of joy and strength. I must mention two people who have shaped me to be the individual that I am; my parents. Last but not the least my paternal Uncle, without whose support and love, I would not have come this far.

Contents

List of Figures viii

Abstract x

1 Introduction 1

1.1 Thesis Goals 1

1.2 Literature Review 2

1.3 Organization of the Thesis 4

2 Kinematics Background 5

2.1 Mobility 5

2.2 Translation 6

2.3 Rotation 6

2.4 Spatial Displacement 8

2.5 Screw Displacement 8

2.6 Dual Quaternions 10

2.7 Kinematic Synthesis 11

2.8 Design Equations for a Serial Chain 12

3 Motion Capture System 14

3.1 Introduction 14

3.2 Vicon Motion Capture System Working Principle 14

3.2.1 Installation of the Program 15

3.2.2 Firewall and Antivirus Settings 16

3.2.3 Camera Setup 17

3.2.4 Camera and software settings 17

3.2.5 Aiming, Mapping, Calibration and Volume Origin 18

3.2.6 Database 19

3.3 System Preparation 20

3.3.1 Custom Model 20

3.3.2 Data Capturing and labelling 21

3.4 Data Export 22

3.5 Potato Sampling 23

3.6 Data Processing 24
3.7 Summary 26
4 Hand topology 27
4.1 Introduction 27
4.2 Tree Topology 27
4.3 Reduced tree Topology 28
4.4 Matrix Associated to a Graph 29
4.5 Solvability of Tree Topologies for Exact Synthesis 29
4.6 Topology 3- (4,5,5,5) 31
4.7 Topology 3, (3,4,4,4) 33
4.8 Topology 3, (4,4,5,5) 35
4.9 Topology 2, (3,3,4,4) 37
4.10 Hand Selection 39
4.11 Conclusion 39
5 Kinematic Synthesis 40
5.1 Hand Task 40
5.2 Kinematic Synthesis 40
5.3 Forward Kinematics 41
5.4 Synthesis Design Equations 41
5.5 Kinematic Solver 42
5.6 Dimension of Equation Set 43
5.7 Input to the Solver 43
5.8 Output solution 44
5.9 Validation of the Solution 46
6 CAD Model of Hand 47
6.1 Solver Output 47
6.2 Solver Output Axis 47
6.3 Initial Model 48
6.4 Final Model 49
7 Conclusion and Future Works 51
8 References 52

9 Appendix 57
9.1 Solution of the Solver 57
9.2 Input of the Solver ArttreeKS 72
9.3 Mathematica Program for processing of Vicon Output 76
9.4 Mathematica Validation File of the Solver Solution 84

List of Figures

Figure 2-1(a) shows pure translation and (b) shows pure translation.[37] 7

Figure 2-2 Representation of Screw axis[37] 9

Figure 2-3A serial chain with its joints [38] 10

Figure 3-1Installation of Nexus.. 15

Figure 3-2Installation of the Nexus 16

Figure 3-3Firewall Settings.. 16

Figure 3-4Camera setup for the potato grasping task 18

Figure 3-5Custom human hand model view 1 21

Figure 3-6Custom human hand model view 2 21

Figure 3-7Human hand with pointers for grasping and trajectory information 22

Figure 3-8(Above) Segregation of potato by different size and shape. 23

Figure 3-9(Below) CAD import of the potatoes in the Solid Work using Faro Scanner from Idaho Museum of Natural History. 23

Figure 3-10(Above) CAD import of the potato in the Solid Work using Faro Scanner from Idaho Museum of Natural History 24

Figure 3-11Retro-reflection markers for tracking the trajectory of the task. 24

Figure 3-12Tranformation of absolute frame to elbow frame. 25

Figure 3-13Tragectory Views of the Potato grasping task 26

Figure 4-1The graph on the left shows the extended tree topology and the same topology is compacted in the right graph. The root vertex is represented by the double circle. 28

Figure 4-2(Left) Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph. 31

Figure 4-3 Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph... 33

Figure 4-4 Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph... 35

Figure 4-5(Left) Kinematic sketch of 3-(4,5,5,5) topology ;(Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph. 37

Figure 5-1 Shows the output solution number 1 which took 11 iterations and (Right) shows the second solution which took 28 iterations to give this result. 44

Figure 6-1Joint Axis in Solid Works 48

Figure 6-2 Palm will be attached to four end effectors and a wrist... 48

Figure 6-3 Left and Middle represent links joining two axes. The right figure represent the end effector. 49

Figure 6-4Shows the CAD model. 49

Figure 6-5Final design top and side view 50

Figure 6-6Final Design of 3-(3,4,4,4) Topology... 50

Abstract

The purpose of this work is to apply Kinematic Synthesis using tree topologies for articulated system with multiple end-effectors to the design of a robotic hand for a specific grasping task. Currently the robotic hands are designed keeping in view the anthropomorphic hands, which have complex mechanical and sensing system in a small space. The present work is part of an effort to develop grippers which are designed for a particular tasks. These robotic hands do not specifically look like a human hand and can have any number of joints, fingers and palms to achieve the specific task. The targeted task is potato sorting, which is currently being performed manually.

In order to design a robotic hand, a potato grasping task is selected .Vicon motion capture system is used for recording the human hand trajectories. Different hand topologies are selected and analyzed for solvability. The selected topologies and the input task positions are used in the numerical solver ArtTreeKS to dimension the links of the hand design. The solutions are analyzed and selected solution is then redefined using SolidWorks to obtain a final design.

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.

Chapter 1

1 Introduction

1.1 Thesis Goals

The goal of this thesis is to apply and validate Kinematic synthesis theory to an articulated system with multiple end effectors to meet certain motion specifications. The advantage of this methodology lies in the possibility of defining simultaneous manipulation and grasping action for the whole hand. The work will help in the development of the methodology to solve dimensional synthesis of articulated system with a tree structure. For this purpose the kinematic design of a multi-fingered hand is solved for a specific application. Tree articulation will help in the designing of a task specific robotic multi-fingered hand.

In order to achieve this goal the process is divided into two stages. The first stage is to create an articulated system to follow a specific motion with the use of kinematic dimensional synthesis. The design of the robotic hand is performed from an anthropomorphic point of view. A human hand task is defined and graph theory is used to select different topologies to perform the specific task. Forward kinematic equations of relative displacement are used to solve each serial chain using the dual quaternions. A solver is used to find the axes and joint angles of the kinematic chain.

The input data (joint axes and their connectivity) are used in the second stage to create a final, detail design. The data are validated to check the accuracy of the solver. The data is used for the CAD implementation. This helps to visualize and simulate the solution obtained. Optimization techniques are applied [1] on the CAD model and validated with the given task.

1.2 Literature Review

In industry the robots replace humans in audacious and repetitive tasks. The robots can be used in hostile, dangerous and dirty environments and are able to manipulate at high speed. The reliability of the system and the low gripper cost play an important role in the pick and place processes. The research field related to robotic hands differ depending on the goals they perform [57]. First approach is to design a hand that imitates the behaviors and appearance of the human hand, such as DLR hand [52], Shadow Hand [53] and Rebonaut hand [54]. The anthropomorphic design has a complex mechanical and sensing system in a small space [3], in order to match the 26 degree of freedom of human hand. The other approach is to reduce the complexity by either simplifying the mechanical structure or to develop an underactuated hand, for example Barret hand [51], Karlsruhe Hand [55] and the Highspeed hand [56]. However the reduction of the mechanical structure and underactuation limit the capabilities of the resultant hand. A thorough review of the under actuated hand can be found in [5]. In this work a hand is designed to do motion specific tasks, which reduces the complexity faced in developing the replica of the human hand and could provide the same results achieved by a human hand in performing a specific task.

Research exists on design of individual motion of the finger or parts of the hand. A finger mechanism designed by Van Varseveld and Bone [6] demonstrates a non-anthropomorphic dexterous hand. A robotic wrist's kinematic design is presented by Schafer and Dollman [7]. Non-anthropomorphic hand is presented by Walker et al [8] by using planar linkages. In this work a complete non-anthropomorphic multiple finger robotic hand is designed. One of the hypotheses of this research is that, in order to perform human tasks, it is not necessary that the robotic fingers should look like human fingers.

The design methodology is based on the use of dimensional kinematic synthesis. Planar linkages were developed for finite position dimensional synthesis problems; see [9, 10, 11]. [12, 13] solved simple systems by using geometric constraints and vector loop equations. Complex systems are been solved recently through robot kinematic equations [14, 15], but the application is limited to serial chain and has limitation in process solution [16]. Early research in the graph theory has been done by Huang and Soni [17], Woo [18] and Freudenstien and Maki [19]. Most recently Mruthyunjaya[20] and Tsai[21] have worked in this field. Work has been done in design of finger mechanism by using the structural synthesis by Chuang and Lee [22]. Articulated systems have been solved as rooted trees [23]. Dynamic systems can be described by using the tree systems by Garcia De Jalon [25]. Mechanical systems have been analyzed and controlled by articulated tree systems and their graph representations in [26]. Tischer et al [27, 28] apply graph theory to the robotic hand. There has been no application or methodology in which tree structure is used for dimensional synthesis of articulated system, which is applied in this work.

Infrared camera based systems has been used with retro reflection objects to capture the moment in many applications in the development of the exoskeletons for human body [30]. Hand motion has been recorded using the Vicon motion capture system with the retro reflection markers at each joint. This creates a challenge in capturing the data due to the close placement of the markers [31]. Similar technique is used in this thesis, but instead of reducing the number of markers on the hand [31], the arrangements and the data capture settings of the Vicon system is changed. A similar system has been used to capture the hand grasping for different tasks and storing it into a data base. Depending on the object, type of grasp is selected [enveloping grasp]. Most recently the Vicon passive marker approach is used in the EMG based tele-operation [32] to give the orientation and the position of the hand. The Vicon motion capture system is widely being used in human robot interaction, anthropomorphic data recording and similar applications.

There has been research to bring the robot cost lower, see [33] for details. In order to sort fresh fruit and horticulture there have been work going on in the image processing [34] and the handling side of the fresh fruit, vegetables and specifically potato [35]. There is minimal literature and work on the potato sorting. Machines have been developed but are being used only for finished product [36]. The research work presented in this paper will provide ground work to be used in other sorting processes with the vision control system mentioned in [34].

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 presents the details of the motion capture system and data gathering. Chapter 4 focuses on the hand topology selection and solvability. Chapter 5 presents the kinematic synthesis and the solution of the solver. Chapter 6 presents the CAD simulation of the output hand. Chapter 7 shows the conclusion and future work.

Chapter 2

2 Kinematics Background

In this chapter, the kinematic theory needed for the design process is reviewed in the context of kinematic synthesis [58]. “Kinematics is defined as the study of the motion, regardless of the force causing it and caused by it. Motion is a concept which includes position and its derivatives, mainly velocity and acceleration” [37].

2.1 Mobility

Mobility is defined as the number of independent parameters required to specify the position which includes the location and orientation of a rigid body, at a given instance of time. A rigid body is defined as a set of particles such that the distance between them remains the same [37]. A robotic system is composed of several rigid bodies whose motion is constrained by links. There are mainly two types of robots named as serial and parallel. As this research is mainly in the serial robot, so the serial robots will be briefly discussed. The mobility or the degree of the freedom is given by the following equation.

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

Where,

n defines the number of links in the serial chain(including the base).

j is the number of joints

f_i is the degree of freedom of each of the joint

2.2 Translation

In order to define the move a rigid body from point A to B, translation and rotation are two important parameters which should be known. Translation involves moving of a body from one point to another. In case of rigid bodies, all points are moved by the same amount in the same direction. The initial location of the body in the space is called fixed frame and the arbitrary position to which it has moved is called Moving frame. The representation of the translation is given below.

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

where,

X is a vector which represents the coordinate of the point x after the translation

x is a vector which represents the coordinates at the initial position

t is the vector between the initial and final location of the point x.

2.3 Rotation

During the rotation a subspace points can remain fixed while the rest of the point move by different amounts in different directions according to their location with respect to their rotation axis[37]. In order to represent the rotation an orthogonal matrix representation is used.

The form is represented below

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

where,

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

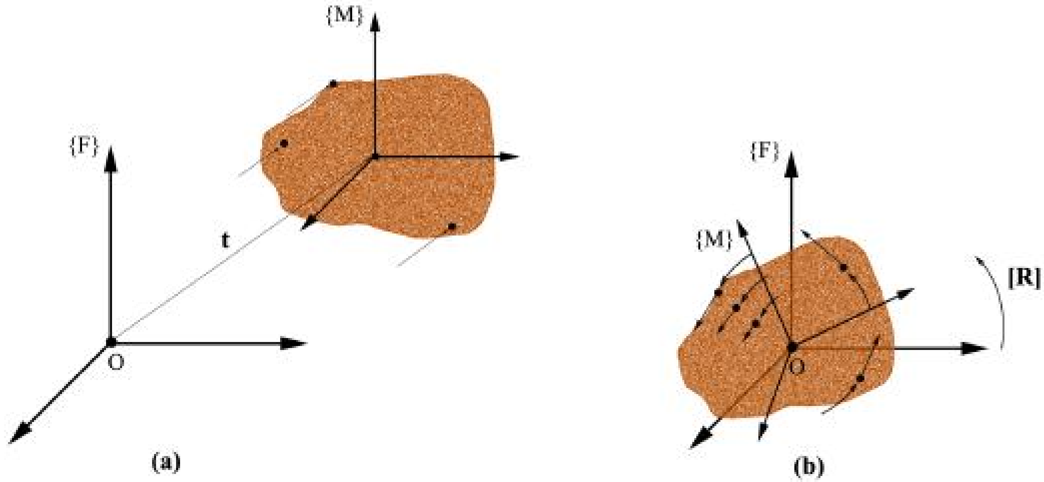


Figure 2-1(a) shows pure translation and (b) shows pure translation.[37]

The rotation about X, Y and Z are represented below

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

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

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

2.4 Spatial Displacement

Spatial displacement is the combination of a rotation and a translation about an axis. An example of the spatial displacement is given below, which consists of rotation of θ about X-axis and translation by a vector d. The expression is given below

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

The above expression can be written as

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

The expression can be expanded as

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

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

2.5 Screw Displacement

According to the Chasles' theorem [38], any displacement in 3 dimensions can be written as a screw displacement. The screw displacement is given by rotation and translation about an axes. Six parameters are required to define the screw axis representation: four are required to define the direction and location of the line, and two are defined for rotation and slide values.

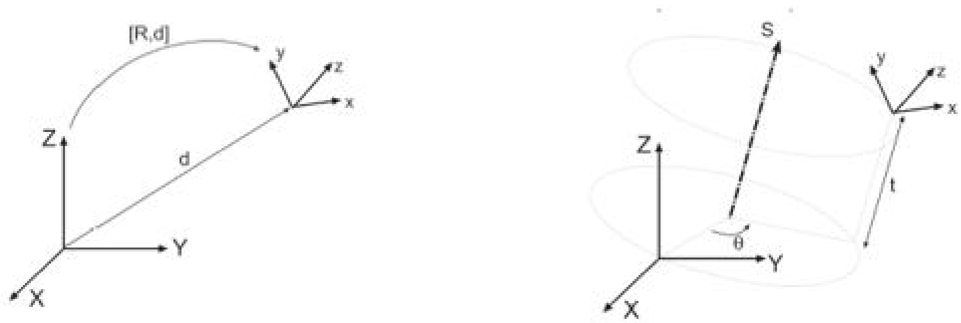


Figure 2-2 Representation of Screw axis[37]

Screw displacement for the X, Y and Z is represented below

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

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

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

The homogenous matrix is used to do all standard transformations (rotation, translation etc). The local transformation along the chain using the DH parameters is shown in the figure 2.3A. DH parameter is another method of representing the transformation from one axis to another.

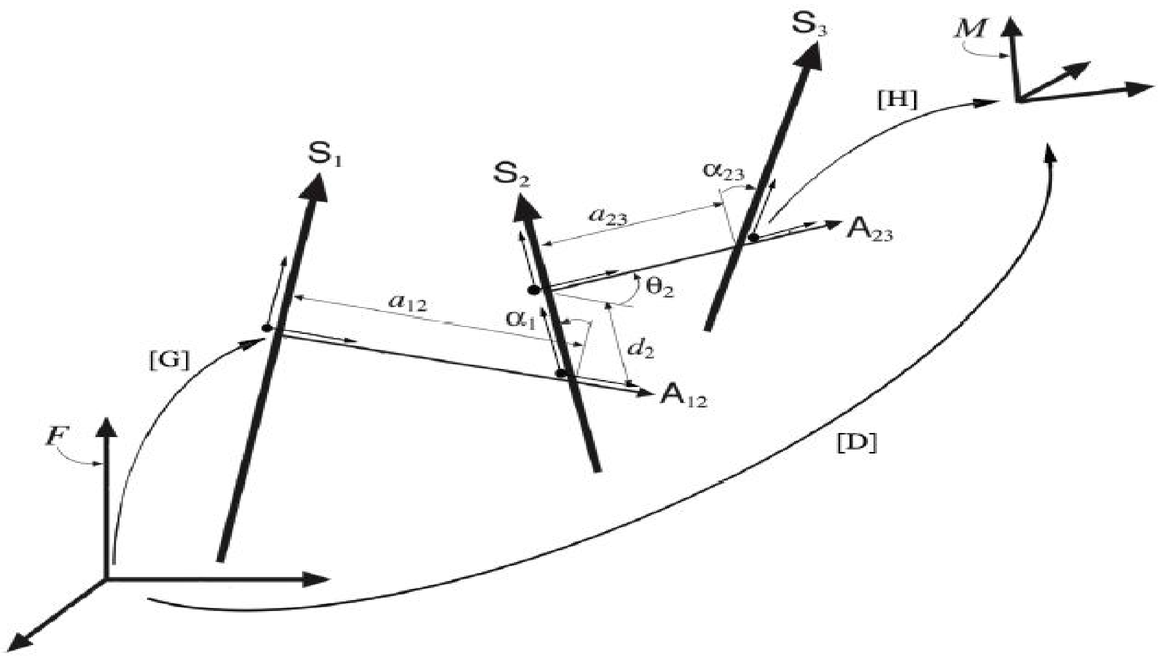


Figure 2-3A serial chain with its joints [38]

2.6 Dual Quaternions

The most common method of representing rigid body orientation and translation in 3D space is homogeneous matrices. Recent studies have shown that dual quaternions are more efficient in terms of computer computational speed, mathematical robustness and interpolation, for details [46]. The dual quaternions contain the basic information which is contained in the transformation matrix. The main information is the axis of the rotation and the translation and rotation angle slide about the axis. The dual quaternion for the given screw axis of transformation is given by S= s+ εs^0, where $\varepsilon^2 = 0$ and, s and s^0 are the direction and the moment of the axis respectively. In general the displacement of the dual quaternion is represented as Eq.

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

The conjugate of the dual quaternion is

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

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

2.7 Kinematic Synthesis

Kinematic synthesis seeks to create or calculate an articulated system able to perform a given motion. One of the possible methods to create design equations for the dimensional kinematic synthesis is to equate the forward kinematics equations of the robot to the desired task positions. Kinematic equations give the synthesis equations for spatial linkage. In case of the matrix exponentials are used in the kinematic equation, then the unknown parameters will be the coordinates of the joint axes S_i where I = 1,………,n.

The dual quaternion is defined by exponential of the screw, corresponding to a relative displacement from the initial to a final position in terms of rotation and slide along the axis S. The product of exponentials defines the relative workspace from a reference configuration,

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

2.8 Design Equations for a Serial Chain

In order to find the axis and joint angles, the position of the end effector in a serial chain should be known. For a given set of positions ($[P_j]$, j= 1,……., m), the goal is to find the dimensions of the serial chain that will position the end effector at a given set of task. For each position $[P_j]$, there is a set of structural parameters and at least one joint parameter vector θ_j, such that the kinematic chain satisfy the relations.

$$[P_j]= [D(\theta_j)],\ j = 1,\dots,m\dots\dots\dots(2.16)$$

Consider a task with m position and $[P_1]$ as a reference position. From this position compute the relative displacements $[P_j][P_1^{-1}]= [P_{1j}]$, j =2,…..,m. Then express them in dual quaternions,

$$P_{1j} = Cos\frac{\hat{\phi}_1\ \ j}{2} + \sin\frac{\hat{\phi}_1 j}{2}P_{1j}, j = 2, \dots\dots, m\dots\dots(2.17)$$

The dual angle ϕ_{1j} define the rotation and slide along the axis P_{1j}.which is the displacement from the first to the j^{th} position. Relative displacement of (m-1) serial chain is given by

$$P_{1j} = e^{\frac{\Delta\hat{\theta}_1 S_1}{2}} e^{\frac{\Delta\hat{\theta}_2 S_2}{2}} \ldots\ldots\ldots\ldots e^{\frac{\Delta\hat{\theta}_n S_n}{2}}, j = 2, \ldots\ldots\ldots, m \ldots\ldots(2.18)$$

There will be 8(m-1) design equations in any given problem of the serial chains. There will be n joint axis S_i unknowns, where i = 1,… ,n and the n(m-1) pairs of joint parameters $\left(\Delta\hat{\theta}_{ij} = \Delta\theta_{ij} + \Delta d_{ij}\varepsilon\right)$. These design equations are then solved to find the position of the joint axes and the value of the joint angles, effectively setting the system dimensions.

Chapter 3

3 Motion Capture System

3.1 Introduction

In this chapter the motion capture system is presented. Vicon System is used for the motion capture. A detailed explanation is given from installation of the Vicon System to the extraction of the data. The camera system is arranged according to the requirement of the task. A custom model is constructed to track the trajectory. A database is set up for recording the data and exported to an excel file for further processing. A Mathematica program is used to get the relative positions of the end effectors.

3.2 Vicon Motion Capture System Working Principle

The Vicon motion capture system is used in biomechanics, virtual reality and 3D animations. The Motion Capture Lab is equipped with eight MX Bonita (Model) Infrared Cameras. The system measures the 3D position of the retro-reflection markers. The retro-reflection objects have the capability to bounce light back directly towards the source; this is very different from a typical object which produces incident reflection which bounces away from the object. The markers used for the motion capture are wrapped with a special tape which produces extremely bright retro reflections. The image coming back from the markers is extremely bright; the cameras can be set in such a way that all lights can be ignored other than the bright reflection spots from the markers. When a camera spots the blob of data coming from a marker, the camera fits a circle around the blob. The x and y coordinates of the circle are sent from each camera to the collection PC for the process. The x and y data from multiple cameras build three dimensional information. The markers should be in the line of sight of at least two cameras in order to accurately measure its 3D position in the space. The precise location and orientation of the cameras should be kept into consideration in order to get the 3D information. Triangulation equations are used within the Vicon system to calculate the position of the markers by the intersecting rays coming from the cameras.

3.2.1 **Installation of the Program**

The installation process consists of the following steps given below.

2.2.1.1 Insert the CD of the Vicon Bonita Camera and follow these steps.

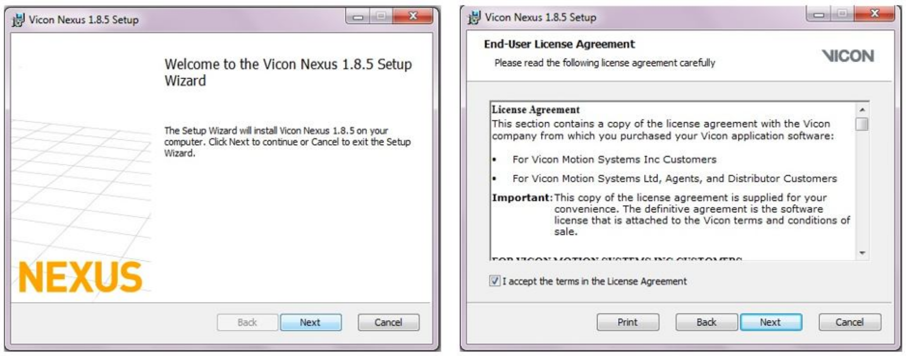


Figure 3-1Installation of Nexus

The CD consists the Nexus 1.8.5 version of the software, several updates have been released. The software can be updated to the latest version from http://www.vicon.com/Software/Nexus. The software version 1.8.5 is supported only on Windows 7. This version does not work on Linux or Windows 8. The user should be logged in from the Idaho State University's account from the Vicon site in order to make the update. Login and password can be obtained from the Motion Lab in the Garrison Hall (Occupational therapy Department, Idaho State University).

2.2.1.2 The typical installation feature should be selected

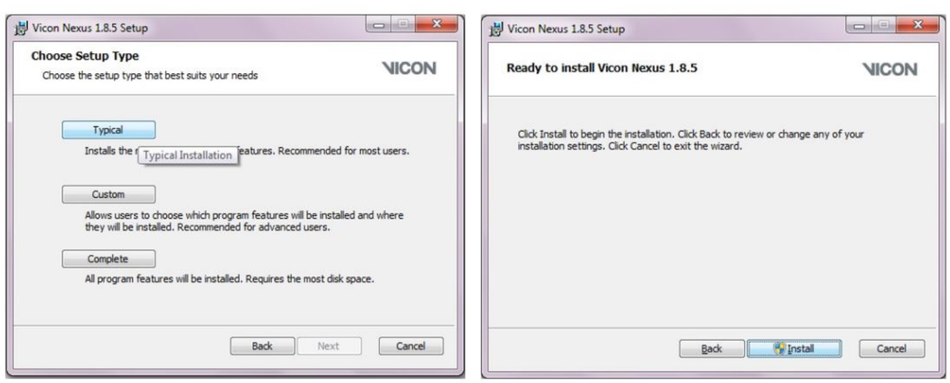


Figure 3-2Installation of the Nexus

3.2.2 Firewall and Antivirus Settings

Open the Vicon software from the icon on the desktop. Before launching the Vicon software and to perform it effectively the firewall settings should be turned off. Likewise the antivirus should also be turned off while the Vicon software is running.

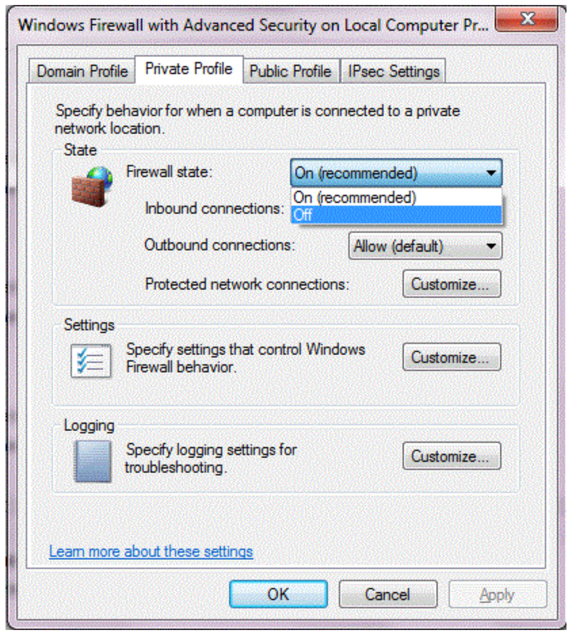


Figure 3-3Firewall Settings

The camera system runs via LAN card, it is important that the IP address should be changed to 192.168.10.1. The camera system should be turned on before launching the software. The Vicon hardware HASP software dongle should be plugged in at all times to run the software. As the software is launched the cameras will start appearing on the Browser pane on the left side of the screen. The workspace of each camera is seen by selecting the individual camera and the camera view option from the View Pane. In order to capture the motion in a small workspace, all the cameras should be zoomed in to get the blob reflection from all three markers on fingertip. The Focus and aperture needs to be adjusted to get a crisp view of each marker which is discussed in detail in the preceding sections.

3.2.3 Camera Setup

The motion Capture system is originally designed to capture the motion of the gait analysis of a human being. For capturing the motion of a hand while performing a task, the whole arrangements of the cameras have to be changed. The cameras should be a minimum of 2 meters away from the hand, to get the desired result. At all times during the motion capture, the marker should be captured by two cameras otherwise the data points will be missing and the Vicon system will not be able to give the 3D positions. The eight MX cameras were adjusted according to the volume area of the task.

3.2.4 Camera and software settings

Three markers are used on each end effector to record the task. The markers are placed perpendicular to each other as shown in the figure (add number). The inner ring of the lens adjusts the aperture, while the focus is adjusted by the outer ring shown in the figure (add number). The initial settings should be at minimum for the aperture, making the aperture wide open to get the maximum amounts of light reflecting back from the markers and the Focus should be kept at infinity setting and then adjusted accordingly. The Vicon system is designed to capture data from the markers which are specifically positioned separately. While using the markers almost touching each other at all times, a challenge is faced with the adjustment of Focus and aperture settings. The optimal setting to capture the marker is a tradeoff between the focus and the aperture settings. Vicon system confuses the markers, if a marker loses line of sight from two cameras at any given time. As mentioned earlier, the Vicon system recognizes the blob of light and then put a circle around it to gather the data. In order to get the crisp sharp image of the blobs, the centroid setting is changed to 0.65, strobe intensity to 0.8, circularized to 0.4, grayscale to “Auto” and threshold to 0.4 in the Vicon software. The configuration is saved for future use. All the reflecting surfaces should be covered or removed from the room where the image capturing is being performed.

Figure 3-4Camera setup for the potato grasping task

3.2.5 Aiming, Mapping, Calibration and Volume Origin

The 5 pointer wand is placed in the workspace. The camera should be adjusted in a way that the wand is exactly in the center of the 2D camera view. Cameras are individually aimed by pressing the aim button in the status pane. Getting the feedback from the cameras, the processing computer sets the cameras to a rough position in the actual workspace. The changes can be seen in real time in the perspective view as each camera is being aimed.

Mapping is done by removing all the objects from the workspace and ensuring that no reflecting surface is seen by the cameras. 2D view of all cameras is selected to individually monitor undesired light reflection coming from an object present in the work space. The map button is pressed from the system preparation tab on the right side of the screen, to map all the unwanted reflection sources from the workspace. The Vicon system will ignore any data coming from the object being mapped.

Calibration T frame wand is used to calibrate the Vicon cameras. The appropriate wand is selected from the drop down menu in the System preparation tab on the right hand side of the screen. Start button should be pressed from the system preparation tab to record the data. A wand is waved thoroughly in front of the cameras in order to record at least 1000 positions per camera in the capture volume. Once the system has recorded at least 1000 positions from each camera, the software will process the data by already known true distance of markers on the wand and give the exact camera positions in the perspective view. The camera position is solved with an arbitrary global origin and coordinate system. The system will set the main frame on any one of the camera and adjust the other cameras accordingly.

The volume origin adjusts the global origin to the desired the location in the capture volume. The T frame wand is placed at the center of the capture volume. The Set Volume button is pressed and the system starts tracking the calibration wand in real time and outputting the position/orientation of the wand. The start button is pressed and the global coordinate system snaps to the object coordinate system. The change is displayed in real time and the cameras are seen positioned accurately on the actual floor in the perspective view. The system is now fully calibrated and is ready for data capture.

3.2.6 Database

In order to capture the data of a particular task, a database should be set up. The database can be setup by pressing the “Show/hide Data Manager” button in the Browser Pane. A dialup box will pop out; in this section the management and storage hierarchy are set up for data collection. A new database is created by selecting the “New” button and selecting the “Generic template.eni” as the base. Root level folder and name should be given before pressing the create button. The database can be accessed by double clicking on the name.

3.3 System Preparation

As the Nexus System in the motion capture lab is specifically for the gait analysis, a new model must be developed in order to track the motion of a hand. The markers are placed on the desired locations, as shown in the figure 2.4. Markers are placed perpendicularly in set of three to get the position and the orientation of the end effectors. In order to get relative positions of the end effectors, markers are placed on the elbow to make it a reference. The hand is placed in the middle of the capture area with the markers placed on the elbow and the end effectors. In order to make a custom model, an initial capture session should be recorded. The data is saved and labeled later. In order to record data, a few things need to be specified in the data management mode. A desired session folder is selected or created in order to record data. Nodes can be selected to make a new hierarchy. Different sessions can be created and named individually. The trials are saved either in a session or sub-session. By double clicking on the session, it becomes active. The place for the storage of the data has been set up. The trail's information is given in the trial information box of the data capture view in the status pane on the right side of the Vicon software. The data are recorded for few seconds by clicking the start button on the capture section of the status pane. The file starts to record; the Vicon system in motion capture lab is capable of recording 200 frames per second. By pressing the stop button, the files are written to the disk and the recorded trial opens.

3.3.1 Custom Model

Open the saved file from the database management system. The perspective view will show a blank space. Click the reconstruction and label button from the Manu bar. The trial will be automatically loaded. In order to make a new subject, the subject option is selected from the Browser pane. Name a new subject and start constructing by clicking the drop down option. Markers, segments and joints can be seen in the drop down menu of the newly created subject. Add the number of markers which are used in the data capture. The markers can be named accordingly in order to make them identifiable for segment construction. The segments are constructed by selecting the parent marker then the child marker and clicking the create button. The segments can be named accordingly to make them identifiable as well. As the segments are constructed, the construction progress can be seen in the view pane. Once the segments are complete, the segments need to be joined together by using different type of joints. For the fingers and thumb, the ball joint is used to join the different segments. The configuration is labeled by individually selecting each marker and segment. The different parts of the custom model can be seen from the browser pane on the left side of the screen. The custom model has to be saved for multiple sessions.

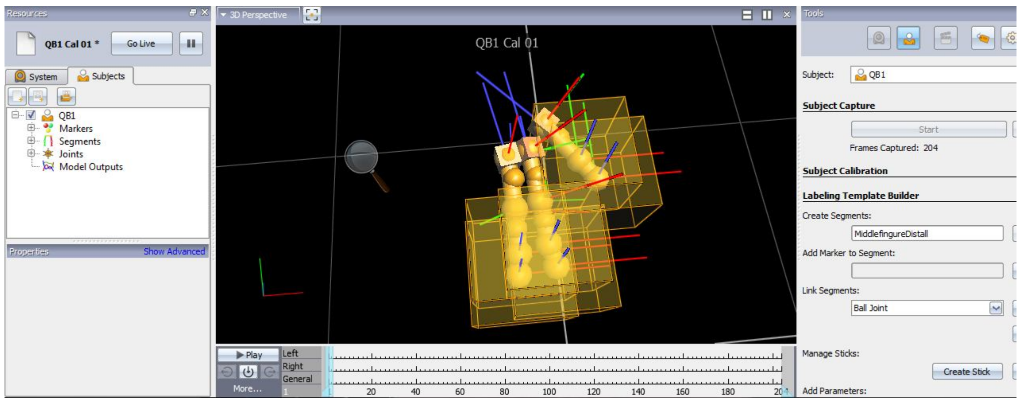


Figure 3-5Custom human hand model view 1

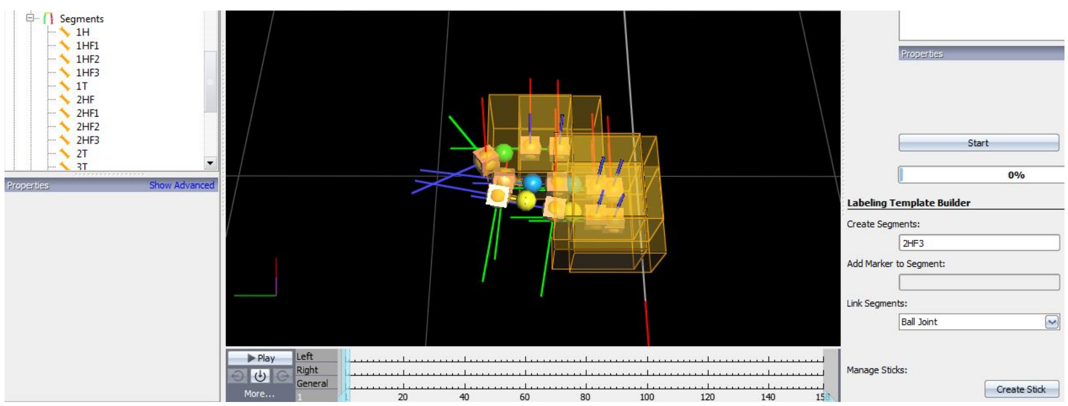


Figure 3-6Custom human hand model view 2

3.3.2 Data Capturing and labelling

The set of three markers is placed on the desired data collection positions. The markers are placed on each end effector and the elbow. The potato to be grasped is placed in the motion capture space. The trail is recorded by selecting a session in the data management system. The task is performed and the data is recorded as explained in the Custom Model section. By pressing the stop button the files are saved in the data management system and the trial is automatically loaded. The reconstruct and label button is clicked from the Menu bar to reproduce the task. The Vicon software displays the markers in the View pane. The task can be replayed by clicking the play button at the bottom of the View pane. In order to get the data, the custom model is opened from the Subject menu in the Browser bar. The markers are labelled and it automatically defines the segments based on the custom model.

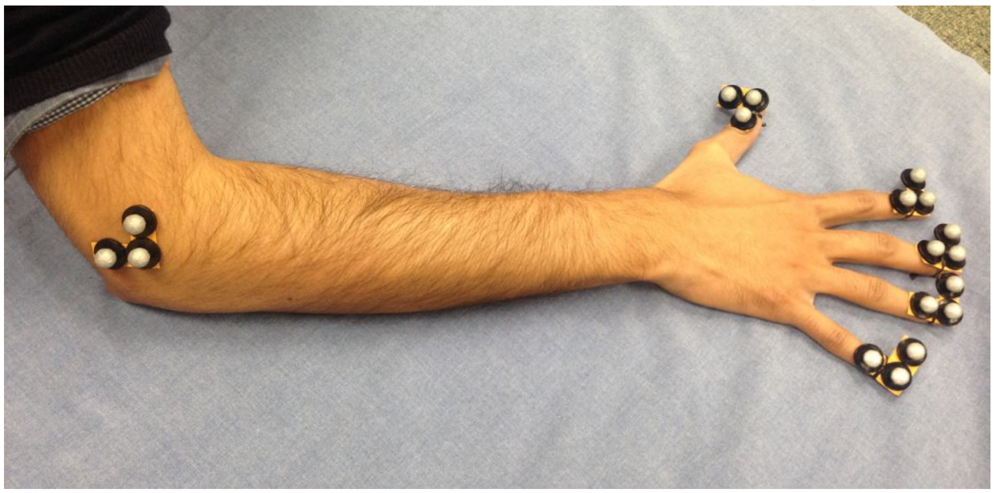


Figure 3-7Human hand with pointers for grasping and trajectory information

3.4 Data Export

The markers have been labeled and defined according to the custom model. The Cartesian positions of each marker can be obtained by selecting the Graph option from the View Pane. The marker is selected individually from the browser pane and the “x, y and z components” option is selected from the view pane in order to see the location of the marker at each frame. The graph will show the marker location by each component (x, y and z). The view can be changed to perspective to see the trajectory of each marker. The Cartesian coordinates are exported into an excel file by using the “ExporttoASCII” option in the Status pane. The data is exported into excel file, displaying the name of each marker with its Cartesian coordinates.

3.5 Potato Sampling

In order to perform the task, different sizes and types of potatoes are collected and the dimensions are documented. Potatoes are segregated according to type and size. The potatoes are numbered by size and weight, as shown in the figure 3.8. In order to create a 3D model of the potatoes, Faro Scanner is used. The potatoes are imported in Solid work with the help of Idaho Museum of Natural History. The CAD model can be used for force and different type of grasping techniques. The figures 3.9 and 3.10 show the process of importing the potatoes.

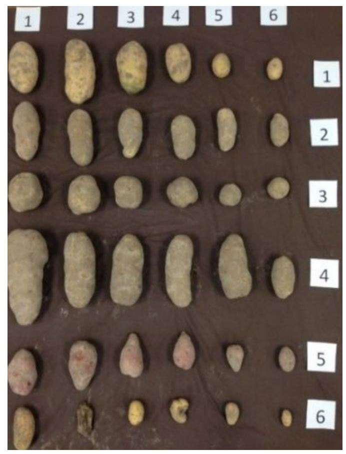


Figure 3-8(Above) Segregation of potato by different size and shape.

Figure 3-9(Below) CAD import of the potatoes in the Solid Work using Faro Scanner from Idaho Museum of Natural History.

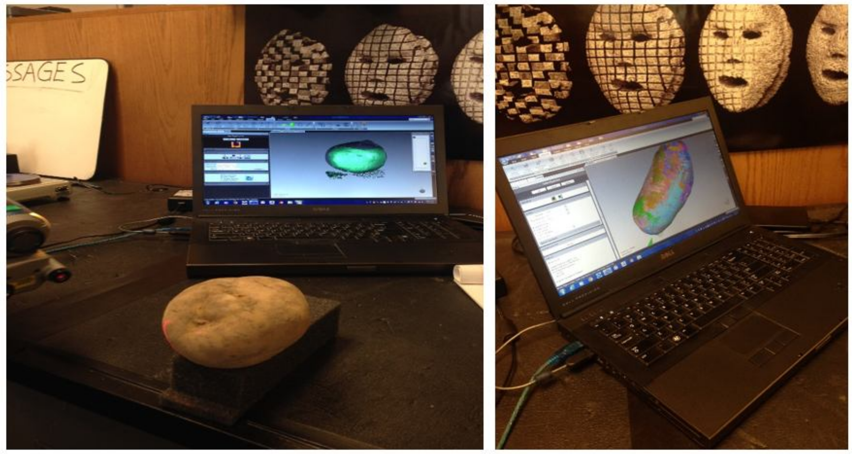


Figure 3-10(Above) CAD import of the potato in the Solid Work using Faro Scanner from Idaho Museum of Natural History

3.6 Data Processing

The positions of the markers are exported into an excel file. The positions of each frame (200 frames per second) are represented in Cartesian coordinate system. In order to make the data useful, the data are processed by a Mathermatica program, attached in Appendix 9.2. The location and orientation of each end effector is obtained from the Cartesian coordinates of the three markers. By subtracting marker 2 from marker 1, the vector representing the X axis of the frame is obtained. Similarly subtracting the marker2 from marker 3 gives the Y axis of the frame. The cross multiplication of both the vectors gives the Z axis. The markers are shown in the figure 3-11. Hence we have the frame attached to each of the End Effector. The positions obtained from the Vicon Motion System are absolute.

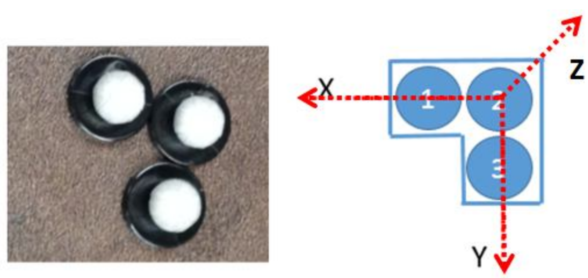


Figure 3-11Retro-reflection markers for tracking the trajectory of the task.

The markers placed on the elbow plays an important role to give the relative positions. The relative frame for each end effector is calculated by taking the inverse of elbow frame and multiplying it with the absolute finger frame.

$$[F1_{rel}] = [T]^{-1}.[F1_{absolute}] \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

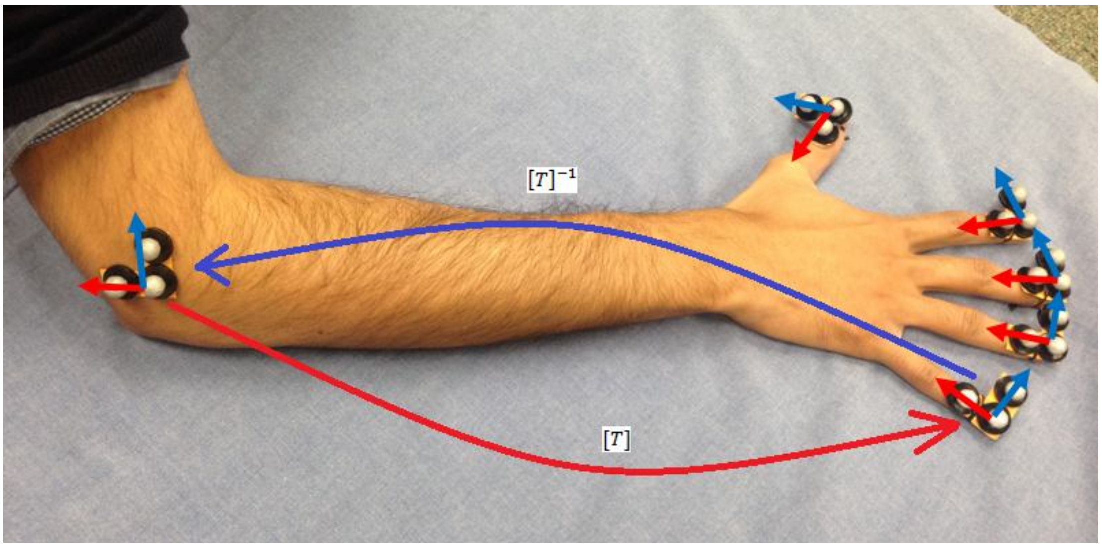


Figure 3-12Tranformation of absolute frame to elbow frame.

The location and orientation is represented in a homogenous matrix. The homogenous matrix is converted into a Dual Quaternion representation. The topology selection pays an important role in giving the number of positions required to define a task. Depending on the number of the positions required to solve the topology selected, the positions can be extracted from the Excel file. The figure 2.11 and 2.12 represent the frames of the markers for every end effector. The smallest frame represent the 13 positions (location and orientation) of the finger 1's trajectory. The rest of the finger's position is represented with frames of different size. Both views represent the same task with 13 positions for each end effector from different views.

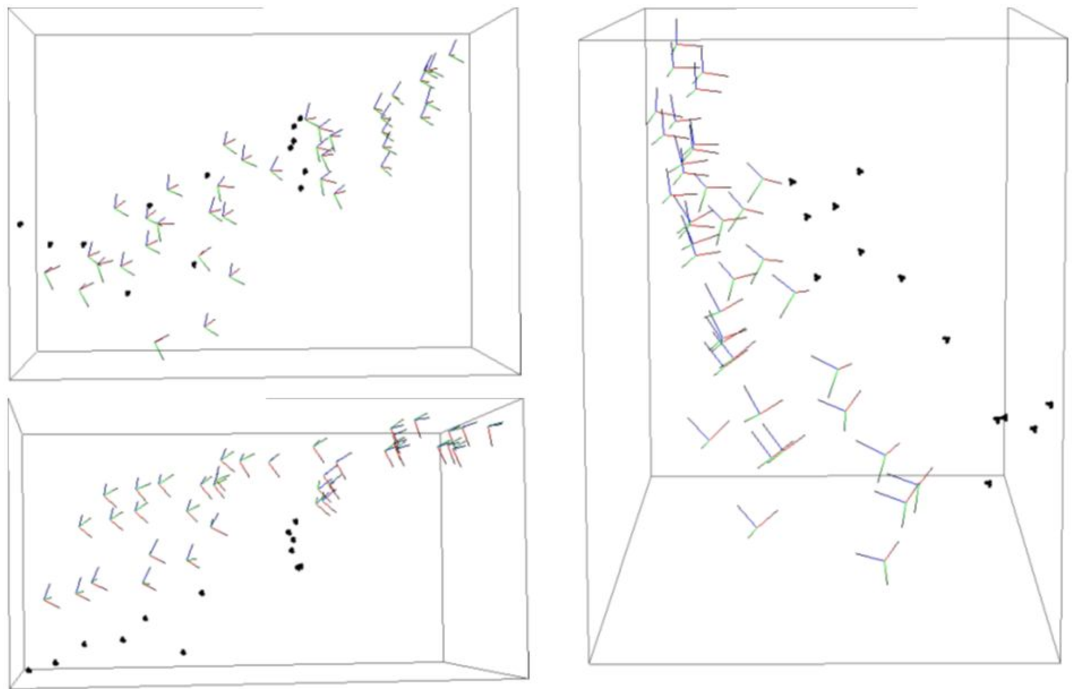


Figure 3-13Tragectory Views of the Potato grasping task

3.7 Summary

In order to record the trajectory of the potato grasping task, a custom model is developed to give the positions and orientation at each end effector. The motion capture system can be improved by synchronized with EMG signals in the future to consider the force at each fingertip, which will help in evaluating grasping and manipulating abilities of the hand. The input task positions are used in the numerical solver ArtTreeKS to dimension the link of the hand design.

Chapter 4

4 Hand topology

4.1 Introduction

A multi-finger hand can be treated as a kinematic chain with a tree topology, where several end effectors are joined through common joints. This tree topology has not been widely studied. Kinematic analysis of the robotic hand can be found in [39], [40] and [41]. The theoretical developments of the kinematic design can be found in [42]. The first step while dealing with the exact kinematic synthesis is to calculate the number of positions which fully define the workspace of the chain. In the case of the tree topology the end effectors should have the same number of positions which means that it is a coordinated action of all the end effectors.

4.2 Tree Topology

The current task under study will be solved using the tree topology. Tree topology is the compact representation of the structure in terms of common joints and ending at multiple end effectors. The use of the graph to represent a mechanism was proposed by Crossley [43]. The graph theory is used in order to define the tree topology; the approach of Tsai [44] is followed. The identifiable joints are defined as edges and the links are defined as vertices of the graph.

For the use of kinematic synthesis, the mechanism is represented as a rooted graph with their root vertex being fixed with the reference system. The rooted, connected graph representation of a kinematic structure is denoted as G (V, E), where V is the vertices and E are the edges which are connecting the vertices. A tree topology is denoted as

$$\text{Serial Chain} - (\text{Branch}_1, \text{Branch}_2, \ldots\ldots\ldots, \text{Branch}_n) \ldots\ldots\ldots (4.1)$$

Where the serial chain is the most common joint branching into multiple branches. The dash indicates a branching with the branches contained in the parenthesis. Each branch is further characterized by its type and number of joints. In the case of revolute joints, the type of the joint is dropped and only the number of the joints is indicated.

4.3 Reduced tree Topology

After the initial graph has been represented as a serial chain, it needs to be contracted. The graph is contracted so that each of the kinematic serial chains represents a set of joints instead of an individual kinematic joint. The main objective of the reduced graph is to convert a hybrid graph, with loops, into an equivalent tree graph which performs the exact same dimensional synthesis. The figures below show the compacted version of the graph. The graph under consideration is 2R-(2R, R-(R, R)) or 2- (2, 1- (1, 1)) chain, with two branches, one of them branch again on two additional branches. The root vertex is represented by a double circle.

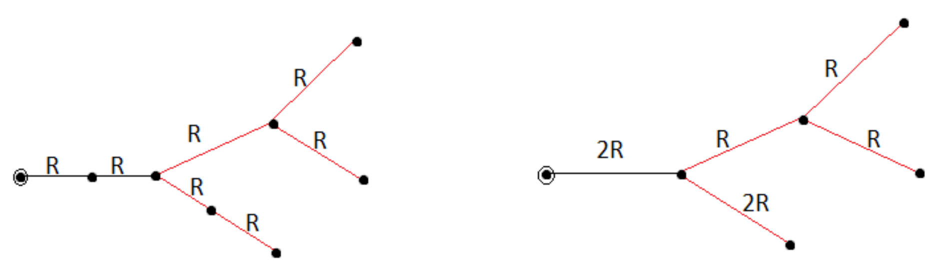


Figure 4-1The graph on the left shows the extended tree topology and the same topology is compacted in the right graph. The root vertex is represented by the double circle.

In the tree topology, a vertex can be connected to several edges which have several branches. Only revolute joints are being considered, so the "R" representing the type of joint is dropped while representing the tree topology, so instead of using 2R-(2R, R-(R, R)) the 2- (2, 1- (1, 1)) representation is used.

4.4 Matrix Associated to a Graph

Matrices, which are used to characterize the tree topologies are covered in this section. They are used to identify connectivity in the graph and paths from root to end effector in the graph. For the tree mechanism e × (v – 1) gives the path matrix [T] where vertices v, edges e and branches b. $\boldsymbol{i}$ and $\boldsymbol{j}$ represent the edge and vertex respectively.

The path matrix is defined as,

$$
\begin{array}{cc}
 & \text{Vertex } j \\
[\mathrm{T}]= & \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,v-1} \\ t_{2,1} & t2{,}2 & \cdots & t_{2,v-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_{e,1} & t_{e,2} & \cdots & t_{e,v-1} \end{bmatrix} \text{Edge } i \ldots\ldots\ldots\ldots \text{(4.2)}
\end{array}
$$

Where, $t_{\boldsymbol{ij}}$ is 1 if the edge $\boldsymbol{i}$ lies on the path originating at the root, and terminating at the vertex $\boldsymbol{j}$ or 0 otherwise.

The incident matrix is defined as

$$
\begin{array}{cc}
 & \text{Edge } j \\
[\mathrm{B}]= & \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,e} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,e} \\ \vdots & \vdots & \ddots & \vdots \\ b_{v-1,1} & b_{v-1,2} & \cdots & b_{v-1,e} \end{bmatrix} \text{Vertex } i \ldots\ldots\ldots \text{(4.3)}
\end{array}
$$

Where, 1 if vertex $\boldsymbol{i}$ is connected to edge $\boldsymbol{j}$ or 0 otherwise.

4.5 Solvability of Tree Topologies for Exact Synthesis

A kinematic chain is considered to be solvable if a positive rational number of positions greater or equal than two are obtained, for which the exact dimensional synthesis yields a finite number of solutions. The task sizing is important when dealing with the tree topologies so that the system equations can be solved simultaneously while not over constraining any of the branches.

The maximum number of positions for the overall system is computed as follows[45]: let $\mathbf{D^e_j}$ be an e×1 vector containing the joint d.o.f. (Degree-of-freedom) for each edge of the contracted graph, and $\mathbf{D^e_s}$ be the e×1 vector containing the number of independent structural parameters for each edge of the contracted graph. The $\mathbf{D^n_{ee}}$ the b×1 vector containing each end effector d.o.f. (Degree-of-freedom), and $\mathbf{D^n_c}$ the b×1 vector with the number of additionally imposed constraints (if any) for each branch. Vectors B is defined as b×1 vector of ones corresponding to branches or end-effector. Vector **E** is defined as e×1 vector defines the edges of the graph considered. The maximum number of the positions for the overall graph is given by the following equation.

$$\boldsymbol{m} = \frac{\boldsymbol{D^e_s.E - D^n_c.B}}{\boldsymbol{D^n_{ee}.B - D^e_j.E}} + \mathbf{1} \ldots\ldots\ldots\ldots \quad (4.4)$$

In order to calculate the overall solvability of the graph, the solvability of the sub-graphs should be calculated. The end effector path matrix [T] and incidence matrix [B] of the graph [45] can be used to find the vectors $\mathbf{E_i}$ and $\mathbf{B_i}$ containing the edges and branches for a given sub-graph. There are 2^b-2 possible sub-graphs for any rooted graph; these will exclude the full graph and the null graph.

The solvability of the sub-graph i, the number of the positions can be calculated by the following formula

$$\boldsymbol{m_i} = \frac{\boldsymbol{D^e_s.E_i - D^n_c.B_i}}{\boldsymbol{D^n_{ee}.B_i - D^e_j.E_i}} + \mathbf{1} \ldots\ldots\ldots\ldots \quad (4.5)$$

Different topologies were selected and checked for the solvability of the graph and sub-graphs. In order for the sub-graph to be solvable, it should be either greater than or equal to m, infinity or a negative number.

4.6 Topology 3- (4,5,5,5)

The hand has a wrist and four fingers. The figure on the left shows the tree topology of the selected hand. The middle graph shows the representation of the graph in terms of edges and vertices. The revolute joints are shown as edges; the links are shown as vertices and the end effector are shown as square blocks. The figure on the right hand side shows the contacted graph. The number of revolute joints in each chain is shown accordingly. As mentioned earlier, in case of revolute joints, the "R" can be dropped off leaving the number of the joints.

Segment	Number of Joints
Wrist	3
Finger 1	4
Finger 2	5
Finger 3	5
Finger 4	5

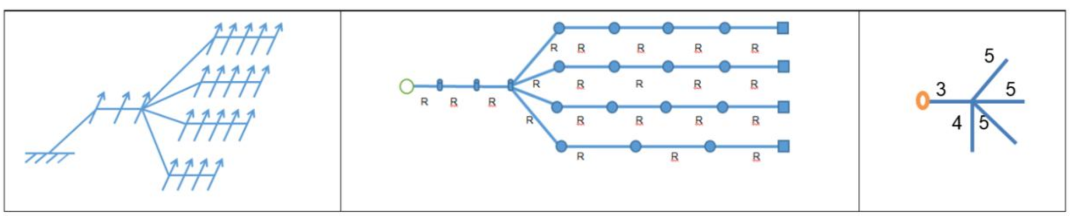


Figure 4-2(Left) Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph.

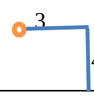
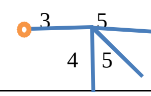
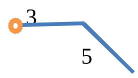
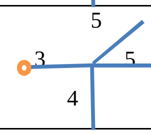
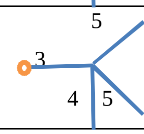
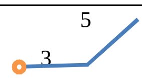
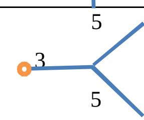
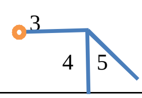
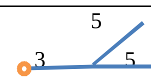
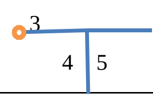
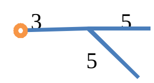
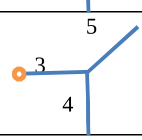
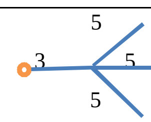
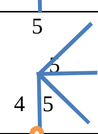
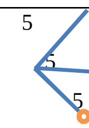
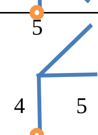
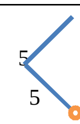
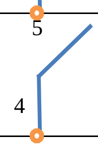
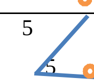
$$[T] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad [E] = [T] \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad ; \quad [B] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D^e_j = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 5 \\ 5 \end{bmatrix} \quad ; \quad D^e_s = \begin{bmatrix} 12 \\ 16 \\ 20 \\ 20 \\ 20 \end{bmatrix}$$

$$m = \frac{\begin{bmatrix}12\\16\\20\\20\\20\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}}{\begin{bmatrix}6\\6\\6\\6\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}3\\4\\5\\5\\5\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}} + 1 = 45$$

In order for a graph to be solvable, m should be a positive whole number. Now the solvability of the sub graphs is checked individually. The topology is solvable according to the result for m=45 positions.

Solvability of Sub-graphs

	-27		69
	-15		69
	-15		69
	-15		-51
	∞		-51
	∞		-51
	∞		∞
	-75		-19
	-27		-9
	-11		-9

4.7 Topology 3, (3,4,4,4)

The hand has a wrist and four fingers. The figure on the left shows the tree topology of the selected hand. The middle graph shows the representation of the graph in terms of edges and vertices. The revolute joints are shown as edges; the links are shown as vertices and the end effector are shown as square blocks. The figure on the right hand side shows the contacted graph. The number of revolute joints in each chain is shown accordingly. As mentioned earlier, in case of revolute joints, the “R” can be dropped off leaving the number of the joints.

Segment	Number of Joints
Wrist	3
Finger 1	3
Finger 2	4
Finger 3	4
Finger 4	4

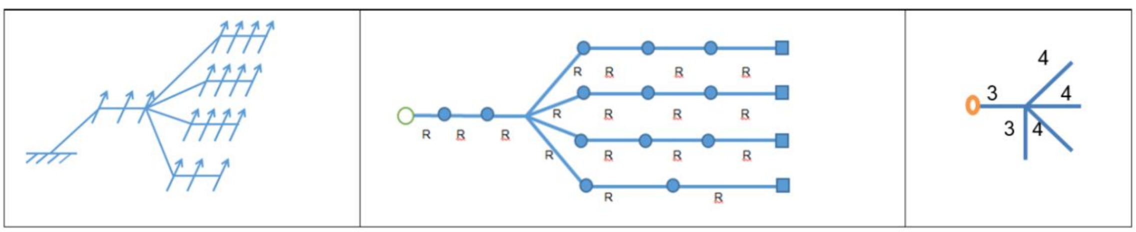


Figure 4-3 Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph.

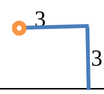
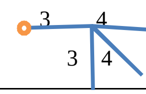
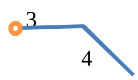
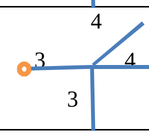
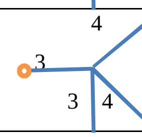
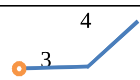
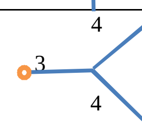
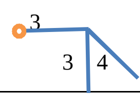
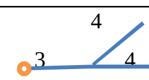
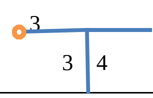
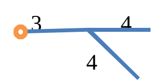
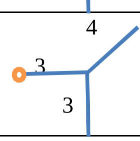
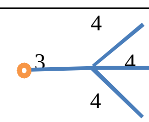
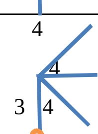
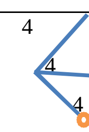
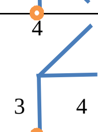
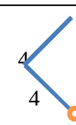
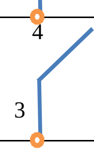
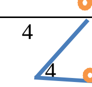
$$[T] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad [E] = [T] \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad ; \quad [B] =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad D^e_j = \begin{bmatrix} 3 \\ 3 \\ 4 \\ 4 \\ 4 \end{bmatrix} \quad ; \quad D^e_s = \begin{bmatrix} 12 \\ 12 \\ 16 \\ 16 \\ 16 \end{bmatrix}$$

$$m = \frac{\begin{bmatrix}12\\12\\16\\16\\16\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}}{\begin{bmatrix}6\\6\\6\\6\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}3\\3\\4\\4\\4\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}} + 1 = 13$$

In order for a graph to be solvable, m should be a positive whole number. Now the solvability of the sub graphs is checked individually. The topology is solvable according to the result for m=13 positions.

Solvability of Sub-graphs

	∞		15
	-27		15
	-27		15
	-27		45
	21		45
	21		45
	21		21
	21		∞
	45		-15
	-27		-15

4.8 Topology 3, (4,4,5,5)

The hand has a wrist and four fingers. The figure on the left shows the tree topology of the selected hand. The middle graph shows the representation of the graph in terms of edges and vertices. The revolute joints are shown as edges; the links are shown as vertices and the end effector are shown as square blocks. The figure on the right hand side shows the contacted graph. The number of revolute joints in each chain is shown accordingly. As mentioned earlier, in case of revolute joints, the "R" can be dropped off leaving the number of the joints.

Segment	Number of Joints
Wrist	3
Finger 1	4
Finger 2	4
Finger 3	5
Finger 4	5

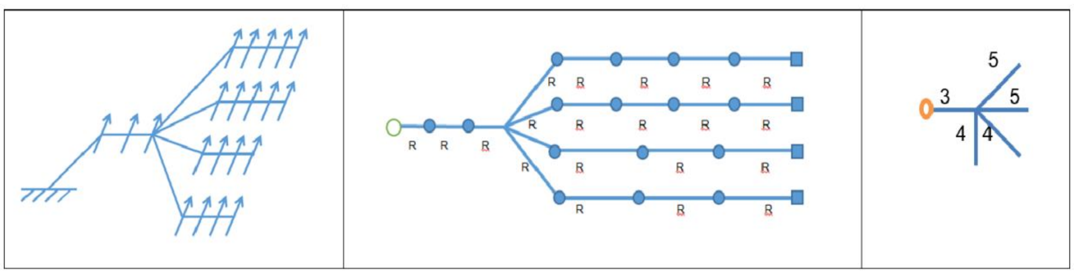


Figure 4-4 Kinematic sketch of 3-(4,5,5,5) topology; (Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph.

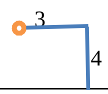
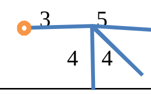
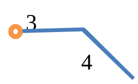
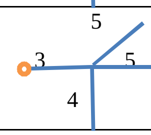
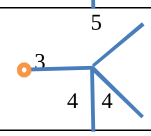
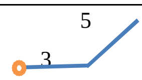
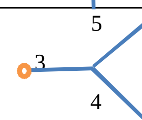
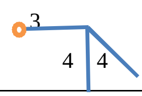
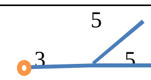
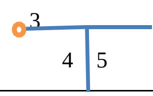
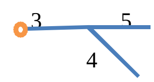
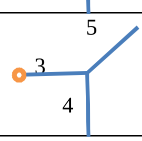
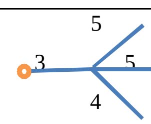
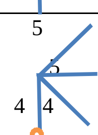
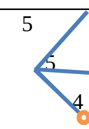
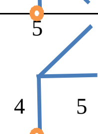
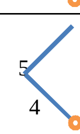
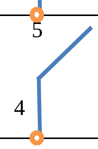
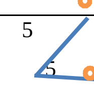
$$\{T\} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad [E] = [T] \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad ; \quad [B] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D^e_j = \begin{bmatrix} 3 \\ 4 \\ 4 \\ 5 \\ 5 \end{bmatrix} \quad ; \quad D^e_s = \begin{bmatrix} 12 \\ 16 \\ 16 \\ 20 \\ 20 \end{bmatrix}$$

$$m = \frac{\begin{bmatrix}12\\16\\16\\20\\20\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}}{\begin{bmatrix}6\\6\\6\\6\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}3\\4\\4\\5\\5\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}} + 1 = 29$$

In order for a graph to be solvable, m should be a positive whole number. Now the solvability of the sub graphs is checked individually. The topology is solvable according to the result for m=29 positions.

Solvability of Sub-graphs

	-27		33
	-27		69
	-15		33
	-15		∞
	45		-51
	∞		∞
	∞		69
	∞		-27
	-27		-11
	-11		-9

4.9 Topology 2, (3,3,4,4)

The hand has a wrist and four fingers. The figure on the left shows the tree topology of the selected hand. The middle graph shows the representation of the graph in terms of edges and vertices. The revolute joints are shown as edges; the links are shown as vertices and the end effector are shown as square blocks. The figure on the right hand side shows the contacted graph. The number of revolute joints in each chain is shown accordingly. As mentioned earlier, in case of revolute joints, the “R” can be dropped off leaving the number of the joints.

Segment	Number of Joints
Wrist	2
Finger 1	3
Finger 2	3
Finger 3	4
Finger 4	4

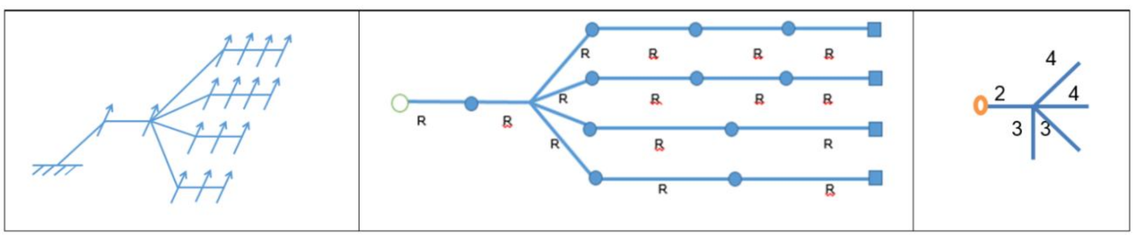


Figure 4-5(Left) Kinematic sketch of 3-(4,5,5,5) topology ;(Middle) Tree graph in terms of vertices and edges; (Right) Reduced graph.

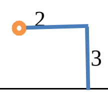
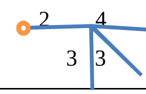
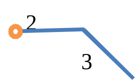
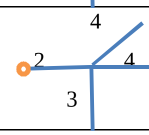
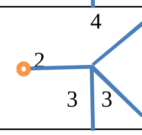
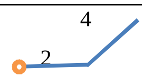
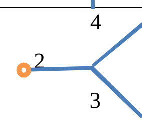
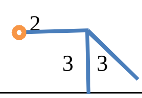
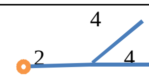
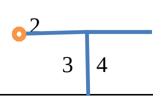
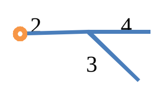
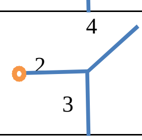
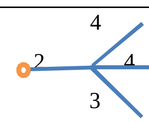
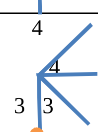
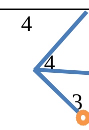
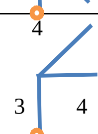
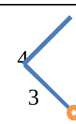
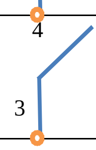
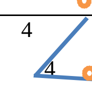
$$[T]=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad ; \quad [E]=[T]\times\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \quad ; \quad [B]=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1 & 1\\ 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

$$D^{e}_{j}=\begin{bmatrix}2\\3\\3\\4\\4\end{bmatrix} \quad ; \quad D^{e}_{s}=\begin{bmatrix}8\\12\\12\\16\\16\end{bmatrix}$$

$$m = \frac{\begin{bmatrix} 8 \\ 12 \\ 12 \\ 16 \\ 16 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}{\begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} + 1 = 9$$

In order for graph to be solvable, m should be a positive whole number. Now the solvability of the sub graphs is checked individually. The topology is solvable according to the result for m = 9 positions.

Solvability of Sub-graphs

	21		9
	21		11.4
	∞		9
	∞		13
	9		21
	13		13
	13		∞
	15		45
	45		-27
	-27		-15

4.10 Hand Selection

The hand is selected on the basis of the solvability of the graph and sub-graphs. The potato grasping task is accomplished by 4 end effectors, so the selected hand should have the same number of end effectors to reach all the positions in the trajectory. The solvable topologies are given in the table below

Topology	Number of positions
3-(5,5,5,5)	93
3-(4,5,5,5)	45
3-(4,4,5,5)	29
3-(3,4,4,4)	13
2-(3,3,4,4)	9
3-(3,3,3,3)	7.66
2-(3,3,3,3)	6.66
3-(3,2,2,3)	5.7
2-(2,2,3,3)	5

4.11 Conclusion

The selected hand has 13 numbers of positions, which completely define the trajectory of the complete task (grasping and release). The trajectories with 93, 45 and 29 positions have certain positions which have identical orientation of the attached frame. Number smaller than 13 do not completely define the task. The input task positions along with the selected hand topology are used in the numerical solver ArtTreeKS to dimension the link of the hand design.

Chapter 5

5 Kinematic Synthesis

5.1 Hand Task

The hand task is defined as a series of finite positions. The position includes the location and orientation of each fingertip. These positions are used to calculate the joint variables. The topology selected for the grasping task is 3-(3,4,4,4) and needs 13 positions to define the task. The 13 positions (location and orientation) are given as input to the solver in the form of dual quaternion. The dual quaternions are used due to their efficient computational capabilities.

5.2 Kinematic Synthesis

The main goal of the kinematic synthesis is to find the position and orientation of the joint axes which are able to perform the task. The number and type of joints have been already defined in chapter 4. The solvability of the tree topology has also been calculated. These types of problems are called form-to-motion problems in which sequence of end effector task positions (location and orientation) is provided to the solver and it has to calculate the set of angles and joints which will be able to perform the particular motion. The forward kinematics approach [47] is used, but expressed in dual quaternions and formulated as relative displacement. See [48] for a complete description of the approach.

The input data is m-1 transformations P_{1j}= $\cos\frac{\Delta\phi_{1j}}{2} + sin\frac{\Delta\phi_{1j}}{2} P_{1j}, j = 2, \dots, m,$ [50] defining the task. The output is in the form of Plücker coordinates and set of joint variables which can reach the task positions, are measured from a reference configuration.

5.3 Forward Kinematics

Each branch of the tree topology is treated as an individual kinematic chain which shares a joint with other branches. A kinematic serial chain with n joints can be written as products of exponentials of the screws corresponding to the joint axes, for detail see [49]. Instead of calculating the exponents using the matrix algebra, the Clifford algebra is used. Dual quaternions express the displacements in the projected space. The exponentials of a screw represented by the Clifford algebra element J= (1+ με) S of the axis S, where μ is the pitch which relate to the slide d and the rotation θ along and about the screw and hence yields a finite displacement [50],

$$e^{\frac{\theta}{2}J} = \left(cos\frac{\theta}{2} - \frac{d}{2}sin\frac{\theta}{2}\varepsilon\right) + \left(sin\frac{\theta}{2} - \frac{d}{2}cos\frac{\theta}{2}\varepsilon\right)S = \cos\frac{\hat{\theta}}{2} + \sin\frac{\hat{\theta}}{2}\mathrm{S} \text{............ (5.1)}$$

Forward kinematics of relative displacements of a series chain with n joints are calculated by

$$\hat{Q}(\Delta\hat{\theta}) = e^{\frac{\Delta\hat{\theta}_1}{2}S_1 \frac{\Delta\hat{\theta}_2}{2}S_2 \cdots\cdots \frac{\Delta\hat{\theta}_n}{2}S_n}$$

$$= \left(cos\frac{\Delta\theta_1}{2} - sin\frac{\Delta\theta_1}{2}S_1\right)\dots.\left(cos\frac{\Delta\theta_n}{2} - sin\frac{\Delta\theta_n}{2}S_n\right), \text{.....[50](5.2)}$$

where $\Delta\hat{\theta} = \theta_j - \theta_1 + \varepsilon(d_j - d_1)$ contains the joint variables.

5.4 Synthesis Design Equations

In order to reach the desired task the dimensioning of the forward kinematics equation needs to be done, so that every end effector can reach the desired task positions.

$$\hat{P}_{1j} = e^{\frac{\Delta\hat{\theta}_1}{2}S_1} e^{\frac{\Delta\hat{\theta}_2}{2}S_2} \dots..e^{\frac{\Delta\hat{\theta}_{nj}}{2}S_n}, j = 2, \dots, m \text{.......[50](5.3)}$$

There will be 8 (m-1) design equations per finger, considering the dual quaternions with 8 components.

There is no explicit limitation imposed on the link dimensions or the placements. The resulting hand design will be performing the human hand task while having a non-anthropomorphic structure. In the selected topology, there are three common joints in series at the base which are connecting four figures. The first finger has three revolute joints while the rest of the three fingers have four revolute joints.

The total system of equation is obtained by using Eq. 5.3. Each of the serial chain branches, a set of design equation [50] is written as

$$P_{1j}^{k} = e^{\frac{\Delta\theta_{1j}}{2}S_1} e^{\frac{\Delta\theta_{2j}}{2}S_2} e^{\frac{\Delta\theta_{3j}}{2}S_3} e^{\frac{\Delta\theta_{4j}^{k}}{2}S_4^k} \dots . e^{\frac{\Delta\theta_{6j}^{k}}{2}S_6^k}, k \in \{1\}; j = 2, \dots \dots , m \dots \dots \dots \quad (5.4)$$

$$P_{1j}^{k} = e^{\frac{\Delta\theta_{1j}}{2}S_1} e^{\frac{\Delta\theta_{2j}}{2}S_2} e^{\frac{\Delta\theta_{3j}}{2}S_3} e^{\frac{\Delta\theta_{4j}^{k}}{2}S_4^k} \dots . e^{\frac{\Delta\theta_{7j}^{k}}{2}S_7^k}, k \in \{2,3,4\}; j = 2, \dots \dots , m \dots \dots \dots \quad (5.5)$$

where k identifies the finger or the kinematic chain and j identifies the index of the position. The fingers are denoted by k=1 (thumb), k=2 (index), k=3 (middle) and k=4 (baby or pinky finger). There are six or seven degrees of freedom in four serial chains, depending on the finger. Exact synthesis does not apply to six or more degree of freedom, but by having the three common joints (at the wrist), the design equation is stated as a whole.

5.5 Kinematic Solver

The four chains are solved simultaneously to reach the number of the positions (location and orientation) of the desired task. The methodology [48] has been developed to solve serial chains up to 5 degree of freedom. A similar methodology is used along with a global solver, which is actually composed of a genetic algorithm paired with a Levenberg-Marquardt local optimizer [50]. The main objective of the solver is to apply generalized inverse kinematics to adjust the joint axes and joint angles to follow the desired task given as an input to the solver.

5.6 Dimension of Equation Set

Unit dual quaternions have a dimension of eight. The equations have two implicit constraints $\hat{q}.\hat{q}^0 = 1$ and $\hat{q}.\hat{q}^* = 0$, which reduces the independent dimension to 6. In the similar way for the Plucker components of each of them is reduced from six to four because of the two implicit constraints ||s||=1 and $s.s^0=0$. The variables x and the independent variable x^0 can be written as [50]

$$x = n(6 + (m-1)) \ldots\ldots..(5.6)$$

$$x^0 = n(4 + (m-1)) \ldots\ldots\ldots (5.7)$$

And similarly the number of the equations *f* and independent equation f^0can be written as [50],

$$f = 8b(m-1) \ldots\ldots.(5.8)$$

$$f^0 = 6b(m-1) \ldots\ldots.(5.9)$$

Applying the above equations to the model. Where m (the number of task positions to define a trajectory) has already been calculated to be 13 and the number of branches is 5. The number of structural parameters and joint variables comes out to be 324 by using the equation number 5.6. The total number of the equations which are solved simultaneously in the solver are 384.

5.7 Input to the Solver

The task positions for each end effector are given as input to the solver. The solver use these positions to calculate the joint axes for each joint. The input 13 positions for 4 end effectors are given in the form of dual quaternions in Appendix 9.2.

5.8 Output solution

It took 28 iterations with an error of 5.723e-12. Which means that all the end effectors are reaching exactly the input positions. The output file contains the joint angles from the 1st position to the 13th, which will help each link to reach the next position by moving the axis by the angle provided in the solution. Each of the 18 axis is given in their initial location (position and orientation) in Plucker coordinates and can be found in Appendix 9.1. The location of the axis and the joint angles help in reaching the end effectors' 13 positions for all fingers. The figure 5.1 shows the ArtreeKS output model for the hand.

Figure 5-1 Shows the output solution number 1 which took 11 iterations and (Right) shows the second solution which took 28 iterations to give this result.

The Plucker Coordinates of the 18 axes are given below. The axes are in their initial position.

Table 1Plucker Coordinates of the output axes

Wrist Joint Axis 1	({ 7.157135813591766205*10^-01, -1.575513640601141629*10^-01, 6.803907973663511077*10^-01 }, { -1.208927104713507816*10^01, 2.646983236127513237*10^02, 7.401046225448509119*10^01 })
Wrist Joint Axis 2	({ -9.808627099816861206*10^-01, 1.675939542232079416*10^-01, -9.909899431988179741*10^-02 }, { -9.354342835341540763*10^00, -9.299756613302859876*10^01, -6.468787924431606484*10^01 })
Wrist Joint Axis 3	({ 7.252855382690187591*10^-01, -2.927661188456781582*10^-01, -6.230962105757488212*10^-01 }, { 1.302090449905773895*10^01,

	-1.795349296594465898*10^02, 9.951210941354081285*10^01 })
Finger 1 Joint Axis 1	({ 2.991778491846188714*10^-01, 7.208914632864753536*10^-01, -6.251464730108852175*10^-01 }, { -7.527989944831786318*10^01, -2.178641780706298903*10^02, -2.872582863115615623*10^02 })
Finger 1 Joint Axis 2	({ -8.098108731880641065*10^-01, -4.232658046337826052*10^-01, 4.062664252606926762*10^-01 }, { 3.939534111227553836*10^01, 1.008955573484752790*10^02, 1.836440577211744483*10^02 })
Finger 1 Joint Axis 3	({ -3.715593471713062312*10^-01, 4.562644560060451493*10^-01, 8.085582215988779131*10^-01 }, { -5.023365906115425439*10^00, 2.798259326672847465*10^02, -1.602124646575747988*10^02 })
Finger 2 Joint Axis 1	({ 1.295895578377645185*10^-01, -8.724890280714954205*10^-01, -4.711363310065036503*10^-01 }, { 4.813266342984618973*10^01, -1.719751135930648900*10^02, 3.317169150357214562*10^02 })
Finger 2 Joint Axis 2	({ 9.339056931870376443*10^-01, -1.402127665014160640*10^-01, -3.288776920723839869*10^-01 }, { -1.954208051888003395*10^00, -6.327599271228667988*10^01, 2.142758884488883098*10^01 })
Finger 2 Joint Axis 3	({ 8.193179577582415662*10^-01, 2.602664222429071361*10^-01, -5.108615013364585788*10^-01 }, { -3.937890770689081421*10^01, -1.350480673067052351*10^02, -1.319581204955530893*10^02 })
Finger 2 Joint Axis 4	({ -3.515980128246372072*10^-01, -7.393560921260252661*10^-01, -5.742224363553715127*10^-01 }, {2.781107140460632365*10^01, -2.439447905520300992*10^02, 2.970691125887967132*10^02 })
Finger 3 Joint Axis 1	({ -3.999837826567627375*10^-02, -8.822209785493367873*10^-01, 4.691335361531583370*10^-01 }, { 7.221212714480301997*10^01, 1.710348631669522774*10^02, 3.277934755785679499*10^02 })
Finger 3 Joint Axis 2	({ -4.358223680974385728*10^-01, 3.212427656523491604*10^-01, 8.407508245502769428*10^-01 }, { 1.613038023697479506*10^00, 2.819980771738672729*10^02, -1.069125852082746491*10^02 })
Finger 3 Joint Axis 3	({ 5.863860437746715171*10^-01, 8.096526739296313613*10^-01, -2.477812068916963525*10^-02 }, { -4.578884117588163605*10^01, 2.338940319680385116*10^01, -3.193399808444744394*10^02 })
Finger 3 Joint Axis 4	({ 5.008482440031977179*10^-01, 2.082640243379740852*10^-01, -8.401054294821959090*10^-01 }, { -4.437129333761157568*10^01, -2.802213715227478019*10^02, -9.592047862468356811*10^01 })
Finger 4 Joint Axis 1	({ -4.732043812667470206*10^-01, 1.263683531667600390*10^-01, 8.718421031745808847*10^-01 }, { 1.509625665707851994*10^01, 3.026098798936037042*10^02, -3.566780873515806149*10^01 })

Finger 4 Joint Axis 2	({ -9.995558470113082850*10^-01, -3.117072620563319726*10^-03, -2.963768823279408146*10^-02 }, { -7.787296815606066236*10^-01, -8.758620914772937738*10^01, 3.547497944711857087*10^01 })
Finger 4 Joint Axis 3	({ 4.234224680097327664*10^-01, -2.156351247419446143*10^-02, 9.056756751256610505*10^-01 }, { 3.369199010157884544*10^01, 3.706833749770769941*10^02, -6.926000325453114392*10^00 })
Finger 4 Joint Axis 4	({ -2.406044451548889906*10^-01, -9.608648918299699249*10^-01, -1.372886033882231016*10^-01 }, { 6.180974690421125928*10^01, -6.884640582312576385*10^01, 3.735225878877795367*10^02 })

5.9 Validation of the Solution

All the 4 end effector positions are given in a homogeneous matrix. The solution is first converted into dual quaternions using a Mathematica file. The input positions of the task with 13 positions files are imported and subtracted from the output positions of the end effectors from the solver. The error varies from 10^{-17} to 10^{-12}, which means that the end effectors are able to reach all the positions. In the next chapter the CAD simulation is presented in detail.

Chapter 6

6 CAD Model of Hand

6.1 Solver Output

The solver gives the output in terms of the joint axis and joint angles in order to reach all the 13 positions. In the case of planar linkage the links are usually located in parallel planes which are perpendicular to the joints, which make it easy to define their geometry. However, in case of the spatial mechanism with the general relative positions between the joints, the geometry and properties can change greatly just by sliding the joint location along the axis. Even though this operation does not change the trajectory of the linkage, it does affect the link size, shape, manufacturability, friction, self-intersection and force transmission issues.

6.2 Solver Output Axis

Optimization needs to be applied to the output of the solver in order to get a better model for both computational and from user interaction and assessment point of view. The output of the kinematic synthesis process yields a set of structural parameters. The Plucker coordinates of the joint axes are given in the first task position. Due to the high resolution and the accuracy of the placement of the joint axes, the joint axes are directly imported into solid works by using Macros. The Macros help in making the process automated and avoids human errors. An infinite line is passed through each axis to represent them in the solid work. The figure 6.2 shows the axis of 13 position hand candidate in Solid Work. Challenges are faced with placing the sliding points to a location where they can be jointed to the links. Optimization techniques have been developed by application of additional constraints to the output axes [1]. The constraints that can be applied to the links are through the optimization are offset length, minimum and maximum link length, obstacle avoidance, avoidance of intersection between any two links and force transmission.

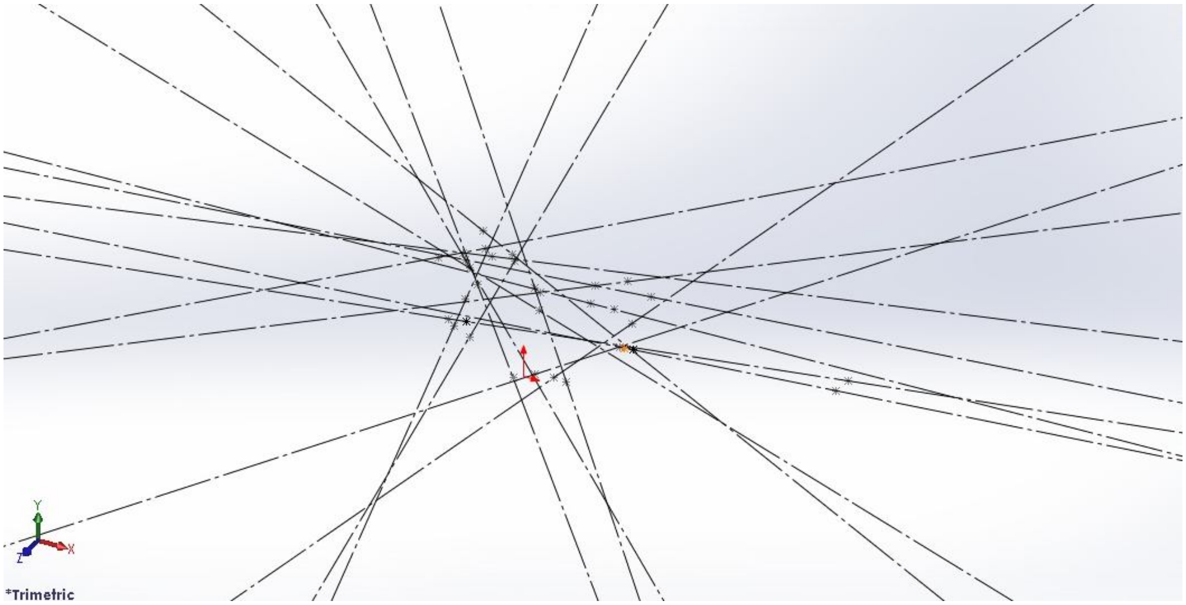


Figure 6-1Joint Axis in Solid Works

6.3 Initial Model

The initial model is obtained by placing the axes and the links without any optimization. The hand consists of joint axis, links, palms and finger tips. First the palm is constructed and then the links are attached to it.

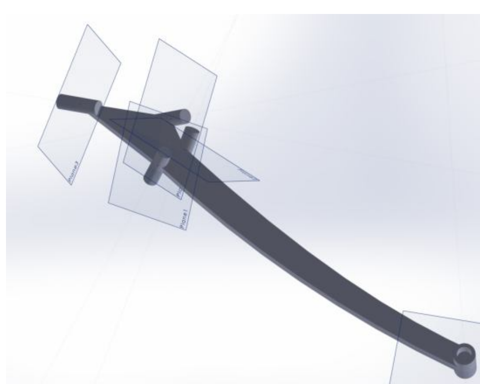


Figure 6-2 Palm will be attached to four end effectors and a wrist.

Due to the complexity of the axes, each link is constructed and attached to the palm. Links are manually rotated along their axes and adjustments are done in the link positions to avoid any possible intersection between links.

The figure 6.4 shows two links and an end effector. Due to the complexity of axes placement, the links are made using the sweep command in solid work and hence every link has a different shape. The 4 end effectors are represented by black color in the final design.

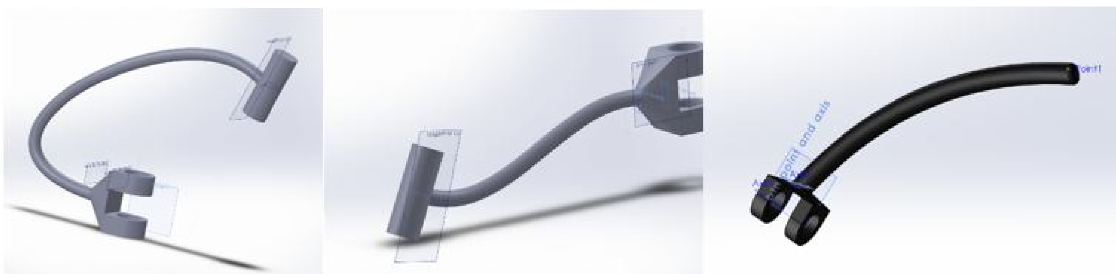


Figure 6-3 Left and Middle represent links joining two axes. The right figure represent the end effector.

Challenge is being faced due to the complex structure of the hand. The adjustment of the link placement is done carefully, so that there is no intersection between serial chains. After repeated modification, the initial design is completed.

Figure 6-4Shows the CAD model.

6.4 Final Model

In the final design the length of the links has manually been optimized. Due to the complexity of the joint axis the palm and the wrist cannot be further contracted. All of the four end effectors' serial chains were manually compacted, keeping in view the strength and manufacturability of the system.

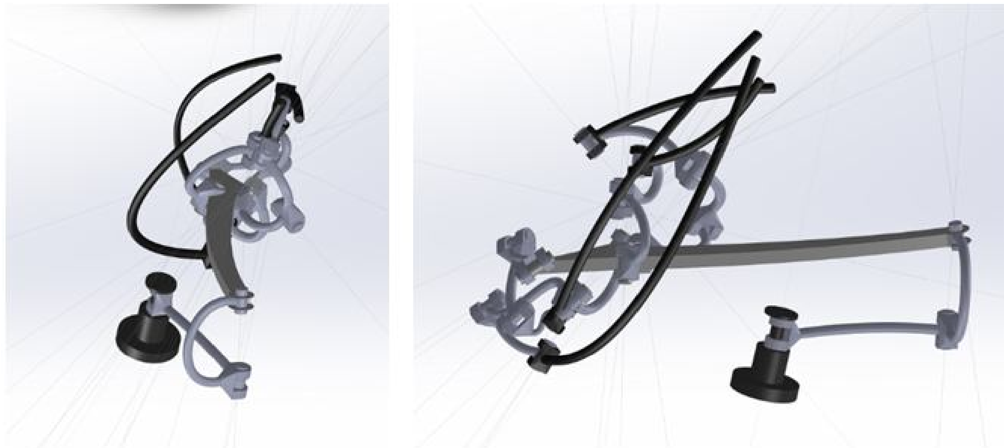


Figure 6-5Final design top and side view

Figure 6-6Final Design of 3-(3,4,4,4) Topology

Chapter 7

7 Conclusion and Future Works

In this research a comprehensive methodology for the kinematic design of non-anthropomorphic hands to solve anthropomorphic tasks has been implemented, including the input data acquisition, selection of the hand topology using tree graph theory, kinematic synthesis and CAD simulation of the hand is presented.

The methodology is based on representing the tree as a rooted graph and in order to simplify the reduction operation is performed. Due to the tree structure, the exact synthesis for general serial kinematic chain with more than five degrees of freedom can be solved. The methodology's main application is in the design of the multi fingered robotic hand, where the non-anthropomorphic hand could be used to perform human tasks. The case presented in this work is about sorting of the potato which are sorted manually in the industry. A developed CAD model can be manufactured and along with a vision system implemented in the industry.

The current methodology is able to consider only the revolute joints, in future the prismatic joints can be added into consideration. The dimensional synthesis, which is presented here has a large number of solutions. A good solution selection process is required to choose the best from the pool of the candidates. Additional constraints can be added either in the solving process of the in the optimization process to give a suitable design. A lot of processes can be made automated such as the CAD simulation process in which macros can be developed to easily modify the parts and better constraints for the optimization of the parts and link lengths to give a suitable design. Future work can focus on the possible consideration of the link collision. In order to perform a real human task in the future, not only the trajectory point, but also the velocities, acceleration and fingertip forces should be kept into consideration.

8 References

[1] Workspace synthesis for articulated systems with application to exoskeleton design by Yimesker Siraw Yihun 2013 Idaho State University.

[2] Siciliano, B., and Khatib, O.,2008. *Handbook of Robotics*. Springer, Berlin.

[3] Soto Martell, J., and Gini, G., 2007. "Robotic hands: Design review and proposal of new design process". World Academy of Science, Engineering and Technology, 26.

[4] Dollar, A., and Howe, R., 2007. "Simple, robust autonomous grasping in unstructured environments". In Proc. of the 2007 International Conference on Robotics and Automation, pp. 4693–4700.

[5] Briglen, L., Laliberte, T., and Gosselin, C., 2008. Underactuated Robotic Hands. Springer.

[6] van Varseveld, R., and Bone, G., 1999. "Design and implementation of a lightweight, large workspace non anthropomorphic dexterous hand". ASME Journal of Mechanical Design, 121.

[7] Schaffer, C., and Dillmann, R., 2001. "Kinematic design of a humanoid robot wrist". Journal of Robotic Systems, 18(12), pp. 747–754.

[8] Ramos, A., Gravagne, I., and Walker, I., 1999. "Goldfinger: A non-anthropomorphic, dextrous robot hand". In Proceedings of the 1999 IEEE International Conference on Robotics and Automation.

[9] R. Hartenberg, J. Denavit, Kinematic Synthesis of Linkages, McGrawHill, 1964.

[10] B. Roth, Finite position theory applied to mechanism synthesis, Journal of Applied Mechanics 34 (1967) 599-605.

[11] G. N. Sandor, A. G. Erdman, Advanced Mechanism Design: Analysis and Synthesis, Vol. 2, Prentice-Hall, Englewood Cli, NJ, 1984.

[12] C. Suh, C. Radcli_e, Kinematics and Mechanisms Design, John Wiley, 1978.

[13] P. Chen, B. Roth, Design equations for finitely and infinitesimally separated position synthesis of binary link and combined link chains, ASME Journal of Engineering for Industry 91 (1967) 209-219.

[14] E. Lee, C. Mavroidis, Geometric design of 3r manipulators for reaching four end-effector spatial poses, The International Journal of Robotics Research 23(3) (2004) 247-254.

[15] J. M. McCarthy, G. Soh, Geometric Design of Linkages, Springer-Verlag, New York, second edition, 2010.

[16] M. L. Husty, M. Pfurner, H.-P. Schrocker, K. Brunnthaler, Algebraic methods in mechanism analysis and synthesis, Robotica 25 (2007) 661-675.

[17] M. Huang, A. Soni, Application of linear and nonlinear graphs in structural synthesis of kinematic chains, Journal of Engineering for Industry (1973) 525-532.

[18] L. Woo, Type synthesis of plane linkages, ASME Journal of Engineering for Industry (1967) 159-172.

[19] F. Freudenstein, E. R. Maki, The creation of mechanisms according to kinematic structure and function, Journal of Environment and Planning 6 (1979) 375-391.

[20] T. Mruthyunjaya, Kinematic structure of mechanisms revisited, Mechanism and Machine Theory 38 (2003) 279-320.

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

[22] C. Chuang, J. Lee, Topological synthesis of under actuated passively adaptive finger mechanisms, in: Proc. of the 13th World Congress in Mechanism and Machine Science, Guanajuato, Mexico.

[23] N. Biggs, Discrete Mathematics, 1985.

[24] G. Song, N. Amato, A motion-planning approach to folding: from paper craft to protein folding, IEEE Transactions on Robotics and Automation 20(1) (2004) 60-71.

[25] J. Garcia de Jalon, E. Bayo, Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time challenge, Springer-Verlag, 1994.

[26] S. Stramigioli, Modeling and IPC control of interactive mechanical systems - A coordinate-free approach, volume LNCIS 266, Springer, 2001.

[27] C. Tischler, A. Samuel, K. Hunt, Kinematic chains for robot hands - 1. Orderly number synthesis, Mechanism and Machine Theory 30(8) (1995) 1193-1215.

[28] C. Tischler, A. Samuel, K. Hunt, Kinematic chains for robot hands ii: Kinematic constraints, classification, connectivity and actuation, Mechanism and Machine Theory 30(8) (1995) 1217-1239.

[29] J. Dai, D. Wang, Geometric analysis and synthesis of the metamorphic robotic hand, ASME Journal of Mechanical Design 129 (2007) 1191-1197.

[30] Jacob Rosen , Joel C. Perry , The Human Arm Kinematics and Dynamics During Daily Activities – Toward a 7 DOF Upper Limb Powered Exoskeleton, Advanced Robotics, 2005. ICAR '05. Proceedings. 12th International Conference ISBN 0-7803-9178-0.

[31] Natsuki MIYATA , Makiko KOUCHI, Modeling of Human Hand Link Structure from
Optical Motion Capture Data Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems September 28 -October 2, 2004, Sendai, Japan.

[32] Jorn Vogel, Claudio Castellini and Patrick van der Smagt, EMG-Based Tele operation and Manipulation with the DLR LWR-III, 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems September 25-30, 2011. (DOI 10.1109/IROS.2011.6094739).

[33] C. Blanes1*, M. Mellado1, C. Ortiz2 and A. Valera1, Review. Technologies for robot grippers in pick and place operations for fresh fruits and vegetables, Spanish Journal of Agricultural Research 2011 9(4), 1130-1141 DOI: http://dx.doi.org/10.5424/sjar/20110904-501-10.

[34] N.J. LUWES AND P.E. HERTZOG, FEATURE EXTRACTION TECHNIQUES FOR A POTATO QUALITY SORTING SYSTEM Journal for New Generation Sciences: Volume 5 Number 2.

[35] David Lee Herriott, Development of an Automated Planter for Sweet Potato Slips, A thesis submitted to the Graduate Faculty of Auburn University Auburn, Alabama August 3, 2013.

[36] http://www.columbiafoodmachinery.com/news.php Best Genius Optical Sorter.

[37] Dr. Alba Perez Garcia, Kinematics of Robotics, Chapter 1Motion: An introduction.

[38] J. Michael McCarthy, 21st Century Kinematics. Springer, New York, first edition, 2012.

[39] Stramigioli, S.: Modeling and IPC control of interactive mechanical systems - A coordinate-free approach, vol. LNCIS 266. Springer (2001).

[40] Tischler, C., Samuel, A., Hunt, K.: Kinematic chains for robot hands - 1. Orderly number synthesis. Mechanism and Machine Theory 30(8), 1193–1215 (1995).

[41] 1. Chen, I., Yang, G., Kang, I.: Numerical inverse kinematics for modular reconfigurable robots. Journal of Robotic Systems 16(4), 213–225 (1999).

[42] Simo-Serra, E., Moreno-Noguer, F., Perez-Gracia, A.: Design of Non-Anthropomorphic
Robotic Hands for Anthropomorphic Tasks. In: Proc. of the 2011 ASME International Design
Engineering Technical Conferences. Washington D.C., USA (2011)

[43] F. R. E. Crossley, A cotribution to grubler's theory in the number synthesis of plane mechanisms, Transactions of the ASME Journal of Engineering for Industry (1964) 1-8.

[44] Tsai, L.W.: Mechanism Design: Enumeration of Kinematic Structures According to Function.
CRC Press, Boca Raton (2001).

[45] Edgar Simo-Serra, Alba Perez-Gracia, Kinematic Synthesis Using Tree Topologies corresponding author. Phone number 001 208 2825655 Email addresses: esimo@iri.upc.edu (Edgar Simo-Serra), perealba@isu.edu (Alba Perez-Gracia) Mechanism and Machine Theory Nov, 2013.

[46] Ben Kenwright. A beginners guide to dual-quaternions: What they are, how they work, and to use them for 3D. In character Hierarchies, the 20th International Conference on Computer Graphics, Visualization and Computer Vision, WSCG 2012 Communication Proceedings,pp.1-13. Citeseer, 2012.

[47] Lee, E., and Mavroidis, C., 2002. “Solving the geometric design problem of spatial 3r robot manipulators using polynomial homotopy continuation”. ASME Journal of Mechanical Design, 124(4), pp. 652–661.

[48] Perez Gracia, A., and McCarthy, J. M., 2006. “The kinematic synthesis of spatial serial chains using Clifford algebra exponentials”. Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science, 220(7), pp. 953–968.

[49] Murray, R. M., Li, Z., and Sastry, S. S., 1994. A Mathematical Introduction to Robotic Manipulation. CRC Press, Inc., Boca Raton, FL.

[50] Edgar Simo-Serra, Alba Perez-Gracia, Design of Non-Anthropomorphic Robotic hands for Anthropomorphic Tasks. Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2011 August 29-31, 2011, WASHINGTON, DC, USA

[51] http://www.barrett.com/robot/index.htm

[52] J. Butterfass, M. Grebenstein, H. Liu, G. Hirzinger. DLR-Hand II next generation of a extrous robot hand. Proceedings of the 2001 IEEE International Conference on Robotics and Automation.

[53] Shadow Robot Company. Design of a dextrous hand for advanced CLAWAR applications. Shadow Robot Company 2003.

[54] C.S. Lovchik, M.A. Diftler. The Robonaut Hand: A dextrous robot hand for space. Proceedings of the 1999 IEEE International Conference on Robotics and Automation.

[55] D. Osswald, H. Worn. Mechanical System an Control System of a Dextrous Robot Hand. Proceeding of the 2001 IEEE-RAS, International Conference on Humanoid Robots.

[56] A. Namiki, Y. Imai, M. Ishikawa, M. Kaneko. Development of a Highspeed Multifingered Hand System and its applications to catching. Proceeding of the IEEE-RSJ, International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, Octubre 2003.

[57] Ramiro Cabas and Carlos Balaguer Design and development of a light weight embodied robotic hand activated with only one actuator. Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference.

[58] Kinematic Synthesis of a Human Hand Model by Karthikeyan Duraisamy 2006 Idaho State University.

9 Appendix

9.1 Solution of the Solver

```
 require "synthesis"

function saved_syn ()
   -- Create new synthesis object
   local s = syn.new( 13 )

   local ko1 = kin_object.new( "chain" )
   -- Joint 0
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 5.745478805470026984*10^00, -5.099085844532194045*10^-01, -
3.543552404305919667*10^-01, 6.653379055604888848*10^00, 1.901995444181898975*10^-01,
1.122795160966440342*10^01, 7.103953252609112923*10^00, 2.899105279589590634*10^00,
2.465546304141887468*10^00, 2.475987634332555309*10^00, 2.727480988701502351*10^00,
6.992307901257458491*10^-01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 7.157135813591766205*10^-01, -1.575513640601141629*10^-01,
6.803907973663511077*10^-01 }, { -1.208927104713507816*10^01, 2.646983236127513237*10^02,
7.401046225448509119*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -8.490568481579526150*10^00, -1.049055215279890518*10^01, -
1.255099905444878594*10^01 }, { 1.150943151842047385*10^01, 9.509447847201094817*10^00,
7.449000945551214059*10^00 } )
   ko1:attach( j )
   -- Joint 1
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 5.046330116405941979*10^00, -5.605707382806268235*10^00, -
5.725318478744084949*10^00, 1.356236762117249750*10^00, 9.905182789729422010*10^-01,
3.374145044141176886*10^00, 9.618491879681373646*10^00, 1.582089951648328885*10^00, -
5.423219867362581326*10^00, 7.196102158262219994*10^00, -3.593722271840094074*10^00, -
1.746820481399927516*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -9.808627099816861206*10^-01, 1.675939542232079416*10^-01, -
9.909899431988179741*10^-02 }, { -9.354342835341540763*10^00, -
9.299756613302859876*10^01, -6.468787924431606484*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -8.991202011170834396*10^00, -1.381123315534200557*10^01, -
7.221765376591331886*10^00 }, { 1.100879798882916560*10^01, 6.188766844657994426*10^00,
1.277823462340866811*10^01 } )
```

```
   ko1:attach( j )
   -- Joint 2
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 5.583838454070531476*10^00, 4.780557489149640737*10^00,
4.027206883143987781*10^00, 1.640656898827947519*10^00, 9.450670952195650010*10^-01,
7.978685418704198895*10^00, 1.117366378469546673*10^00, 2.505802679699343116*10^00,
3.012959079294954545*10^00, 3.341781191843997245*10^00, -2.164689828767958346*10^00,
3.302564316335587336*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                         { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 7.252855382690187591*10^-01, -2.927661188456781582*10^-01, -
6.230962105757488212*10^-01 }, { 1.302090449905773895*10^01, -1.795349296594465898*10^02,
9.951210941354081285*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                        { -1.117008946438236272*10^01, -1.126365625084558353*10^01, -
6.574544238198003931*10^00 }, { 8.829910535617637279*10^00, 8.736343749154416471*10^00,
1.342545576180199518*10^01 } )
   ko1:attach( j )
   local ko2 = kin_object.new( "splitter" )
   local sp1 = kin_object.new( "chain" )
   -- Joint 0
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 6.680540858680529936*10^00, -5.878521512303617413*10^00, -
6.175074989382607349*10^00, 3.945393142129349773*10^-01, -1.487028439853322892*10^-01,
2.420847299675171205*10^-01, -1.245698567239292487*10^01, 5.353006764076679858*10^00, -
2.723382229130645982*10^00, 2.737597477503005639*10^00, -8.133471561123991833*10^00,
1.333375074216652045*10^01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                         { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 2.991778491846188714*10^-01, 7.208914632864753536*10^-01, -
6.251464730108852175*10^-01 }, { -7.527989944831786318*10^01, -
2.178641780706298903*10^02, -2.872582863115615623*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                        { -6.199205051838911018*10^00, -1.102348275147422285*10^01, -
1.790544389759164545*10^01 }, { 1.380079494816108898*10^01, 8.976517248525777148*10^00,
2.094556102408353659*10^00 } )
   sp1:attach( j )
   -- Joint 1
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 2.687176064879369664*10^-01, 8.419912955914380603*10^00,
1.219442342989938144*10^00, 6.328811125201961651*10^00, 5.740308249375465799*10^00,
8.568911312645836276*10^00, -2.363937965936733665*10^00, -5.169820879933477720*10^00, -
6.086951499958346279*10^-01, 5.061292028582228042*10^00, 5.979994701122524070*10^00, -
7.001717586323856324*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                         { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
```

```
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -8.098108731880641065*10^-01, -4.232658046337826052*10^-01,
4.062664252606926762*10^-01 }, { 3.939534111227553836*10^01, 1.008955573484752790*10^02,
1.836440577211744483*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -9.527103660618427838*10^00, -9.677886833782155307*10^00, -
1.056451220431291560*10^01 }, { 1.047289633938157216*10^01, 1.032211316621784469*10^01,
9.435487795687084400*10^00 } )
   sp1:attach( j )
   -- Joint 2
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 6.295819979872327110*10^00, 9.071047701827618326*10^00,
2.968022314863567157*10^00, 1.035883814580943474*10^00, 6.437061561092051187*10^-01,
8.004371647092444420*10^00, 8.088829283417849902*10^-01, 7.961134005070499065*10^00,
8.256428379590726507*10^00, 7.881251266609655914*10^00, 3.146179295721165836*10^00,
5.889212073513281354*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -3.715593471713062312*10^-01, 4.562644560060451493*10^-01,
8.085582215988779131*10^-01 }, { -5.023365906115425439*10^00, 2.798259326672847465*10^02,
-1.602124646575747988*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -1.267465220013781746*10^01, -6.391167990449186931*10^00, -
7.450926720439166928*10^00 }, { 7.325347799862181652*10^00, 1.360883200955081307*10^01,
1.254907327956083307*10^01 } )
   sp1:attach( j )
   local sp2 = kin_object.new( "tcp" )
   sp2:setFK( {
       -- Frame 0
       { { 1.960587083138197462*10^-03, 9.162577011903609847*10^-01, -
4.005845492622561177*10^-01, -4.536312206933137077*10^02 },
         { 5.638054998306087917*10^-01, 3.298333361295876509*10^-01,
7.571877764058159155*10^-01, 2.569865729445355385*10^01 },
         { 8.259052696641708824*10^-01, -2.273363045951568018*10^-01, -
5.159444642148719451*10^-01, 5.817158364568079776*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 1
       { { -7.251720756445072247*10^-02, 8.755055966467171213*10^-01, -
4.777354967420048837*10^-01, -4.502532678694315678*10^02 },
         { 4.677628826294981668*10^-01, 4.529044250605243072*10^-01,
7.589963553237513150*10^-01, 1.508107328612590692*10^01 },
         { 8.808740774033463339*10^-01, -1.684266368508013367*10^-01, -
4.423723858446994006*10^-01, 5.096907494736984745*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 2
       { { -1.352545380659739283*10^-01, 9.213351271422390854*10^-01, -
3.644829123928234327*10^-01, -4.484842490482496942*10^02 },
         { 3.798341830549240017*10^-01, 3.879735828843249301*10^-01,
8.397633549797816066*10^-01, 2.019740374302818253*10^01 },
         { 9.151132188508472876*10^-01, -2.486126460368465185*10^-02, -
4.024297630726970665*10^-01, 2.956944706995417960*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 3
```

```
        { { -1.402126149131615040*10^-01, 8.821546939236339924*10^-01, -
4.496037350909274921*10^-01, -4.465618609413003810*10^02 },
          { 5.254511905347150069*10^-01, 4.511657647839479313*10^-01,
7.213532415208000481*10^-01, 5.453180985731108876*10^00 },
          { 8.391909609766206035*10^-01, -1.351019936026657264*10^-01, -
5.267883657976144551*10^-01, 4.978663372044769631*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 4
        { { -1.751892433933240067*10^-01, 8.853185020553784312*10^-01, -
4.307201863364377714*10^-01, -4.449486468465836424*10^02 },
          { 5.599923116498146669*10^-01, 4.494289860399912073*10^-01,
6.960044521410493346*10^-01, 9.231730977086776591*10^00 },
          { 8.097637556055292674*10^-01, -1.192674994518052589*10^-01, -
5.745066785357174632*10^-01, 3.331618628782479163*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 5
        { { -9.557470723174954497*10^-02, 9.082425142844630095*10^-01, -
4.073831250601859244*10^-01, -4.440101957131225845*10^02 },
          { 5.444343557934481215*10^-01, 3.903138979424918276*10^-01,
7.424596240232274047*10^-01, 1.888537772732497899*10^01 },
          { 8.333406911757875735*10^-01, -1.508330080558492048*10^-01, -
5.317825646930196548*10^-01, 3.761426601138587955*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 6
        { { -3.278121058446619163*10^-01, 7.946953819262089214*10^-01, -
5.108800967025873829*10^-01, -4.364875470835577858*10^02 },
          { 3.864850100555831114*10^-01, 6.062468129579522413*10^-01,
6.950497383501855753*10^-01, -4.202586007862932860*10^01 },
          { 8.620722477055063893*10^-01, 3.039821908405881956*10^-02, -
5.058728970922375945*10^-01, 1.680024854801794731*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 7
        { { -4.490332036979958197*10^-01, 7.423295433324755832*10^-01, -
4.973087884529209246*10^-01, -4.262111544435011865*10^02 },
          { 2.782767858458746746*10^-01, 6.450793435282508037*10^-01,
7.116422352646376570*10^-01, -7.527070071360685688*10^01 },
          { 8.490766823061407509*10^-01, 1.811615015640982707*10^-01, -
4.962351236208878413*10^-01, -5.046054544410537801*10^00 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 8
        { { -5.011602062481741449*10^-01, 6.754093512826409018*10^-01, -
5.409811973379945460*10^-01, -4.137104784061127702*10^02 },
          { 9.949880339689515241*10^-02, 6.659846148640895525*10^-01,
7.392999383668099090*10^-01, -1.087106671846043042*10^02 },
          { 8.596152835983512652*10^-01, 3.166807555951148823*10^-01, -
4.009674091990939604*10^-01, -2.237063342684490408*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 9
        { { -5.623776105622709798*10^-01, 6.205123533389439450*10^-01, -
5.465307333462927630*10^-01, -4.053410748227997260*10^02 },
          { 2.239283445877810197*10^-01, 7.505450371334396742*10^-01,
6.217219987458542096*10^-01, -1.420732071015912368*10^02 },
          { 7.959821101183410796*10^-01, 2.272588097041113864*10^-01, -
5.610400286819325899*10^-01, -2.131272317303910313*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
        -- Frame 10
        { { -6.012559049320483906*10^-01, 6.363065685175141040*10^-01, -
4.833273090213401346*10^-01, -4.062178224922228651*10^02 },
          { 1.923229364489505155*10^-01, 7.023304655671046559*10^-01,
6.853785853467755373*10^-01, -1.384208265559440179*10^02 },
          { 7.755663897436484433*10^-01, 3.191329942167691613*10^-01, -
5.446566873749399962*10^-01, -2.728916997456998672*10^01 },
          { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
```

```
       -- Frame 11
       { { -6.691301121840937949*10^-01, 6.571379265668518821*10^-01, -
3.470369410249017905*10^-01, -4.158103714071125410*10^02 },
         { 1.890289096028273386*10^-01, 6.021282404938095301*10^-01,
7.756994607025298194*10^-01, -7.856269515450770768*10^01 },
         { 7.187022779307635778*10^-01, 4.534438526071868925*10^-01, -
5.271202028284285301*10^-01, -6.486975499864297490*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 12
       { { -5.641569611390155892*10^-01, 7.145929368401385817*10^-01, -
4.136228448920312717*10^-01, -4.264445936288518055*10^02 },
         { 1.565791527575264297*10^-01, 5.844590778670440967*10^-01,
7.961724406311362356*10^-01, -6.832994729400547840*10^01 },
         { 8.106848290921243594*10^-01, 3.844015100347429881*10^-01, -
4.416170138965241909*10^-01, -7.322966568763007444*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } }
             } )
   sp1:attach( sp2 )
   ko2:attach( sp1 )
   local sp1 = kin_object.new( "chain" )
   -- Joint 0
   local j = kin_joint.new( "revolute" )
   j:setPositions( { -3.076428380105318897*10^-02, -1.376253225196899077*10^00, -
1.391624478045797808*10^00, 4.848577406360933040*10^00, -6.632593112375878341*10^00,
1.234345554900934339*10^01, 8.265159919911244657*10^00, 1.710157915561577591*10^00,
1.924534107216127499*10^00, -3.038692867898668215*10^00, 1.407297992355766425*10^01,
1.765915819119608043*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 1.295895578377645185*10^-01, -8.724890280714954205*10^-01, -
4.711363310065036503*10^-01 }, { 4.813266342984618973*10^01, -1.719751135930648900*10^02,
3.317169150357214562*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -4.341290126024377471*10^00, -1.244838624689096562*10^01, -
1.553852952133377840*10^01 }, { 1.565870987397562253*10^01, 7.551613753109034377*10^00,
4.461470478666221595*10^00 } )
   sp1:attach( j )
   -- Joint 1
   local j = kin_joint.new( "revolute" )
   j:setPositions( { -5.795800643453759005*10^-01, -7.101451047597515664*10^-01,
3.561252955309220369*10^-01, 1.035578356760965990*10^00, 7.205852070792950315*10^00,
3.339859176177227162*10^00, -3.545098906029519714*10^00, -2.928342461022319210*10^00,
5.770719924036399995*10^00, -6.505478458136129127*10^00, 1.872948106882977443*10^-02,
3.306065544817258228*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 9.339056931870376443*10^-01, -1.402127665014160640*10^-01, -
3.288776920723839869*10^-01 }, { -1.954208051888003395*10^00, -
6.327599271228667988*10^01, 2.142758884488883098*10^01 } )
```

```
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                        { -1.591392640607642761*10^01, -6.840556211727092162*10^00, -
6.784742481932322100*10^00 }, { 4.086073593923573277*10^00, 1.315944378827290784*10^01,
1.321525751806767701*10^01 } )
   sp1:attach( j )
   -- Joint 2
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 6.758369688772449813*10^00, 4.835915800995042702*10^00,
3.987265587965649782*10^00, -1.062089313101297838*10^00, 5.359776405683122746*10^00,
6.118131310332183936*10^00, 1.222722052920168556*10^00, 5.668550812766834035*10^00, -
3.073904457807028479*10^00, 9.332021774421908944*10^00, 1.967133001905390755*10^00,
7.293483825109880136*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                         { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 8.193179577582415662*10^-01, 2.602664222429071361*10^-01, -
5.108615013364585788*10^-01 }, { -3.937890770689081421*10^01, -
1.350480673067052351*10^02, -1.319581204955530893*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                        { -1.021692482720529505*10^01, -1.011331926550848692*10^01, -
9.530342693967037704*10^00 }, { 9.783075172794704955*10^00, 9.886680734491513078*10^00,
1.046965730603296230*10^01 } )
   sp1:attach( j )
   -- Joint 3
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 1.452985531533280798*10^-02, 3.754939310920250239*10^00, -
2.362714222520498897*10^00, -3.839419732848636108*10^-02, -6.997985225765883222*10^00, -
6.910009441673605757*10^-01, 8.869306727181674788*10^00, 1.231333015857429158*10^00,
1.380427168154233941*10^00, -3.126912230749645083*10^00, 5.063870173657161189*10^-01,
4.082385822747988335*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                         { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -3.515980128246372072*10^-01, -7.393560921260252661*10^-01, -
5.742224363553715127*10^-01 }, { 2.781107140460632365*10^01, -2.439447905520300992*10^02,
2.970691125887967132*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                        { -6.956684251900922433*10^00, -1.262185686080047020*10^01, -
1.047930806137305915*10^01 }, { 1.304331574809907757*10^01, 7.378143139199529799*10^00,
9.520691938626940853*10^00 } )
   sp1:attach( j )
   local sp2 = kin_object.new( "tcp" )
   sp2:setFK( {
       -- Frame 0
       { { -1.412896512983793929*10^-02, 9.749830519014031083*10^-01, -
2.218297113765099193*10^-01, -4.481666610848691334*10^02 },
         { 4.227011162631618135*10^-01, 2.068772083801299899*10^-01,
8.823409697858992651*10^-01, 5.665897578160024750*10^01 },
         { 9.061590029648420863*10^-01, -8.130110182445338118*10^-02, -
4.150493852397484362*10^-01, 7.993540499465805738*10^01 },
```

```
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 1
       { { -1.035256278017372966*10^-01, 9.461060519130082280*10^-01, -
3.068644373690681548*10^-01, -4.484847014976296578*10^02 },
         { 3.948208793410721285*10^-01, 3.222637404879843959*10^-01,
8.603851200497565355*10^-01, 4.581531988557893698*10^01 },
         { 9.129068504642723347*10^-01, -3.208457729613473086*10^-02, -
4.069049794179590562*10^-01, 7.220173128713416588*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 2
       { { -1.705515907344699322*10^-01, 9.446755401996530521*10^-01, -
2.801790117878828479*10^-01, -4.481457302269723186*10^02 },
         { 2.983281372300973233*10^-01, 3.205048640263067350*10^-01,
8.990422429854438846*10^-01, 4.385286898609959394*10^01 },
         { 9.391019526306827458*10^-01, 6.974780200100483274*10^-02, -
3.364859085924201132*10^-01, 5.328611887759799970*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 3
       { { -1.327688229665697772*10^-01, 9.299479924461586666*10^-01, -
3.428836114389698375*10^-01, -4.459909357178103164*10^02 },
         { 2.774708048591561504*10^-01, 3.669870671194422895*10^-01,
8.878797469352938121*10^-01, 2.887145535039636712*10^01 },
         { 9.515158391213901501*10^-01, 2.274255729746965660*10^-02, -
3.067578588865941258*10^-01, 7.488399982070633598*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 4
       { { -1.130955291129311230*10^-01, 9.191548120256736176*10^-01, -
3.773113208276607233*10^-01, -4.443986231888932252*10^02 },
         { 2.984589655349009774*10^-01, 3.936375696368730193*10^-01,
8.694663361293593073*10^-01, 3.250306062622463088*10^01 },
         { 9.476980780747116251*10^-01, -1.427919116839926339*10^-02, -
3.188486435804195329*10^-01, 6.060242784995880072*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 5
       { { -3.436902375108873142*10^-02, 9.214915111976316187*10^-01, -
3.868748699477663755*10^-01, -4.404313515902176732*10^02 },
         { 3.894437183187540175*10^-01, 3.688527148508720099*10^-01,
8.439675734346689939*10^-01, 4.005754413899742872*10^01 },
         { 9.204088007339232824*10^-01, -1.216596463000177430*10^-01, -
3.715461882373367608*10^-01, 6.506509237818949032*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 6
       { { -2.509805159312512024*10^-01, 8.398843973388830930*10^-01, -
4.812514724440679070*10^-01, -4.417370386994934961*10^02 },
         { 2.601259155821583624*10^-01, 5.373964418875053495*10^-01,
8.022091823765122864*10^-01, -2.297709964784367997*10^01 },
         { 9.323858046245803166*10^-01, 7.615289458286927937*10^-02, -
3.533517340854910871*10^-01, 4.457618335289434697*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 7
       { { -3.661334226641571599*10^-01, 8.215906706220520439*10^-01, -
4.369611959373912446*10^-01, -4.362508621313282902*10^02 },
         { 1.179766669982340588*10^-01, 5.067604547530800696*10^-01,
8.539761984636562442*10^-01, -5.748448935297754758*10^01 },
         { 9.230535319537086503*10^-01, 2.611180029129684388*10^-01, -
2.824704687264237846*10^-01, 2.238127293366508752*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 8
       { { -4.551663349015802607*10^-01, 7.083736222550378603*10^-01, -
5.394723522716793340*10^-01, -4.261041307775522000*10^02 },
         { 3.687119718957793113*10^-02, 6.203472397441185393*10^-01,
7.834601565872126860*10^-01, -9.735793073560179778*10^01 },
```

```
         { 8.896426936641863570*10^-01, 3.367136965362852452*10^-01, -
3.084797629907035099*10^-01, 6.778879344990727418*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 9
       { { -4.848870300524289467*10^-01, 6.130623379877668810*10^-01, -
6.237300199829303171*10^-01, -4.199847395791510962*10^02 },
         { 8.967426981997789204*10^-02, 7.442715555699485064*10^-01,
6.618295678660341963*10^-01, -1.273158869047956046*10^02 },
         { 8.699672944537555352*10^-01, 2.649800394566744766*10^-01, -
4.158635416459954270*10^-01, 5.893317304634521747*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 10
       { { -6.050273919963272684*10^-01, 6.426668199400690229*10^-01, -
4.700225669925226502*10^-01, -4.209096391836665703*10^02 },
         { 9.076473973245724469*10^-02, 6.421515067109115904*10^-01,
7.611853942701504172*10^-01, -1.253190845740185182*10^02 },
         { 7.910142963028043983*10^-01, 4.178765379595076324*10^-01, -
4.468507380183615041*10^-01, -2.496430401783015895*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 11
       { { -6.887699853916170456*10^-01, 6.579780994819611761*10^-01, -
3.044022467488336492*10^-01, -4.315011964103144919*10^02 },
         { 1.408241033328250236*10^-01, 5.333050870404880683*10^-01,
8.341188500790776006*10^-01, -6.437713496266508173*10^01 },
         { 7.111712024148173494*10^-01, 5.316488547329368819*10^-01, -
4.599837128823356380*10^-01, -3.958973761924980295*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 12
       { { -5.525592687864038233*10^-01, 6.696215968602328150*10^-01, -
4.962712680546652044*10^-01, -4.363072007847148370*10^02 },
         { 1.640409559525609073*10^-01, 6.711519687820693969*10^-01,
7.229423210534302591*10^-01, -6.735129027934351598*10^01 },
         { 8.171712300665041617*10^-01, 3.180596670725523989*10^-01, -
4.806966079902087463*10^-01, 3.067636982645122146*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } }
           } )
   sp1:attach( sp2 )
   ko2:attach( sp1 )
   local sp1 = kin_object.new( "chain" )
   -- Joint 0
   local j = kin_joint.new( "revolute" )
   j:setPositions( { -1.996361518221262954*10^-01, 8.791406628175634452*10^00,
9.786302109631950685*10^00, 4.281022692216226666*10^00, 3.485513531161005041*10^00,
5.129359644708666011*10^00, 1.345044215835715384*10^01, 2.024834395043844548*10^00,
1.608749525972543637*10^01, -7.371525359370203923*10^00, 1.418365097872510106*10^01, -
6.898405355636993086*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -3.999837826567627375*10^-02, -8.822209785493367873*10^-01,
4.691335361531583370*10^-01 }, { 7.221212714480301997*10^01, 1.710348631669522774*10^02,
3.277934755785679499*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -1.688502259682623929*10^01, -8.303614712283277299*10^00, -
8.108298559644127934*10^00 }, { 3.114977403173762482*10^00, 1.169638528771672270*10^01,
1.189170144035587207*10^01 } )
   sp1:attach( j )
```

```
   -- Joint 1
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 4.794116961176350666*10^-01, -2.067344100594064482*10^00,
1.056744926613240132*10^01, 4.976636957012777196*10^-01, 7.603802378554017416*10^00, -
2.786951940302019537*10^00, 1.130044988857184052*10^01, 2.231242482206128308*10^00,
9.494106742444887725*10^-01, 7.002685516610510419*10^00, 4.196875083303218901*10^00,
5.648836213441452703*10^-01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -4.358223680974385728*10^-01, 3.212427656523491604*10^-01,
8.407508245502769428*10^-01 }, { 1.613038023697479506*10^00, 2.819980771738672729*10^02,
-1.069125852082746491*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -6.664385821632340168*10^00, -1.317525460569578399*10^01, -
7.826592957196185196*10^00 }, { 1.333561417836765983*10^01, 6.824745394304216006*10^00,
1.217340704280381480*10^01 } )
   sp1:attach( j )
   -- Joint 2
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 6.106192475491839744*10^00, 5.391128054850106821*10^00,
1.063355927402615286*10^01, 5.522282731529623234*10^00, 4.629316337822287686*10^00,
8.647439630488326756*10^00, 7.374005370391725656*10^00, 6.838331009416471318*10^00, -
4.405944657816009524*10^00, 3.707278462963935262*10^00, -6.676503783485590482*10^00,
5.312890410597709057*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 5.863860437746715171*10^-01, 8.096526739296313613*10^-01, -
2.477812068916963525*10^-02 }, { -4.578884117588163605*10^01, 2.338940319680385116*10^01,
-3.193399808444744394*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -7.500480143139031775*10^00, -9.011988259737696794*10^00, -
1.256243907003505988*10^01 }, { 1.249951985686096734*10^01, 1.098801174026230321*10^01,
7.437560929964940115*10^00 } )
   sp1:attach( j )
   -- Joint 3
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 6.818541167765426181*10^00, -3.074109433518056878*10^-01, -
5.156709367247378495*10^-01, 1.182962800672062542*10^01, 1.201148571604153936*10^01,
8.672933231182118341*10^-01, 8.759168363425525738*10^-01, 5.617209852425039784*10^00, -
2.005447971773827831*10^00, -2.308670825741050869*10^00, 5.499285143071144244*10^-01, -
1.033900102324440340*10^01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
```

```
   j:setPlucker( { 5.008482440031977179*10^-01, 2.082640243379740852*10^-01, -
8.401054294821959090*10^-01 }, { -4.437129333761157568*10^01, -
2.802213715227478019*10^02, -9.592047862468356811*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                      { -1.002246454402270714*10^01, -1.306226560759607480*10^01, -
4.231632106898902457*10^00 }, { 9.977535455977292855*10^00, 6.937734392403925199*10^00,
1.576836789310109666*10^01 } )
   sp1:attach( j )
   local sp2 = kin_object.new( "tcp" )
   sp2:setFK( {
      -- Frame 0
      { { -6.351825564895436926*10^-01, 6.307027212383015113*10^-01, -
4.458219345815199186*10^-01, -4.051152220333344758*10^02 },
        { 7.003455777682189298*10^-01, 7.137053661245890135*10^-01,
1.186263315903698934*10^-02, -3.163562424837625642*10^01 },
        { 3.256673020613325975*10^-01, -3.046944826995849898*10^-01, -
8.950430607409484152*10^-01, 3.975144188313363003*10^01 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 1
      { { -6.773016984386358574*10^-01, 5.313675986032923015*10^-01, -
5.088328649435979179*10^-01, -4.001623029795779871*10^02 },
        { 5.593274188312283846*10^-01, 8.212011499978024709*10^-01,
1.130553394841776887*10^-01, -3.382487173599695751*10^01 },
        { 4.779280780993465649*10^-01, -2.080315995152115516*10^-01, -
8.534094010306001454*10^-01, 2.845151493622103089*10^01 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 2
      { { -6.983596431682863592*10^-01, 5.560657662142646229*10^-01, -
4.506491677995267953*10^-01, -3.970835172959771739*10^02 },
        { 5.442093306820902088*10^-01, 8.214831986675510400*10^-01,
1.702984401146405147*10^-01, -2.881658555612166950*10^01 },
        { 4.648978524282669778*10^-01, -1.263179241100453321*10^-01, -
8.763068919369021437*10^-01, 7.699936022085623222*10^00 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 3
      { { -6.329570779948424653*10^-01, 4.056470975210151320*10^-01, -
6.594056184845610291*10^-01, -3.922414987758213556*10^02 },
        { 4.770146904011843825*10^-01, 8.751995483575136792*10^-01,
8.051543762699032991*10^-02, -4.374539917619753027*10^01 },
        { 6.097723530611179887*10^-01, -2.635833508163586969*10^-01, -
7.474633734269038321*10^-01, 4.382405654688317043*10^01 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 4
      { { -6.679763155901399241*10^-01, 3.611185892534554132*10^-01, -
6.506927126579916676*10^-01, -3.902373545496211023*10^02 },
        { 4.154725881206757787*10^-01, 9.063820915160580860*10^-01,
7.651165074211326023*10^-02, -4.020899052459636636*10^01 },
        { 6.174060012106529216*10^-01, -2.192370148368426030*10^-01, -
7.554766449034020503*10^-01, 3.328689672450741455*10^01 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 5
      { { -6.507997972278972698*10^-01, 4.424958792185360035*10^-01, -
6.169740843850275436*10^-01, -3.908567278390998467*10^02 },
        { 4.166138743424137592*10^-01, 8.874721705496489976*10^-01,
1.970432089804128872*10^-01, -3.282565663541564760*10^01 },
        { 6.347381378438921029*10^-01, -1.288042832149227102*10^-01, -
7.619166312609003677*10^-01, 3.155764852801995346*10^01 },
        { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
      -- Frame 6
      { { -8.495104001937898719*10^-01, 4.170010187008728608*10^-01, -
3.231752316700975403*10^-01, -3.702309755789268593*10^02 },
        { 4.467387332577459902*10^-01, 8.944363822983357748*10^-01, -
2.020055019869020085*10^-02, -7.700324439371955521*10^01 },
```

```
         { 2.806360350522566494*10^-01, -1.615354711000020482*10^-01, -
9.461235159357624935*10^-01, 1.438237561723714464*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 7
       { { -9.069241237919013487*10^-01, 3.973446737147259267*10^-01, -
1.400208697117302781*10^-01, -3.574639635299275824*10^02 },
         { 4.059195841264038318*10^-01, 9.131202213362464848*10^-01, -
3.795724712740751461*10^-02, -1.052553952630881327*10^02 },
         { 1.127737775679199528*10^-01, -9.126155629497968713*10^-02, -
9.894207413611658675*10^-01, -1.479355507487944976*10^-01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 8
       { { -9.220641492565737618*10^-01, 3.841841081595004370*10^-01, -
4.690709640811880493*10^-02, -3.440828345351681605*10^02 },
         { 3.786883256103102946*10^-01, 9.205610450218605356*10^-01,
9.572102399535854789*10^-02, -1.366008288854275179*10^02 },
         { 7.995534192416997588*10^-02, 7.049775475821644377*10^-02, -
9.943023734618355869*10^-01, -1.844406239584474250*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 9
       { { -9.545807612712562396*10^-01, 2.930551363747990190*10^-01, -
5.379830160086046531*10^-02, -3.352133010011987153*10^02 },
         { 2.926475460075880775*10^-01, 9.560950547246030995*10^-01,
1.548096078720347796*10^-02, -1.633749294689238525*10^02 },
         { 5.597306518787222018*10^-02, -9.661136094087528423*10^-04, -
9.984318116916989228*10^-01, -1.215329645842677664*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 10
       { { -9.731547564633885594*10^-01, 2.280821159468449100*10^-01, -
3.079558990980093419*10^-02, -3.313196763865407206*10^02 },
         { 2.248322406077757907*10^-01, 9.707153026341812652*10^-01,
8.463016492431421622*10^-02, -1.653267410931745189*10^02 },
         { 4.919637746795860883*10^-02, 7.543440605611867467*10^-02, -
9.959364271011430514*10^-01, -8.436437610916055263*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 11
       { { -9.655471384686270531*10^-01, 2.513708870178261723*10^-01,
6.731567837374254071*10^-02, -3.452015686980592477*10^02 },
         { 2.578705255665851848*10^-01, 9.589750228014982669*10^-01,
1.177696806774947286*10^-01, -1.158023914173495399*10^02 },
         { -3.495018510765007802*10^-02, 1.310709075376272070*10^-01, -
9.907567318762985353*10^-01, -4.197446149355384648*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 12
       { { -9.782006167231181992*10^-01, 1.086970988649428316*10^-01,
1.769420643624807499*10^-01, -3.384388387789482522*10^02 },
         { 1.930428164300226901*10^-01, 7.900211148914916270*10^-01,
5.818944140051263147*10^-01, -8.406058726727459884*10^01 },
         { -7.653773231077590222*10^-02, 6.033668690970275250*10^-01, -
7.937823359138035384*10^-01, -2.852880711280004888*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } }
             } )
   sp1:attach( sp2 )
   ko2:attach( sp1 )
   local sp1 = kin_object.new( "chain" )
   -- Joint 0
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 1.524187793600215945*10^00, 7.859692655110619874*10^00,
1.438230368542438908*10^00, 5.268887097078213122*10^-01, 1.601803191350011701*10^00,
2.754922808891668407*10^00, 4.951366040769046073*10^00, 5.080011357062993227*10^00,
4.181417475014528873*10^-01, 2.137020744610398371*10^00, -8.689111900337657957*10^-02,
1.888606195963908607*10^-01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
```

```
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -4.732043812667470206*10^-01, 1.263683531667600390*10^-01,
8.718421031745808847*10^-01 }, { 1.509625665707851994*10^01, 3.026098798936037042*10^02,
-3.566780873515806149*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -7.657484901363642571*10^00, -1.129441117946721107*10^01, -
1.097965666594999945*10^01 }, { 1.234251509863635832*10^01, 8.705588820532788930*10^00,
9.020343334050000550*10^00 } )
   sp1:attach( j )
   -- Joint 1
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 1.375923443891604814*10^00, 6.419547092046407499*10^00,
5.914597086528771142*10^-01, 6.292685118035204361*10^00, 1.820218614141176472*10^00,
1.345306103088056426*10^-01, 1.204219755810755288*10^01, 6.564954924136075398*10^00,
3.009966245036853838*10^00, 2.277391350468262576*10^00, -2.298180348939328432*10^00,
8.076071035279499544*10^-01  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -9.995558470113082850*10^-01, -3.117072620563319726*10^-03, -
2.963768823279408146*10^-02 }, { -7.787296815606066236*10^-01, -
8.758620914772937738*10^01, 3.547497944711857087*10^01 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -1.163684041036047034*10^01, -1.741525754199729192*10^01, -
7.002860699888097784*10^00 }, { 8.363159589639529656*10^00, 2.584742458002709853*10^00,
1.299713930011190222*10^01 } )
   sp1:attach( j )
   -- Joint 2
   local j = kin_joint.new( "revolute" )
   j:setPositions( { -1.831822923845571882*10^00, 1.298077142549996399*10^00,
7.949457195764706441*10^00, 3.232323143210258376*10^-01, -8.709342868232235801*10^00,
6.484802899264421328*10^00, -1.924219712532043158*10^-01, 4.343413520337799483*10^00,
8.140671561605457296*10^-02, 7.274748885804673826*10^00, 6.664426707272481210*10^00, -
1.669208046624347297*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { 4.234224680097327664*10^-01, -2.156351247419446143*10^-02,
9.056756751256610505*10^-01 }, { 3.369199010157884544*10^01, 3.706833749770769941*10^02,
-6.926000325453114392*10^00 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -6.896681125713225669*10^00, -6.698357404882660404*10^00, -
1.881434286748511298*10^01 }, { 1.310331887428677433*10^01, 1.330164259511733960*10^01,
1.185657132514887024*10^00 } )
```

```
   sp1:attach( j )
   -- Joint 3
   local j = kin_joint.new( "revolute" )
   j:setPositions( { 3.749120213161623472*10^00, 3.833338107003912310*10^00,
4.746774941671798587*10^00, 5.437977442423612473*10^00, 8.971598970291029929*10^00,
4.237565580485382455*10^00, 4.506131542862945594*10^00, 9.158104785118222679*10^00,
6.932961874755911857*10^00, -6.210994975261526996*10^00, 7.011165649040895786*10^-01,
3.945965262834181253*10^00  } )
   j:setPositionBounds( { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00  },
                        { 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00, 6.283185307179586232*10^00, 6.283185307179586232*10^00,
6.283185307179586232*10^00  } )
   j:setPlucker( { -2.406044451548889906*10^-01, -9.608648918299699249*10^-01, -
1.372886033882231016*10^-01 }, { 6.180974690421125928*10^01, -6.884640582312576385*10^01,
3.735225878877795367*10^02 } )
   j:setPluckerBounds( { -1.000000000000000000*10^00, -1.000000000000000000*10^00, -
1.000000000000000000*10^00 }, { 1.000000000000000000*10^00, 1.000000000000000000*10^00,
1.000000000000000000*10^00 },
                       { -9.524277591039204793*10^00, -1.158077085401276030*10^01, -
7.675140172494174351*10^00 }, { 1.047572240896079521*10^01, 8.419229145987239704*10^00,
1.232485982750582565*10^01 } )
   sp1:attach( j )
   local sp2 = kin_object.new( "tcp" )
   sp2:setFK( {
       -- Frame 0
       { { -4.815293714264609060*10^-01, 8.347920116570513027*10^-01, -
2.669302562978042026*10^-01, -4.344368929065384464*10^02 },
         { 7.794104494564202401*10^-01, 5.471647210676582329*10^-01,
3.051722780612503017*10^-01, -1.021605033694313036*10^01 },
         { 4.008101991364225736*10^-01, -6.109881580297399362*10^-02, -
9.141215012106957971*10^-01, 3.563604210958478546*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 1
       { { -5.341397431473045021*10^-01, 8.069523185321334502*10^-01, -
2.520370814109467172*10^-01, -4.287531167433492669*10^02 },
         { 7.201554824125178111*10^-01, 5.904716286663461711*10^-01,
3.643066522743012570*10^-01, -1.501604533722658275*10^01 },
         { 4.427988436544607609*10^-01, 1.308477572330624117*10^-02, -
8.965255003080198071*10^-01, 2.498149810828755335*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 2
       { { -5.421477896140467490*10^-01, 8.127889960110763479*10^-01, -
2.131896390069450142*10^-01, -4.255115233244308115*10^02 },
         { 6.679568265537179084*10^-01, 5.707931735170895804*10^-01,
4.775236443638972705*10^-01, -9.120559607438615046*10^00 },
         { 5.098131540838187759*10^-01, 1.164869135551192258*10^-01, -
8.523622157824174428*10^-01, 4.374289607486943510*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 3
       { { -5.640795784960674908*10^-01, 7.813175796344550283*10^-01, -
2.671274393952394188*10^-01, -4.237169195439265081*10^02 },
         { 7.297694917237106393*10^-01, 6.230834514261075485*10^-01,
2.814311665545375929*10^-01, -2.622479774403342034*10^01 },
         { 3.863298047950963765*10^-01, -3.619188186718004518*10^-02, -
9.216503836129491534*10^-01, 3.218370354854476290*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 4
       { { -5.749623093872939794*10^-01, 6.791224141076823706*10^-01, -
4.563015334628899300*10^-01, -4.207968272844204876*10^02 },
         { 6.799021718375632961*10^-01, 7.068252954180105352*10^-01,
1.952717042682028392*10^-01, -2.689128525376548495*10^01 },
```

```
         { 4.551388573991412834*10^-01, -1.979665335701906170*10^-01, -
8.681347084822772509*10^-01, 2.831124618108758284*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 5
       { { -5.340354291374557194*10^-01, 6.534721723857083164*10^-01, -
5.364515638372938655*10^-01, -4.216627813292556652*10^02 },
         { 6.629132153551423601*10^-01, 7.174523156121306311*10^-01,
2.140286049347142894*10^-01, -1.828904916321526031*10^01 },
         { 5.247401541081849752*10^-01, -2.413219731816735047*10^-01, -
8.163402941949045033*10^-01, 2.914260792366090413*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 6
       { { -7.467884714886121600*10^-01, 4.366495184245732242*10^-01, -
5.016414824466699507*10^-01, -4.037791349107225187*10^02 },
         { 4.037317668181263919*10^-01, 8.970379337457189406*10^-01,
1.797876688850688631*10^-01, -7.128126660113895241*10^01 },
         { 5.284956379324414755*10^-01, -6.826524357827684630*10^-02, -
8.461868689631045770*10^-01, 8.886840943556947536*10^00 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 7
       { { -8.373986434183787742*10^-01, 4.505531255319930040*10^-01, -
3.094598408104234921*10^-01, -3.903361886594749421*10^02 },
         { 4.175454002452239077*10^-01, 8.926556707366557486*10^-01,
1.697695268172377814*10^-01, -1.020690563831509365*10^02 },
         { 3.527312726922809394*10^-01, 1.295123835952342872*10^-02, -
9.356350328465903310*10^-01, -1.008658395324969348*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 8
       { { -8.674624351905949604*10^-01, 4.417629979137753238*10^-01, -
2.288107890975391578*10^-01, -3.765142106015270542*10^02 },
         { 3.650412600988063860*10^-01, 8.776647195354634334*10^-01,
3.105632278751203623*10^-01, -1.319869466334877757*10^02 },
         { 3.380144996278715563*10^-01, 1.858765551568354601*10^-01, -
9.226028963125736393*10^-01, -2.989016149839460468*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 9
       { { -9.098798146278467280*10^-01, 3.551772728756942832*10^-01, -
2.144010908679824523*10^-01, -3.669332940983762796*10^02 },
         { 3.162542464316378532*10^-01, 9.282780197955345924*10^-01,
1.956608534644553865*10^-01, -1.628937123757886241*10^02 },
         { 2.685181084149691477*10^-01, 1.102226056535900028*10^-01, -
9.569477533576121742*10^-01, -2.429194632873783632*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 10
       { { -9.008723726238404561*10^-01, 4.221050821869445357*10^-01, -
1.012732335567780445*10^-01, -3.675822714096008781*10^02 },
         { 3.453081536043548505*10^-01, 8.382309400727449011*10^-01,
4.220558851137063971*10^-01, -1.573481361709500277*10^02 },
         { 2.630422918419093037*10^-01, 3.452480133132059814*10^-01, -
9.008954223470339029*10^-01, -3.327943074013290214*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 11
       { { -8.861076844289570431*10^-01, 4.541485209135525114*10^-01,
9.253265665691057240*10^-02, -3.762452574880841780*10^02 },
         { 4.438728774928589749*10^-01, 7.740837079247800956*10^-01,
4.514103252603273164*10^-01, -1.034747649636097435*10^02 },
         { 1.333793095729712830*10^-01, 4.410708946161123745*10^-01, -
8.875057327704317656*10^-01, -6.904088766058983140*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } },
       -- Frame 12
       { { -8.832619388762996682*10^-01, 4.680778740178110908*10^-01, -
2.741261000793374092*10^-02, -3.879047619019104900*10^02 },
```

```
         { 3.487592525162951373*10^-01, 6.949351208602262409*10^-01,
6.288339698038415415*10^-01, -8.151334559880817210*10^01 },
         { 3.133932531449200587*10^-01, 5.458647100243335659*10^-01, -
7.770562317061070479*10^-01, -2.697408267563278272*10^01 },
         { 0.000000000000000000*10^00, 0.000000000000000000*10^00,
0.000000000000000000*10^00, 1.000000000000000000*10^00 } }
             } )
   sp1:attach( sp2 )
   ko2:attach( sp1 )
   ko1:attach( ko2 )
   s:addObject( ko1 )
   s:finalize()
   return s
end
```

9.2 Input of the Solver ArttreeKS

P[1]={}
P[2]={}
P[3]={}
P[4]={}
P[5]={}
P[6]={}
P[7]={}
P[8]={}
P[9]={}
P[10]={}
P[11]={}
P[12]={}
P[13]={}
P[1][1] = luadq.raw({-0.5449935126146968, -0.6789361555517063, -0.19510357029818937, 0.4516219267810892, -85.19445718241812, -54.30106466920125, 174.13190060705054, -109.21442057591622})
P[1][2] = luadq.raw({-0.5463614215264885, -0.6395419373953746, -0.3131303129329542, 0.44093617962539194, -82.11624720720779, -79.51259800608639, 176.4120326758282, -91.79749225425356})
P[1][3] = luadq.raw({-0.3695115485769071, -0.9005458044090653, 0.08129288041397963, 0.21417268085244728, -26.769184984395835, 5.734439997260832, 180.82437893700623, -90.70779547182033})
P[1][4] = luadq.raw({-0.47048576193030955, -0.8577318336322863, -0.07113921061571293, 0.19462389911731154, -26.629436259624157, -24.830017798988635, 187.37973599400965, -105.31194212630537})
P[2][1] = luadq.raw({-0.4787877278136222, -0.7013904081304638, -0.21049964172132635, 0.4842558289921181, -92.73155486632947, -55.939210753461616, 173.8530138831376, -97.13452345068822})
P[2][2] = luadq.raw({-0.49525640105107094, -0.6768852095643104, -0.30592356394032216, 0.4505089159129616, -83.59502930225375, -76.16009887396453, 179.39525764417345, -84.50749791507481})
P[2][3] = luadq.raw({-0.29787016995419413, -0.9154114169960738, 0.025938135101782663, 0.26948564457525676, -41.33525432130405, -3.605355537315994, 181.95249767399756, -75.44903310080637})
P[2][4] = luadq.raw({-0.4392932188454455, -0.8690708935584122, -0.10856174990868041, 0.19987895412663056, -31.17893175563956, -30.260891392576344, 185.50684155398892, -99.34315476658189})
P[3][1] = luadq.raw({-0.4688287646151838, -0.6938403786215882, -0.29362015943602726, 0.46105565871856913, -96.09504471321011, -68.1174483815549, 167.13938295376016, -93.78317820365574})
P[3][2] = luadq.raw({-0.4597361889800615, -0.67593312601048, -0.3583157888356801, 0.45096212831550975, -90.89603736811112, -82.64963246361489, 173.5536583913942, -78.64697279215578})

P[3][3] = luadq.raw({-0.2985230519160391, -0.9214322728465643, -0.01193265020681511, 0.2484032324479506, -45.5990007314612, -7.097500057954619, 179.59792089172657, -72.49961736511867})
P[3][4] = luadq.raw({-0.4299480354215151, -0.8610027843906127, -0.17247637542178085, 0.20993044569608413, -41.994233572815446, -38.59304280484468, 181.68176812718306, -95.0231045516079})
P[4][1] = luadq.raw({-0.46971374475765526, -0.7461037055982426, -0.19147203290686365, 0.43131974113507043, -89.83532422026123, -42.87752129112141, 173.13616964154647, -95.1434070562624})
P[4][2] = luadq.raw({-0.4608447461973845, -0.7124019608559768, -0.25617720341229794, 0.4631185663765616, -91.1100298427329, -57.81456692582873, 176.89908287257373, -81.74424399049435})
P[4][3] = luadq.raw({-0.2990571368086063, -0.9310802044903037, 0.06830368715403234, 0.19745654724849485, -31.209806220484715, 7.928117336842051, 175.54377032654244, -70.6083902607556})
P[4][4] = luadq.raw({-0.34495131317715844, -0.904894625613391, 0.006642397837341843, 0.24925927590427885, -46.33660299246011, -2.778845647916743, 186.42648772168693, -79.18162070354383})
P[5][1] = luadq.raw({-0.4979240656614663, -0.7302322729631466, -0.1669198233226053, 0.43700140153114214, -81.03746539744223, -44.146101219409275, 175.31886040622607, -99.13748267530458})
P[5][2] = luadq.raw({-0.4916687216409426, -0.6907880666442581, -0.23481555300759918, 0.47532659425660667, -84.57631039501194, -58.933075186087365, 178.2875900135079, -85.05539754626336})
P[5][3] = luadq.raw({-0.32240095969123106, -0.9094527691781448, 0.09833122733520716, 0.24348357553463257, -28.584970551257666, 8.285312650565526, 174.50728266256533, -77.37785452953338})
P[5][4] = luadq.raw({-0.3562509168401752, -0.8903956301174484, 0.028025266051247128, 0.28194944685358614, -41.29690159089877, -3.6401596730611487, 187.22418232249126, -82.28513692776373})
P[6][1] = luadq.raw({-0.5113447857167359, -0.7102237622515984, -0.20825359816593725, 0.4367369419964731, -85.56702871725327, -51.72631877801276, 170.7155353089301, -102.89807406134538})
P[6][2] = luadq.raw({-0.49201739035205244, -0.6661020805475875, -0.2710950574350626, 0.49064689514518656, -91.80784200363618, -65.87890492577071, 172.5026167607502, -86.1893228953737})
P[6][3] = luadq.raw({-0.2364554204725249, -0.9083210609468502, -0.01878160947142321, 0.34451260574210185, -52.68703716207557, -13.055873988015751, 179.0668004962453, -60.821861878288075})
P[6][4] = luadq.raw({-0.37578336493427905, -0.8757608176529265, 0.007791330605571742, 0.3029342306012028, -51.17831605993471, -6.6031860858993525, 185.61565761189047, -87.34887558888421})
P[7][1] = luadq.raw({-0.3780919128630047, -0.7810140549143229, -0.23221347710375903, 0.4394774766757258, -84.47312222673568, -63.08987403667135, 166.19830103995295, -96.97697743871825})
P[7][2] = luadq.raw({-0.37582402007679544, -0.7317323096433982, -0.30009679270122763, 0.4829762395477554, -86.91765315512251, -80.20703107102403, 168.06357751621297, -84.7256531200085})

P[7][3] = luadq.raw({-0.2248203357370262, -0.9604777845463016, 0.04730354627216543, 0.1571642979248018, -24.007832805902186, 1.0888133633262793, 170.2735639516995, -78.93784758928514})
P[7][4] = luadq.raw({-0.22492246649055225, -0.9340788609905258, -0.02984823604415215, 0.27570935479867337, -50.44852721500156, -16.851928568342544, 181.78949065669752, -78.56803336606241})
P[8][1] = luadq.raw({-0.31706567789443924, -0.8047278531974544, -0.2773620091815284, 0.41827353974682846, -80.72815805344095, -74.0492969269594, 158.50380024036784, -98.5544709153052})
P[8][2] = luadq.raw({-0.3199908558495015, -0.7340579585672481, -0.37977048963309823, 0.463183700966069, -81.90211234195633, -99.73144236108725, 156.1027735814685, -86.6467433307904})
P[8][3] = luadq.raw({-0.20577681856383728, -0.9758925512690055, 0.03310272305448921, 0.06475985674624936, -13.388962235786112, 2.5235739873346406, 163.58885934847908, -88.13542815748143})
P[8][4] = luadq.raw({-0.22670504995012603, -0.9572995021154068, -0.047717763555969406, 0.17293206359123173, -36.14350864841917, -16.995151493995728, 174.39238737900718, -93.34157561626813})
P[9][1] = luadq.raw({-0.24177602656530617, -0.8012665334193356, -0.3294723744811509, 0.43699456501678074, -81.44845440077401, -89.20173076276288, 147.71644124552657, -97.25100038598325})
P[9][2] = luadq.raw({-0.24133285538080218, -0.7720092838878951, -0.36274623780501086, 0.46279075775447237, -78.3237363406711, -100.62999359993682, 154.29894007682702, -87.7675699895837})
P[9][3] = luadq.raw({-0.19472877238828976, -0.9794038451706566, -0.04242855987680211, 0.032382565924627224, -11.705347128169208, -7.715417433100317, 154.8993367092134, -100.78638001926744})
P[9][4] = luadq.raw({-0.2106393491174643, -0.9575659312860172, -0.12960981766687163, 0.14798596895676944, -33.61690826289315, -31.01805510460718, 164.15610583728844, -104.78448505355756})
P[10][1] = luadq.raw({-0.2424812135971018, -0.8310045079422566, -0.3305378635545542, 0.37600410847345245, -70.66570326586272, -86.64779262297847, 149.4274554610604, -105.71262116635671})
P[10][2] = luadq.raw({-0.24421822626824494, -0.801567114061165, -0.41104366952358906, 0.35901353924139123, -57.11006213032631, -105.72247002433411, 155.3168691716794, -97.0686439702411})
P[10][3] = luadq.raw({0.1444248071691421, 0.9858730966864845, 0.06502862526218349, 0.05447008537232585, -5.13802589651511, 5.444261678373344, -154.50069052502175, 99.53490524469659})
P[10][4] = luadq.raw({-0.19741081894831997, -0.9467376849262328, -0.2343205384277877, 0.09904852231902646, -21.85582206856526, -46.254323031131435, 159.5814985620713, -108.14965756871474})
P[11][1] = luadq.raw({-0.2454945604671246, -0.8438369300217449, -0.29760239917967884, 0.3729671142540976, -66.66956417410535, -82.909236004025, 149.31103865065307, -112.32509678736957})
P[11][2] = luadq.raw({-0.22342327503592926, -0.820674971704024, -0.35917445178737784, 0.38414625362491256, -59.364102727995736, -99.38154254818058, 158.23590577034562, -98.89195014417234})

P[11][3] = luadq.raw({-0.11409028004564294, -0.9924472890535188, -0.040320673134459806, 0.020150180332007016, -4.191392687474674, -7.863940245855041, 154.89257235516803, -101.10929675643672})
P[11][4] = luadq.raw({-0.2011169495905609, -0.953939035662607, -0.2010882954102885, 0.09547662685426048, -17.600597472328275, -41.12325203481996, 157.91414627699285, -115.35982921572558})
P[12][1] = luadq.raw({-0.25291394294825137, -0.8364177418470884, -0.3673832021411213, 0.3185427465354091, -78.92448861068657, -80.69042845299136, 153.6288904822122, -97.35376546970976})
P[12][2] = luadq.raw({-0.243879396654489, -0.8188496176518679, -0.41697757297868965, 0.31006103784841116, -69.68296440610828, -95.11602078625535, 162.67954886474803, -87.22873824663631})
P[12][3] = luadq.raw({0.12868038953757963, 0.9893531843010391, 0.06287961472295717, 0.025841596586572292, 12.66269885115024, 6.656166451553188, -163.85473074051208, 80.81473749719824})
P[12][4] = luadq.raw({-0.23838746842939132, -0.9417665747312555, -0.23691678060753027, 0.010843093716645623, -22.292577343303275, -36.90115963460526, 164.45975166112322, -101.74908712344114})
P[13][1] = luadq.raw({-0.27064782231271733, -0.8047100491640737, -0.3667699787930603, 0.38035677423686837, -71.51629074552443, -90.20744397174482, 160.94277926451272, -86.54397007107465})
P[13][2] = luadq.raw({-0.2534686225264062, -0.8222541951183165, -0.3165085670302328, 0.39934198752618555, -63.847375289299606, -86.38333111598354, 176.96717496483018, -78.13036615230598})
P[13][3] = luadq.raw({0.07993842051728611, 0.9436636142620437, 0.3140052411655827, 0.06715311283658024, -11.100544637768563, 49.173045654437516, -157.2842727190136, 57.66868938446976})
P[13][4] = luadq.raw({-0.22296814344743363, -0.9158675247330578, -0.3206519087382634, 0.0930281547138016, -17.32667034130419, -62.97553865136564, 167.2925728222509, -84.89756087919189})

9.3 Mathematica Program for processing of Vicon Output

```
<< "D:\\Softwares\\Mathematica\\Matrix.m"
```

The data is imported from the Excel file generated by the Vicon Nexus Software after image capturing

```
Qt = Import["D:\\Softwares\\Mathematica\\Qttest13.xls"];
```

Importing Eucledian coordinated of the three markers placed on the elbow

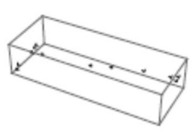
```
positions = 13;
QtelbowM1 = Table[Qt[[1, i, {1, 2, 3}]], {i, 1, positions}];
QtelbowM2 = Table[Qt[[1, i, {4, 5, 6}]], {i, 1, positions}];
QtelbowM3 = Table[Qt[[1, i, {7, 8, 9}]], {i, 1, positions}];
```

Calculating the X,Y and Z positions for the contact points

```
QtelbowM1[[1]]
Xelb = Table[(QtelbowM1[[i]] - QtelbowM2[[i]]) /
     Sqrt[(QtelbowM1[[i]] - QtelbowM2[[i]]).(QtelbowM1[[i]] - QtelbowM2[[i]])],
    {i, 1, Length[QtelbowM1]}];
Yelb = Table[(QtelbowM3[[i]] - QtelbowM2[[i]]) /
     Sqrt[(QtelbowM3[[i]] - QtelbowM2[[i]]).(QtelbowM3[[i]] - QtelbowM2[[i]])],
    {i, 1, Length[QtelbowM1]}];
Zelb = Table[Xelb[[i]] × Yelb[[i]], {i, 1, Length[QtelbowM1]}];

{-195.828, 365.59, 216.986}
```

Ploting the attached coordinate frame for Elbow

```
xplotelbow = Table[Line[{QtelbowM2[[i]], Xelb[[i]] + QtelbowM2[[i]]},
     VertexColors → {Black}], {i, 1, Length[Xelb]}];
yplotelbow = Table[Line[{QtelbowM2[[i]], Yelb[[i]] + QtelbowM2[[i]]},
     VertexColors → {Black}], {i, 1, Length[Xelb]}];
zplotelbow = Table[Line[{QtelbowM2[[i]], Zelb[[i]] + QtelbowM2[[i]]},
     VertexColors → {Black}], {i, 1, Length[Xelb]}];
plotelbow = Table[Graphics3D[{xplotelbow[[j]], yplotelbow[[j]], zplotelbow[[j]]},
     AspectRatio → Automatic], {j, 1, Length[xplotelbow]}];
Show[plotelbow]
```



Calculating the Homogenous Transformation for Elbow

```
OE = Table[Orthogonalize[{Xelb[[i]], Yelb[[i]], Zelb[[i]]}], {i, 1, Length[Xelb]}];
De = PoseElbow =
   Table[Append[Transpose[{OE[[i, 1]], OE[[i, 2]], OE[[i, 3]], QtelbowM2[[i]]}],
     {0, 0, 0, 1}], {i, 1, Length[Xelb]}];
Length[PoseElbow]
MatrixForm[PoseElbow[[1]]]
```

13

$$\begin{pmatrix} -0.956336 & 0.270073 & -0.111726 & -179.285 \\ 0.0365932 & 0.489903 & 0.871008 & 364.957 \\ 0.28997 & 0.828888 & -0.478395 & 211.97 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
Table[Det[De[[i, {1, 2, 3}, {1, 2, 3}]]], {i, 1, Length[De]}];
```

Importing Eucledian coordinated of the three markers placed on the Thumb

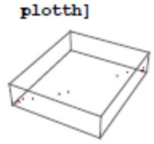
```
QtthM1 = Table[Qt[[1, i, {10, 11, 12}]], {i, 1, positions}];
QtthM2 = Table[Qt[[1, i, {13, 14, 15}]], {i, 1, positions}];
QtthM3 = Table[Qt[[1, i, {16, 17, 18}]], {i, 1, positions}];
```

Calculating the X,Y and Z positions for the contact points

```
Xth = Table[(QtthM1[[i]] - QtthM2[[i]]) /
    Sqrt[(QtthM1[[i]] - QtthM2[[i]]).(QtthM1[[i]] - QtthM2[[i]])], {i,
    1, Length[QtthM1]}];
Yth = Table[(QtthM3[[i]] - QtthM2[[i]]) / Sqrt[(QtthM3[[i]] - QtthM2[[i]]).
      (QtthM3[[i]] - QtthM2[[i]])], {i, 1, Length[QtthM1]}];
Zth = Table[Xth[[i]] * Yth[[i]], {i, 1, Length[QtthM1]}];
Length[Xth]
13
```

Ploting the attached coordinate frame for Thumb

```
xplotth = Table[Line[{QtthM2[[i]], Xth[[i]] + QtthM2[[i]]}, VertexColors → {Red}],
   {i, 1, Length[Xth]}];
yplotth = Table[Line[{QtthM2[[i]], Yth[[i]] + QtthM2[[i]]}, VertexColors → {Red}],
   {i, 1, Length[Xth]}];
zplotth = Table[Line[{QtthM2[[i]], Zth[[i]] + QtthM2[[i]]}, VertexColors → {Red}],
   {i, 1, Length[Xth]}];
plotth = Table[Graphics3D[{xplotth[[j]], yplotth[[j]], zplotth[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotth]}];
Show[
 plotth]
```



Calculating the Homogenous Transformation for Thumb

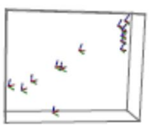
```
OT = Table[Orthogonalize[{Xth[[i]], Yth[[i]], Zth[[i]]}], {i, 1, Length[Xth]}];
Dth = Table[Append[Transpose[{OT[[i, 1]], OT[[i, 2]], OT[[i, 3]], QtthM2[[i]]}],
    {0, 0, 0, 1}], {i, 1, Length[Xth]}];
MatrixForm[Dth[[1]]];
Table[Det[Dth[[i, {1, 2, 3}, {1, 2, 3}]]], {i, 1, Length[Dth]}];
```

Calculating the Transformation from Elbow to Thumb

```
DelTh = Table[Inverse[De[[i]]].Dth[[i]], {i, 1, Length[Xth]}];
Table[HM2dq[DelTh[[i]]], {i, 1, Length[DelTh]}]

Xet = Table[DelTh[[i, {1, 2, 3}, 1]], {i, 1, Length[DelTh]}];
Yet = Table[DelTh[[i, {1, 2, 3}, 2]], {i, 1, Length[DelTh]}];
Zet = Table[DelTh[[i, {1, 2, 3}, 3]], {i, 1, Length[DelTh]}];
Pet = Table[DelTh[[i, {1, 2, 3}, 4]], {i, 1, Length[DelTh]}];
xplotth = Table[Line[{Pet[[i]], Pet[[i]] + 10 * Xet[[i]]}, VertexColors → {Blue}],
  {i, 1, Length[DelTh]}];
yplotth = Table[Line[{Pet[[i]], Pet[[i]] + 10 * Yet[[i]]}, VertexColors → {Red}],
   {i, 1, Length[DelTh]}];
zplotth = Table[Line[{Pet[[i]], Pet[[i]] + 10 * Zet[[i]]}, VertexColors → {Green}],
   {i, 1, Length[DelTh]}];
```

```
ploteth = Table[Graphics3D[{xplotth[[j]], yplotth[[j]], zplotth[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotth]}];
Show[ploteth]
```

```
{{{-0.544994, -0.678936, -0.195104, 0.451622},
  {-85.1945, -54.3011, 174.132, -109.214}},
 {{-0.478788, -0.70139, -0.2105, 0.484256}, {-92.7316, -55.9392, 173.853, -97.1345}},
 {{-0.468829, -0.69384, -0.29362, 0.461056}, {-96.095, -68.1174, 167.139, -93.7832}},
 {{-0.469714, -0.746104, -0.191472, 0.43132},
  {-89.8353, -42.8775, 173.136, -95.1434}},
 {{-0.497924, -0.730232, -0.16692, 0.437001},
  {-81.0375, -44.1461, 175.319, -99.1375}},
 {{-0.511345, -0.710224, -0.208254, 0.436737},
  {-85.567, -51.7263, 170.716, -102.898}},
 {{-0.378092, -0.781014, -0.232213, 0.439477},
  {-84.4731, -63.0899, 166.198, -96.977}},
 {{-0.317066, -0.804728, -0.277362, 0.418274},
  {-80.7282, -74.0493, 158.504, -98.5545}},
 {{-0.241776, -0.801267, -0.329472, 0.436995},
  {-81.4485, -89.2017, 147.716, -97.251}},
 {{-0.242481, -0.831005, -0.330538, 0.376004},
  {-70.6657, -86.6478, 149.427, -105.713}},
 {{-0.245495, -0.843837, -0.297602, 0.372967},
  {-66.6696, -82.9092, 149.311, -112.325}},
 {{-0.252914, -0.836418, -0.367383, 0.318543},
  {-78.9245, -80.6904, 153.629, -97.3538}},
 {{-0.270648, -0.80471, -0.36677, 0.380357}, {-71.5163, -90.2074, 160.943, -86.544}}}
```



Calculating the Relative Transformations between Thumb positions

```
(*Dthrel=Table[DelTh[[i]].Inverse[DelTh[[1]]],{i,2,Length[Xth]}];
Table[HM2dq[Dthrel[[i]]],{i,1,Length[Dthrel]}];*)
```

Importing Eucledian coordinated of the three markers placed on the Index Finger

```
Qtf1M1 = Table[Qt[[1, i, {19, 20, 21}]], {i, 1, positions}];
Qtf1M2 = Table[Qt[[1, i, {22, 23, 24}]], {i, 1, positions}];
Qtf1M3 = Table[Qt[[1, i, {25, 26, 27}]], {i, 1, positions}];
```

Calculating the X,Y and Z positions for the contact points

```
Xf1 = Table[(Qtf1M1[[i]] - Qtf1M2[[i]]) /
    Sqrt[(Qtf1M1[[i]] - Qtf1M2[[i]]).(Qtf1M1[[i]] - Qtf1M2[[i]])], {i,
    1, Length[Qtf1M1]}];
Yf1 = Table[(Qtf1M3[[i]] - Qtf1M2[[i]]) / Sqrt[(Qtf1M3[[i]] - Qtf1M2[[i]]).
      (Qtf1M3[[i]] - Qtf1M2[[i]])], {i, 1, Length[Qtf1M1]}];
Zf1 = Table[Xf1[[i]] × Yf1[[i]], {i, 1, Length[Qtf1M1]}];
```

Ploting the attached coordinate frame for Index Finger

```
xplotf1 = Table[Line[{Qtf1M2[[i]], Xf1[[i]] + Qtf1M2[[i]]},
    VertexColors → {Yellow}], {i, 1, Length[Xf1]}];
yplotf1 = Table[Line[{Qtf1M2[[i]], Yf1[[i]] + Qtf1M2[[i]]}, VertexColors → {Yellow}],
   {i, 1, Length[Xf1]}];
zplotf1 = Table[Line[{Qtf1M2[[i]], Zf1[[i]] + Qtf1M2[[i]]}, VertexColors → {Yellow}],
   {i, 1, Length[Xf1]}];
plotf1 = Table[Graphics3D[{xplotf1[[j]], yplotf1[[j]], zplotf1[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotf1]}];
Show[
  plotf1];
```

Calculating the Homogenous Transformation for Index Finger

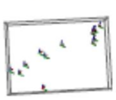
```
Of1 = Table[Orthogonalize[{Xf1[[i]], Yf1[[i]], Zf1[[i]]}], {i, 1, Length[Xf1]}];
Df1 = Posef1 =
   Table[Append[Transpose[{Of1[[i, 1]], Of1[[i, 2]], Of1[[i, 3]], Qtf1M2[[i]]}],
     {0, 0, 0, 1}], {i, 1, Length[Xf1]}];
Length[Posef1]
MatrixForm[Posef1[[1]]];
Table[Det[Df1[[i, {1, 2, 3}, {1, 2, 3}]]], {i, 1, Length[Df1]}];
MatrixForm[Posef1[[1]]];
```

```
13
```

Calculating the Relative Transformation from Elbow to Index Finger

```
Delf1 = Table[Inverse[De[[i]]].Df1[[i]], {i, 1, Length[Xf1]}];
Table[HM2dq[Delf1[[i]]], {i, 1, Length[Delf1]}]

Xef1 = Table[Delf1[[i, {1, 2, 3}, 1]], {i, 1, Length[Delf1]}];
Yef1 = Table[Delf1[[i, {1, 2, 3}, 2]], {i, 1, Length[Delf1]}];
Zef1 = Table[Delf1[[i, {1, 2, 3}, 3]], {i, 1, Length[Delf1]}];
Pef1 = Table[Delf1[[i, {1, 2, 3}, 4]], {i, 1, Length[Delf1]}];
xplotf1 = Table[Line[{Pef1[[i]], Pef1[[i]] + 10 * Xef1[[i]]}, VertexColors → {Blue}],
   {i, 1, Length[Delf1]}];
yplotf1 = Table[Line[{Pef1[[i]], Pef1[[i]] + 10 * Yef1[[i]]}, VertexColors → {Red}],
   {i, 1, Length[Delf1]}];
zplotf1 = Table[Line[{Pef1[[i]], Pef1[[i]] + 10 * Zef1[[i]]}, VertexColors → {Green}],
   {i, 1, Length[Delf1]}];

plotef1 = Table[Graphics3D[{xplotf1[[j]], yplotf1[[j]], zplotf1[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotf1]}];
Show[plotef1]
```

```
{{{-0.546361, -0.639542, -0.31313, 0.440936},
  {-82.1162, -79.5126, 176.412, -91.7975}},
 {{-0.495256, -0.676885, -0.305924, 0.450509},
  {-83.595, -76.1601, 179.395, -84.5075}},
 {{-0.459736, -0.675933, -0.358316, 0.450962},
  {-90.896, -82.6496, 173.554, -78.647}},
 {{-0.460845, -0.712402, -0.256177, 0.463119},
  {-91.11, -57.8146, 176.899, -81.7442}},
 {{-0.491669, -0.690788, -0.234816, 0.475327},
  {-84.5763, -58.9331, 178.288, -85.0554}},
 {{-0.492017, -0.666102, -0.271095, 0.490647},
  {-91.8078, -65.8789, 172.503, -86.1893}},
 {{-0.375824, -0.731732, -0.300097, 0.482976},
  {-86.9177, -80.207, 168.064, -84.7257}},
 {{-0.319991, -0.734058, -0.37977, 0.463184},
  {-81.9021, -99.7314, 156.103, -86.6467}},
 {{-0.241333, -0.772009, -0.362746, 0.462791},
  {-78.3237, -100.63, 154.299, -87.7676}},
 {{-0.244218, -0.801567, -0.411044, 0.359014},
  {-57.1101, -105.722, 155.317, -97.0686}},
 {{-0.223423, -0.820675, -0.359174, 0.384146},
  {-59.3641, -99.3815, 158.236, -98.892}},
 {{-0.243879, -0.81885, -0.416978, 0.310061}, {-69.683, -95.116, 162.68, -87.2287}},
 {{-0.253469, -0.822254, -0.316509, 0.399342},
  {-63.8474, -86.3833, 176.967, -78.1304}}}
```



Calculating the Relative Transformations between Index Finger positions

```
(*Df1rel=Table[Delf1[[i]].Inverse[Delf1[[1]]],{i,2,Length[Xf1]}];
Table[HM2dq[Df1rel[[i]]],{i,1,Length[Df1rel]}];*)
```

Importing Eucledian coordinated of the three markers placed on the Middle Finger

```
Qtf2M1 = Table[Qt[[1, i, {28, 29, 30}]], {i, 1, positions}];
Qtf2M2 = Table[Qt[[1, i, {31, 32, 33}]], {i, 1, positions}];
Qtf2M3 = Table[Qt[[1, i, {34, 35, 36}]], {i, 1, positions}];
```

Calculating the X,Y and Z positions for the contact points

```
Xf2 = Table[(Qtf2M1[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]]) /
     Sqrt[(Qtf2M1[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]]).
       (Qtf2M1[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]])], {i, 1, Length[Qtf2M1]}];
Yf2 = Table[(Qtf2M3[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]]) /
     Sqrt[(Qtf2M3[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]]).
       (Qtf2M3[[i, {1, 2, 3}]] - Qtf2M2[[i, {1, 2, 3}]])], {i, 1, Length[Qtf2M1]}];
Zf2 = Table[Xf2[[i, {1, 2, 3}]] × Yf2[[i, {1, 2, 3}]], {i, 1, Length[Qtf2M1]}];
```

Ploting the attached coordinate frame for Middle Finger

```
xplotf2 =
   Table[Line[{Qtf2M2[[i, {1, 2, 3}]], Xf2[[i, {1, 2, 3}]] + Qtf2M2[[i, {1, 2, 3}]]},
     VertexColors → {Blue}], {i, 1, Length[Xf2]}];
yplotf2 = Table[Line[{Qtf2M2[[i, {1, 2, 3}]], Yf2[[i, {1, 2, 3}]] +
       Qtf2M2[[i, {1, 2, 3}]]}, VertexColors → {Blue}], {i, 1, Length[Xf2]}];
zplotf2 = Table[Line[{Qtf2M2[[i, {1, 2, 3}]], Zf2[[i, {1, 2, 3}]] +
       Qtf2M2[[i, {1, 2, 3}]]}, VertexColors → {Blue}], {i, 1, Length[Xf2]}];
plotf2 = Table[Graphics3D[{xplotf2[[j]], yplotf2[[j]], zplotf2[[j]]},
     AspectRatio → Automatic], {j, 1, Length[xplotf2]}];
Show[
 plotf2]
```

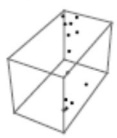
Calculating the Homogenous Transformation for Middle Finger

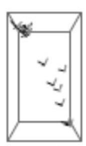
```
Of2 = Table[Orthogonalize[{Xf2[[i]], Yf2[[i]], Zf2[[i]]}], {i, 1, Length[Xf2]}];
Df2 = Posef2 =
    Table[Append[Transpose[{Of2[[i, 1]], Of2[[i, 2]], Of2[[i, 3]], Qtf2M2[[i]]}],
      {0, 0, 0, 1}], {i, 1, Length[Xf2]}];
Length[Posef2]
MatrixForm[Posef2[[1]]];
Table[Det[Df2[[i, {1, 2, 3}, {1, 2, 3}]]], {i, 1, Length[Df2]}];
MatrixForm[Posef2[[1]]];
13
```

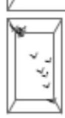
Calculating the Relative Transformation from Elbow to Middle Finger

```
Delf2 = Table[Inverse[De[[i]]].Df2[[i]], {i, 1, Length[Xf2]}];
Table[HM2dq[Delf2[[i]]], {i, 1, Length[Delf2]}]

Xef2 = Table[Delf2[[i, {1, 2, 3}, 1]], {i, 1, Length[Delf2]}];
Yef2 = Table[Delf2[[i, {1, 2, 3}, 2]], {i, 1, Length[Delf2]}];
Zef2 = Table[Delf2[[i, {1, 2, 3}, 3]], {i, 1, Length[Delf2]}];
Pef2 = Table[Delf2[[i, {1, 2, 3}, 4]], {i, 1, Length[Delf2]}];
xplotf2 = Table[Line[{Pef2[[i]], Pef2[[i]] + Xef2[[i]]}], {i, 1, Length[Delf2]}];
yplotf2 = Table[Line[{Pef2[[i]], Pef2[[i]] + Yef2[[i]]}], {i, 1, Length[Delf2]}];
zplotf2 = Table[Line[{Pef2[[i]], Pef2[[i]] + Zef2[[i]]}], {i, 1, Length[Delf2]}];

plotef2 = Table[Graphics3D[{Thick , xplotf2[[j]], yplotf2[[j]], zplotf2[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotf2]}];
Show[plotef2];
```

```
{{{-0.369512, -0.900546, 0.0812929, 0.214173},
  {-26.7692, 5.73444, 180.824, -90.7078}},
 {{-0.29787, -0.915411, 0.0259381, 0.269486},
  {-41.3353, -3.60536, 181.952, -75.449}},
 {{-0.298523, -0.921432, -0.0119327, 0.248403},
  {-45.599, -7.0975, 179.598, -72.4996}},
 {{-0.299057, -0.93108, 0.0683037, 0.197457}, {-31.2098, 7.92812, 175.544, -70.6084}},
 {{-0.322401, -0.909453, 0.0983312, 0.243484}, {-28.585, 8.28531, 174.507, -77.3779}},
 {{-0.236455, -0.908321, -0.0187816, 0.344513},
  {-52.687, -13.0559, 179.067, -60.8219}},
 {{-0.22482, -0.960478, 0.0473035, 0.157164}, {-24.0078, 1.08881, 170.274, -78.9378}},
 {{-0.205777, -0.975893, 0.0331027, 0.0647599},
  {-13.389, 2.52357, 163.589, -88.1354}},
 {{-0.194729, -0.979404, -0.0424286, 0.0323826},
  {-11.7053, -7.71542, 154.899, -100.786}},
 {{0.144425, 0.985873, 0.0650286, 0.0544701}, {-5.13803, 5.44426, -154.501, 99.5349}},
 {{-0.11409, -0.992447, -0.0403207, 0.0201502},
  {-4.19139, -7.86394, 154.893, -101.109}},
 {{0.12868, 0.989353, 0.0628796, 0.0258416}, {12.6627, 6.65617, -163.855, 80.8147}},
 {{0.0799384, 0.943664, 0.314005, 0.0671531}, {-11.1005, 49.173, -157.284, 57.6687}}}
```







Calculating the Relative Transformations between Middle Finger positions

```
(*Df2rel=Table[Delf2[[i]].Inverse[Delf2[[1]]],{i,2,Length[Xf2]}];
Table[HM2dq[Df2rel[[i]]],{i,1,Length[Df2rel]}];*)
```

Importing Eucledian coordinated of the three markers placed on the Baby Finger

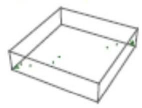
```
Qtf3M1 = Table[Qt[[1, i, {37, 38, 39}]], {i, 1, positions}];
Qtf3M2 = Table[Qt[[1, i, {40, 41, 42}]], {i, 1, positions}];
Qtf3M3 = Table[Qt[[1, i, {43, 44, 45}]], {i, 1, positions}];
```

Calculating the X,Y and Z positions for the contact points

```
Xf3 = Table[(Qtf3M1[[i]] - Qtf3M2[[i]]) /
    Sqrt[(Qtf3M1[[i]] - Qtf3M2[[i]]).(Qtf3M1[[i]] - Qtf3M2[[i]])], {i,
    1, Length[Qtf3M1]}];
Yf3 = Table[(Qtf3M3[[i]] - Qtf3M2[[i]]) / Sqrt[(Qtf3M3[[i]] - Qtf3M2[[i]]).
      (Qtf3M3[[i]] - Qtf3M2[[i]])], {i, 1, Length[Qtf3M1]}];
Zf3 = Table[Xf3[[i]] *Yf3[[i]], {i, 1, Length[Qtf3M1]}];
```

Ploting the attached coordinate frame for Baby Finger

```
xplotf3 = Table[Line[{Qtf3M2[[i]], Xf3[[i]] + Qtf3M2[[i]]}, VertexColors → {Green}],
   {i, 1, Length[Xf3]}];
yplotf3 = Table[Line[{Qtf3M2[[i]], Yf3[[i]] + Qtf3M2[[i]]}, VertexColors → {Green}],
   {i, 1, Length[Xf3]}];
zplotf3 = Table[Line[{Qtf3M2[[i]], Zf3[[i]] + Qtf3M2[[i]]}, VertexColors → {Green}],
   {i, 1, Length[Xf3]}];
plotf3 = Table[Graphics3D[{xplotf3[[j]], yplotf3[[j]], zplotf3[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotf3]}];
Show[
 plotf3]
```



Calculating the Homogenous Transformation for Baby Finger

```
Of3 = Table[Orthogonalize[{Xf3[[i]], Yf3[[i]], Zf3[[i]]}], {i, 1, Length[Xf3]}];
Df3 = Posef3 =
   Table[Append[Transpose[{Of3[[i, 1]], Of3[[i, 2]], Of3[[i, 3]], Qtf3M2[[i]]}],
     {0, 0, 0, 1}], {i, 1, Length[Xf3]}];
Length[Posef3]
MatrixForm[Posef3[[1]]];
Table[Det[Df3[[i, {1, 2, 3}, {1, 2, 3}]]], {i, 1, Length[Df3]}];
MatrixForm[Posef3[[1]]];
13
```

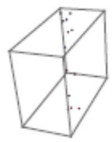
Calculating the Transformation from Elbow to Baby Finger

```
Delf3 = Table[Inverse[De[[i]]].Df3[[i]], {i, 1, Length[Xf3]}];
Table[HM2dq[Delf3[[i]]], {i, 1, Length[Delf3]}]
{{{-0.470486, -0.857732, -0.0711392, 0.194624},
  {-26.6294, -24.83, 187.38, -105.312}},
 {{-0.439293, -0.869071, -0.108562, 0.199879},
  {-31.1789, -30.2609, 185.507, -99.3432}},
 {{-0.429948, -0.861003, -0.172476, 0.20993},
  {-41.9942, -38.593, 181.682, -95.0231}},
 {{-0.344951, -0.904895, 0.0066424, 0.249259},
  {-46.3366, -2.77885, 186.426, -79.1816}},
 {{-0.356251, -0.890396, 0.0280253, 0.281949},
  {-41.2969, -3.64016, 187.224, -82.2851}},
 {{-0.375783, -0.875761, 0.00779133, 0.302934},
  {-51.1783, -6.60319, 185.616, -87.3489}},
 {{-0.224922, -0.934079, -0.0298482, 0.275709},
  {-50.4485, -16.8519, 181.789, -78.568}},
 {{-0.226705, -0.9573, -0.0477178, 0.172932},
  {-36.1435, -16.9952, 174.392, -93.3416}},
 {{-0.210639, -0.957566, -0.12961, 0.147986},
  {-33.6169, -31.0181, 164.156, -104.784}},
 {{-0.197411, -0.946738, -0.234321, 0.0990485},
  {-21.8558, -46.2543, 159.581, -108.15}},
 {{-0.201117, -0.953939, -0.201088, 0.0954766},
  {-17.6006, -41.1233, 157.914, -115.36}},
 {{-0.238387, -0.941767, -0.236917, 0.0108431},
  {-22.2926, -36.9012, 164.46, -101.749}},
 {{-0.222968, -0.915868, -0.320652, 0.0930282},
  {-17.3267, -62.9755, 167.293, -84.8976}}}
```

Calculating the Relative Transformations between Baby Finger Positions

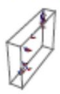
```
(*Df3rel=Table[Delf3[[i]].Inverse[Delf3[[1]]],{i,2,Length[Xf3]}];
Table[HM2dq[Df3rel[[i]]],{i,1,Length[Df3rel]}];*)
```

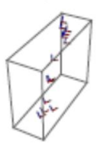
```
Xef3 = Table[Delf3[[i, {1, 2, 3}, 1]], {i, 1, Length[Delf3]}];
Yef3 = Table[Delf3[[i, {1, 2, 3}, 2]], {i, 1, Length[Delf3]}];
Zef3 = Table[Delf3[[i, {1, 2, 3}, 3]], {i, 1, Length[Delf3]}];
Pef3 = Table[Delf3[[i, {1, 2, 3}, 4]], {i, 1, Length[Delf3]}];
xplotf3 = Table[Line[{Pef3[[i]], Pef3[[i]] + Xef3[[i]]}, VertexColors → {Blue}],
   {i, 1, Length[Delf3]}];
yplotf3 = Table[Line[{Pef3[[i]], Pef3[[i]] + Yef3[[i]]}, VertexColors → {Red}],
   {i, 1, Length[Delf3]}];
zplotf3 = Table[Line[{Pef3[[i]], Pef3[[i]] + Zef3[[i]]}, VertexColors → {Green}],
   {i, 1, Length[Delf3]}];

plotef3 = Table[Graphics3D[{xplotf3[[j]], yplotf3[[j]], zplotf3[[j]]},
    AspectRatio → Automatic], {j, 1, Length[xplotf3]}];
Show[plotef3]
```

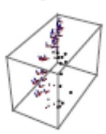


Plotting Elbow(Black), Thumb(Red), Index Finger(Yellow), Middle Finger(Blue) and Baby Finger(-Green).

```
Show[plotef1]
Show[plotef2]
Show[plotef3]
Show[ploteth]
Show[plotef1, plotef2, plotef3, ploteth]
```







9.4 Mathematica Validation File of the Solver Solution

Importing the MatrixNew file

```
<< "D:\\Thesis\\Results\\MatrixNew.m"
```

Reading lua file

```
ReadAllPoses[fileName_, nee_] := Module[{i, str, myP, myP2, myP3, myP4, myP5, myPnumber},
  str = OpenRead[fileName];
  For [i = 1, i ≤ nee, i++,
   Find[str, "sp2:setFK( {"];
   myP =
    ToString[Table[Read[str, String], {i, 1, 65}][[{2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15,
        17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 30, 32, 33, 34, 35, 37, 38, 39, 40, 42, 43,
        44, 45, 47, 48, 49, 50, 52, 53, 54, 55, 57, 58, 59, 60, 62, 63, 64, 65}]]];
   myP2 = StringToStream[myP];
   myP3 = ToString[
     Table[{Read[myP2, Record, RecordSeparators → {{"{ { "}, {"} }"}}]}, {i, 1, 13}]];
   Close[myP2];
   myP4 = StringToStream[myP3];
   myP5 =
    ToString[Table[{Read[myP4, Record, RecordSeparators → {{"{"}, {"}"}}]}, {i, 1, 52}]];
   myPnumber[i] = ToExpression[myP5];
  ];
  Close[str];
  myPnumber
 ]
```

Giving the Arttreeks file address with the number of the end effectors

```
qhat = ReadAllPoses["D:\\Thesis\\Results\\potato13OutNew.lua", 4];
qhat[3][[1]]
{-0.635182556489543693, 0.6307027212383015113,
 -0.4458219345815199186, -405.115222033334476}
```

Taking the positions of the four End Effectors for 13 positions from the Arttreeks output file:

```
posi1 = Table[qhat[i][[{1, 2, 3, 4}]], {i, 1, 4}];
posi2 = Table[qhat[i][[{5, 6, 7, 8}]], {i, 1, 4}];
posi3 = Table[qhat[i][[{9, 10, 11, 12}]], {i, 1, 4}];
posi4 = Table[qhat[i][[{13, 14, 15, 16}]], {i, 1, 4}];
posi5 = Table[qhat[i][[{17, 18, 19, 20}]], {i, 1, 4}];
posi6 = Table[qhat[i][[{21, 22, 23, 24}]], {i, 1, 4}];
posi7 = Table[qhat[i][[{25, 26, 27, 28}]], {i, 1, 4}];
posi8 = Table[qhat[i][[{29, 30, 31, 32}]], {i, 1, 4}];
posi9 = Table[qhat[i][[{33, 34, 35, 36}]], {i, 1, 4}];
posi10 = Table[qhat[i][[{37, 38, 39, 40}]], {i, 1, 4}];
posi11 = Table[qhat[i][[{41, 42, 43, 44}]], {i, 1, 4}];
posi12 = Table[qhat[i][[{45, 46, 47, 48}]], {i, 1, 4}];
posi13 = Table[qhat[i][[{49, 50, 51, 52}]], {i, 1, 4}];
```

Changing to dual quarternion

```
pos1dq = Table[HM2dq[posi1[[i]]], {i, 1, 4}];
pos1dq[[1]];
pos2dq = Table[HM2dq[posi2[[i]]], {i, 1, 4}];
pos3dq = Table[HM2dq[posi3[[i]]], {i, 1, 4}];
pos4dq = Table[HM2dq[posi4[[i]]], {i, 1, 4}];
pos5dq = Table[HM2dq[posi5[[i]]], {i, 1, 4}];
pos6dq = Table[HM2dq[posi6[[i]]], {i, 1, 4}];
pos7dq = Table[HM2dq[posi7[[i]]], {i, 1, 4}];
pos8dq = Table[HM2dq[posi8[[i]]], {i, 1, 4}];
pos9dq = Table[HM2dq[posi9[[i]]], {i, 1, 4}];
pos10dq = Table[HM2dq[posi10[[i]]], {i, 1, 4}];
pos11dq = Table[HM2dq[posi11[[i]]], {i, 1, 4}];
pos12dq = Table[HM2dq[posi12[[i]]], {i, 1, 4}];
pos13dq = Table[HM2dq[posi13[[i]]], {i, 1, 4}];
```

Importing the 13 positions of each finger from the input file.

```
f1 = {{{-0.5449935126146968`, -0.6789361555517063`,
      -0.19510357029818937`, 0.4516219267810892`}, {-85.19445718241812`,
      -54.30106466920125`, 174.13190060705054`, -109.21442057591622`}},
    {{-0.4787877278136222`, -0.7013904081304638`, -0.21049964172132635`,
      0.4842558289921181`}, {-92.73155486632947`,
      -55.939210753461616`, 173.8530138831376`, -97.13452345068822`}},
    {{-0.4688287646151838`, -0.6938403786215882`, -0.29362015943602726`,
      0.46105565871856913`}, {-96.09504471321011`,
      -68.1174483815549`, 167.13938295376016`, -93.78317820365574`}},
    {{-0.48358297791085064`, -0.7276961497587306`, -0.20140660635366395`,
      0.44276539613918897`}, {-81.29545213434064`,
      -55.800998847245395`, 174.82110546895998`, -100.97704939437325`}},
    {{-0.48731110419895085`, -0.7414723373075074`, -0.194456665363787`,
      0.41825024330864113`}, {-81.59601045388546`,
      -49.44870196210438`, 174.1751705981321`, -101.75239438674869`}},

    {{-0.5113447857167359`, -0.7102237622515984`, -0.20825359816593725`,
      0.4367369419964731`}, {-85.56702871725327`,
      -51.72631877801276`, 170.7155353089301`, -102.89807406134538`}},
    {{-0.3780919128630047`, -0.7810140549143229`, -0.23221347710375903`,
      0.4394774766757258`}, {-84.47312222673568`,
      -63.08987403667135`, 166.19830103995295`, -96.97697743871825`}},
    {{-0.31706567789443924`, -0.8047278531974544`, -0.2773620091815284`,
      0.41827353974682846`}, {-80.72815805344095`,
      -74.0492969269594`, 158.50380024036784`, -98.5544709153052`}},
    {{-0.24177602656530617`, -0.8012665334193356`, -0.3294723744811509`,
      0.43699456501678074`}, {-81.44845440077401`,
      -89.20173076276288`, 147.71644124552657`, -97.25100038598325`}},
    {{-0.2490569116618841`, -0.8476382890681676`, -0.25039595880669624`,
      0.39595687829902504`}, {-71.49425496302182`,
      -76.22127481764441`, 149.879690812792`, -113.35815310439723`}},
    {{-0.245494560467246`, -0.8438369300217449`, -0.29760239917967884`,
      0.372967114254097 6`}, {-66.66956417410535`, -82.909236004025`,
      149.31103865065307`, -112.32509678736957`}},
    {{-0.25291394294825137`, -0.8364177418470884`, -0.3673832021411213`,
      0.3185427465354091`}, {-78.92448861068657`,
      -80.69042845299136`, 153.6288904822122`, -97.35376546970976`}},
    {{-0.27064782231271733`, -0.8047100491640737`, -0.3667699787930603`,
      0.38035677423686837`}, {-71.51629074552443`, -90.20744397174482`,
      160.94277926451272`, -86.54397007107465`}}};
 f2 = {{{-0.5463614215264885`, -0.6395419373953746`, -0.3131303129329542`,
      0.44093617962539194`}, {-82.11624720720779`,
      -79.51259800608639`, 176.4120326758282`, -91.79749225425356`}},
    {{-0.49525640105107094`, -0.6768852095643104`, -0.30592356394032216`,
      0.4505089159129616`}, {-83.59502930225375`,
      -76.16009887396453`, 179.39525764417345`, -84.50749791507481`}},
    {{-0.4597361889800615`, -0.67593312601048`, -0.3583157888356801`,
      0.45096212831550975`}, {-90.89603736811112`,
      -82.64963246361489`, 173.5536583913942`, -78.64697279215578`}},
```

```
{{-0.4597361889800615`, -0.67593312601048`, -0.3583157888356801`,
  0.45096212831550975`}, {-90.89603736811112`,
  -82.64963246361489`, 173.5536583913942`, -78.64697279215578`}},
 {{-0.44916644153380975`, -0.6720330626116185`, -0.33875651173096766`,
  0.4815237235241581`}, {-87.10554289392029`,
  -85.4077113453969`, 174.37358285757298`, -77.77707321389691`}},
 {{-0.45058726813754263`, -0.6755704519756294`, -0.31646852761990346`,
  0.49032983718705164`}, {-93.62344536023114`,
  -76.00381996968828`, 172.2916112997767`, -79.55174655852404`}},
 {{-0.49201739035205244`, -0.6661020805475875`, -0.2710950574350626`,
  0.49064689514518656`}, {-91.80784200363618`,
  -65.87890492577071`, 172.5026167607502`, -86.1893228953737`}},
 {{-0.37582402007679544`, -0.7317323096433982`, -0.30009679270122763`,
  0.4829762395477554`}, {-86.91765315512251`,
  -80.20703107102403`, 168.06357751621297`, -84.7256531200085`}},
 {{-0.3199908558495015`, -0.7340579585672481`, -0.37977048963309823`,
  0.463183700966069`}, {-81.90211234195633`,
  -99.73144236108725`, 156.1027735814685`, -86.6467433307904`}},
 {{-0.24133285538080218`, -0.7720092838878951`, -0.36274623780501086`,
  0.46279075775447237`}, {-78.3237363406711`,
  -100.62999359993682`, 154.29894007682702`, -87.7675699895837`}},
 {{-0.21604684447106462`, -0.8131761997360064`, -0.28493505086093074`,


  0.4592169922464553`}, {-75.89753240935134`,
  -89.70361219477023`, 158.36085515412978`, -96.29370706269928`}},
 {{-0.22342327503592926`, -0.820674971704024`, -0.35917445178737784`,
  0.38414625362491256`}, {-59.364102727995736`,
  -99.38154254818058`, 158.23590577034562`, -98.89195014417234`}},
 {{-0.243879396654489`, -0.8188496176518679`, -0.41697757297868965`,
  0.31006103784841116`}, {-69.68296440610828`,
  -95.11602078625535`, 162.67954886474803`, -87.22873824663631`}},
 {{-0.2534686225264062`, -0.8222541951183165`, -0.3165085670302328`,
  0.39934198752618555`}, {-63.847375289299606`,
  -86.38333111598354`, 176.96717496483018`, -78.13036615230598`}}};
f3 = {{{-0.3695115485769071`, -0.9005458044090653`, 0.08129288041397963`,
  0.21417268085244728`}, {-26.769184984395835`,
  5.734439997260832`, 180.82437893700623`, -90.7077954718203`}},
 {{-0.29787016995419413`, -0.9154114169960738`, 0.025938135101782663`,
  0.26948564457525676`}, {-41.33525432130405`,
  -3.605355537315994`, 181.95249767399756`, -75.44903310080637`}},
 {{-0.2985230519160391`, -0.9214322728465643`, -0.01193265020681511`,
  0.2484032324479506`}, {-45.5990007314612`, -7.097500057954619`,
  179.5979208917265`, -72.49961736511867`}},
 {{-0.2445949442826587`, -0.9021668359814692`, 0.05073006064067082`,
  0.35170267874149286`}, {-50.31749107560926`,
  -3.103040861762053`, 179.29020324897127`, -68.81456647404862`}},
 {{-0.21278994207739457`, -0.912391781529404`, 0.03910747745214355`,
  0.34746551304500567`}, {-53.39790189708258`,
  -2.89657790178045`, 179.52966751009293`, -60.513351566l945`}},
 {{-0.2364554204725249`, -0.9083210609468502`, -0.01878160947142321`,
  0.34451260574210185`}, {-52.68703716207557`,
  -13.055873988015751`, 179.0668004962453`, -60.821861878288075`}},
 {{-0.2248203357370262`, -0.9604777845463016`, 0.04730354627216543`,
  0.1571642979248018`}, {-24.007832805902186`,
  1.0888133633262793`, 170.2735639516995`, -78.93784758928514`}},
 {{-0.20577681856383728`, -0.9758925512690055`, 0.03310272305448921`,
  0.06475985674624936`}, {-13.388962235786112`,
  2.5235739873346406`, 163.58885934847908`, -88.13542815748143`}},
 {{-0.19472877238828976`, -0.9794038451706566`, -0.04242855987680211`,
  0.032382565924627224`}, {-11.705347128169208`,
  -7.715417433100317`, 154.8993367092134`, -100.78638001926744`}},
 {{-0.14811819241389604`, -0.9885731672071062`, -0.0036706557641062202`,
  0.027760051160111935`}, {-10.360134015345317`,
  -1.9828123662850048`, 153.42334964528365`, -105.6019351262104`}},
 {{-0.11409028004564294`, -0.9924472890535188`, -0.040320673134459806`,
  0.020150180332007016`}, {-4.191392687474674`, -7.863940245855041`,
  154.89257235516803`, -101.10929675643672`}}, {{0.12868038953757963`,
  0.9893531843010391`, 0.06287961472295717`, 0.025841596586572292`},
 {12.66269885115024`, 6.656166451553188`, -163.85473074051208`, 80.81473749719824`}},
 {{0.07993842051728611`, 0.943663614262043`, 0.3140052411655827`,
  0.06715311283658024`}, {-11.100544637768563`,
  49.173045654437516`, -157.2842727190136`, 57.66868938446976`}}};
f4 = {{{-0.47048576193030955`, -0.8577318336322863`, -0.07113921061571293`,
  0.19462389911731154`}, {-26.629436259624157`,
  -24.830017798988635`, 187.37973599400965`, -105.31194212630537`}},
```

```
{{-0.4392932188454455`, -0.8690708935584122`, -0.10856174990868041`,
  0.19987895412663056`}, {-31.17893175563956`,
  -30.260891392576344`, 185.50684155398892`, -99.34315476658189`}},
{{-0.4299480354215151`, -0.8610027843906127`, -0.17247637542178085`,
  0.20993044569608413`}, {-41.994233572815446`,
  -38.59304280484468`, 181.68176812718306`, -95.0231045516079`}},
{{-0.4285111882118056`, -0.8815913755157664`, -0.06954448838155938`,
  0.18530615836844952`}, {-24.160344486913242`,
  -24.05893498043629`, 184.13570057738642`, -101.22439848874583`}},
{{-0.38286652493949275`, -0.887401024521549`, 0.0007591915119323693`,
  0.2567724077604718`}, {-41.47300064435934`,
  -8.71245156246053`, 185.19465477998162`, -92.49693335531774`}},
{{-0.37578336493427905`, -0.8757608176529265`, 0.007791330605571742`,
  0.302934230601202 8`}, {-51.17831605993471`,
  -6.6031860858993525`, 185.61565761189047`, -87.34887558888421`}},
{{-0.22492246649055225`, -0.9340788609905258`, -0.02984823604415215`,
  0.27570935479867337`}, {-50.44852721500156`,
  -16.851928568342544`, 181.78949065669752`, -78.56803336606241`}},
{{-0.22670504995012603`, -0.9572995021154068`, -0.047717763555969406`,
  0.17293206359123173`}, {-36.14350864841917`,
  -16.995151493995728`, 174.39238737900718`, -93.3415756162681 3`}},
{{-0.2106393491174643`, -0.9575659312860172`, -0.12960981766687163`,
  0.14798596895676944`}, {-33.61690826289315`,
  -31.01805510460718`, 164.15610583728844`, -104.78448505355756`}},
{{-0.17232956720643683`, -0.9740515371094323`, -0.07850802741551462`,
  0.12394600821534776`}, {-28.176530354337746`,
  -22.405506963832487`, 163.16482319977632`, -111.90371973952836`}},
{{-0.2011169495905609`, -0.953939035662607`, -0.2010882954102885`,
  0.09547662685426048`}, {-17.600597472328275`,
  -41.12325203481996`, 157.91414627699285`, -115.35982921572558`}},
{{-0.23838746842939132`, -0.9417665747312555`, -0.23691678060753027`,
  0.010843093716645623`}, {-22.292577343303275`,
  -36.90115963460526`, 164.45975166112322`, -101.74908712344114`}},
{{-0.22296814344743363`, -0.9158675247330578`, -0.3206519087382634`,
  0.0930281547138016`}, {-17.32667034130419`, -62.97553865136564`,
  167.2925728222509`, -84.89756087919189`}}};
```

Subtracting the End effector positions of the Arttreeks output from the fingers position from the input file(Actual Data).

```
f1[[1]] - pos1dq[[1]]
f2[[1]] - pos1dq[[2]]
f3[[1]] - pos1dq[[3]]
f4[[1]] - pos1dq[[4]]

f1[[2]] - pos2dq[[1]]
f2[[2]] - pos2dq[[2]]
f3[[2]] - pos2dq[[3]]
f4[[2]] - pos2dq[[4]]

f1[[3]] - pos3dq[[1]]
f2[[3]] - pos3dq[[2]]
f3[[3]] - pos3dq[[3]]
f4[[3]] - pos3dq[[4]]

f1[[4]] - pos4dq[[1]]
f2[[4]] - pos4dq[[2]]
f3[[4]] - pos4dq[[3]]
f4[[4]] - pos4dq[[4]]

f1[[5]] - pos5dq[[1]]
f2[[5]] - pos5dq[[2]]
f3[[5]] - pos5dq[[3]]
f4[[5]] - pos5dq[[4]]

f1[[6]] - pos6dq[[1]]
f2[[6]] - pos6dq[[2]]
f3[[6]] - pos6dq[[3]]
f4[[6]] - pos6dq[[4]]
```

```
f1[[7]] - pos7dq[[1]]
f2[[7]] - pos7dq[[2]]
f3[[7]] - pos7dq[[3]]
f4[[7]] - pos7dq[[4]]

f1[[8]] - pos8dq[[1]]
f2[[8]] - pos8dq[[2]]
f3[[8]] - pos8dq[[3]]
f4[[8]] - pos8dq[[4]]

f1[[9]] - pos9dq[[1]]
f2[[9]] - pos9dq[[2]]
f3[[9]] - pos9dq[[3]]
f4[[9]] - pos9dq[[4]]


f1[[10]] - pos10dq[[1]]
f2[[10]] - pos10dq[[2]]
f3[[10]] - pos10dq[[3]]
f4[[10]] - pos10dq[[4]]


f1[[11]] - pos11dq[[1]]
f2[[11]] - pos11dq[[2]]
f3[[11]] - pos11dq[[3]]
f4[[11]] - pos11dq[[4]]


f1[[12]] - pos12dq[[1]]
f2[[12]] - pos12dq[[2]]
f3[[12]] - pos12dq[[3]]

f4[[12]] - pos12dq[[4]]


f1[[13]] - pos13dq[[1]]
f2[[13]] - pos13dq[[2]]
f3[[13]] - pos13dq[[3]]
f4[[13]] - pos13dq[[4]]
```

$$\{\{0., -2.22045\times10^{-16}, 8.32667\times10^{-17}, 5.55112\times10^{-17}\}, \{0., 8.52651\times10^{-14}, 2.84217\times10^{-14}, 5.68434\times10^{-14}\}\}$$

$$\{\{-1.11022\times10^{-16}, -1.11022\times10^{-16}, 1.11022\times10^{-16}, 0.\}, \{4.26326\times10^{-14}, 2.84217\times10^{-14}, 0., -1.42109\times10^{-14}\}\}$$

$$\{\{-5.55112\times10^{-17}, -1.11022\times10^{-16}, 2.63678\times10^{-16}, 0.\}, \{2.13163\times10^{-14}, 8.88178\times10^{-14}, -8.52651\times10^{-14}, 2.84217\times10^{-14}\}\}$$

$$\{\{1.66533\times10^{-16}, -3.33067\times10^{-16}, 5.68989\times10^{-16}, 5.55112\times10^{-17}\}, \{-3.33955\times10^{-13}, 2.84217\times10^{-13}, 8.52651\times10^{-14}, 8.52651\times10^{-14}\}\}$$

$$\{\{-1.11022\times10^{-16}, 0., -2.77556\times10^{-17}, 0.\}, \{-1.23634\times10^{-12}, -9.52127\times10^{-13}, 2.33058\times10^{-12}, -1.62004\times10^{-12}\}\}$$

$$\{\{1.11022\times10^{-16}, -1.11022\times10^{-16}, -2.22045\times10^{-16}, -2.22045\times10^{-16}\}, \{1.64846\times10^{-12}, 8.2423\times10^{-13}, -2.47269\times10^{-12}, 1.39266\times10^{-12}\}\}$$

$$\{\{0., 0., 4.51028\times10^{-17}, -5.55112\times10^{-17}\}, \{1.06581\times10^{-13}, 1.3145\times10^{-13}, -1.25056\times10^{-12}, 6.82121\times10^{-13}\}\}$$

$$\{\{5.55112\times10^{-17}, -1.11022\times10^{-16}, 9.4369\times10^{-16}, 1.11022\times10^{-16}\}, \{-1.38556\times10^{-12}, 6.78568\times10^{-13}, 0., 1.13687\times10^{-13}\}\}$$

$$\{\{1.11022\times10^{-16}, 1.11022\times10^{-16}, -5.55112\times10^{-17}, 2.77556\times10^{-16}\}, \{-1.63425\times10^{-12}, -8.66862\times10^{-13}, 2.30216\times10^{-12}, -1.47793\times10^{-12}\}\}$$

$$\{\{-5.55112\times10^{-17}, 0., -1.11022\times10^{-16}, 5.55112\times10^{-17}\}, \{1.62004\times10^{-12}, 1.0516\times10^{-12}, -2.359\times10^{-12}, 1.37845\times10^{-12}\}\}$$

$$\{\{0., -1.11022\times10^{-16}, -4.51028\times10^{-17}, 0.\},$$

$\{5.96856\times10^{-13}, 5.86198\times10^{-14}, -1.19371\times10^{-12}, 6.6791\times10^{-13}\}\}$

$\{\{5.55112\times10^{-17}, -1.11022\times10^{-16}, 7.49401\times10^{-16}, 5.55112\times10^{-17}\},$
$\{-1.0516\times10^{-12}, 5.61329\times10^{-13}, -2.84217\times10^{-14}, 7.10543\times10^{-14}\}\}$

$\{\{-5.55112\times10^{-17}, 0., 5.55112\times10^{-17}, -1.11022\times10^{-16}\},$
$\{-1.22213\times10^{-12}, -8.88178\times10^{-13}, 2.24532\times10^{-12}, -1.80478\times10^{-12}\}\}$

$\{\{1.66533\times10^{-16}, 0., 5.55112\times10^{-17}, 2.22045\times10^{-16}\},$
$\{1.29319\times10^{-12}, 1.22213\times10^{-12}, -2.18847\times10^{-12}, 1.42109\times10^{-12}\}\}$

$\{\{0., 0., -1.52656\times10^{-16}, -1.11022\times10^{-16}\},$
$\{3.05533\times10^{-13}, 1.38556\times10^{-13}, -1.16529\times10^{-12}, 7.10543\times10^{-13}\}\}$

$\{\{1.11022\times10^{-16}, -3.33067\times10^{-16}, 9.85323\times10^{-16}, 2.77556\times10^{-17}\},$
$\{-1.00897\times10^{-12}, 5.25802\times10^{-13}, -1.13687\times10^{-13}, 9.9476\times10^{-14}\}\}$

$\{\{-2.22045\times10^{-16}, 1.11022\times10^{-16}, 2.77556\times10^{-17}, 1.11022\times10^{-16}\},$
$\{-1.33582\times10^{-12}, -7.10543\times10^{-13}, 2.30216\times10^{-12}, -1.79057\times10^{-12}\}\}$

$\{\{5.55112\times10^{-17}, -1.11022\times10^{-16}, -5.55112\times10^{-17}, -5.55112\times10^{-17}\},$
$\{1.35003\times10^{-12}, 1.20792\times10^{-12}, -2.27374\times10^{-12}, 1.39266\times10^{-12}\}\}$

$\{\{2.77556\times10^{-17}, 2.22045\times10^{-16}, -1.11022\times10^{-16}, 0.\},$
$\{1.84741\times10^{-13}, 1.84741\times10^{-13}, -1.10845\times10^{-12}, 7.4607\times10^{-13}\}\}$

$\{\{0., 1.11022\times10^{-16}, 4.87457\times10^{-16}, 0.\},$
$\{-8.73968\times10^{-13}, 3.6593\times10^{-13}, -2.27374\times10^{-13}, 7.10543\times10^{-14}\}\}$

$\{\{0., 0., 8.32667\times10^{-17}, 1.11022\times10^{-16}\},$
$\{-1.50635\times10^{-12}, -6.25278\times10^{-13}, 2.33058\times10^{-12}, -1.62004\times10^{-12}\}\}$

$\{\{0., -1.11022\times10^{-16}, -1.11022\times10^{-16}, 0.\},$
$\{1.33582\times10^{-12}, 1.09424\times10^{-12}, -2.41585\times10^{-12}, 1.42109\times10^{-12}\}\}$

$\{\{2.77556\times10^{-17}, 0., 9.36751\times10^{-17}, 5.55112\times10^{-17}\},$
$\{-4.9738\times10^{-14}, 3.0731\times10^{-13}, -1.3074\times10^{-12}, 7.4607\times10^{-13}\}\}$

$\{\{5.55112\times10^{-17}, 0., 3.14852\times10^{-16}, 0.\},$
$\{-5.8975\times10^{-13}, 2.66454\times10^{-13}, -2.27374\times10^{-13}, 7.10543\times10^{-14}\}\}$

$\{\{-1.66533\times10^{-16}, 0., 0., -1.11022\times10^{-16}\},$
$\{-1.42109\times10^{-12}, -9.23706\times10^{-13}, 1.98952\times10^{-12}, -1.87583\times10^{-12}\}\}$

$\{\{0., 1.11022\times10^{-16}, 0., 5.55112\times10^{-17}\},$
$\{1.32161\times10^{-12}, 1.25056\times10^{-12}, -2.04636\times10^{-12}, 1.66267\times10^{-12}\}\}$

$\{\{2.77556\times10^{-17}, 1.11022\times10^{-16}, -1.17961\times10^{-16}, 0.\},$
$\{2.91323\times10^{-13}, -3.01981\times10^{-14}, -1.08002\times10^{-12}, 8.2423\times10^{-13}\}\}$

$\{\{-8.32667\times10^{-17}, 1.11022\times10^{-16}, 3.71231\times10^{-16}, 5.55112\times10^{-17}\},$
$\{-1.01608\times10^{-12}, 2.55795\times10^{-13}, -1.98952\times10^{-13}, 4.26326\times10^{-14}\}\}$

$\{\{-1.11022\times10^{-16}, 0., 0., -1.11022\times10^{-16}\},$
$\{-1.49214\times10^{-12}, -1.02318\times10^{-12}, 1.84741\times10^{-12}, -1.98952\times10^{-12}\}\}$

$\{\{5.55112\times10^{-17}, -1.11022\times10^{-16}, 0., -5.55112\times10^{-17}\},$
$\{1.79057\times10^{-12}, 1.29319\times10^{-12}, -1.93268\times10^{-12}, 1.62004\times10^{-12}\}\}$

$\{\{1.66533\times10^{-16}, 0., -3.747\times10^{-16}, 4.16334\times10^{-17}\},$
$\{1.41753\times10^{-12}, -2.52243\times10^{-13}, -9.09495\times10^{-13}, 7.95808\times10^{-13}\}\}$

$\{\{5.55112\times10^{-17}, -1.11022\times10^{-16}, 5.55112\times10^{-16}, 2.77556\times10^{-17}\},$
$\{-1.1795\times10^{-12}, 3.51719\times10^{-13}, -2.27374\times10^{-13}, 1.13687\times10^{-13}\}\}$

$\{\{-8.32667\times10^{-17}, 0., -5.55112\times10^{-17}, 5.55112\times10^{-17}\},$
$\{-1.90425\times10^{-12}, -1.10845\times10^{-12}, 1.62004\times10^{-12}, -1.8332\times10^{-12}\}\}$

$\{\{-8.32667\times10^{-17}, 0., 5.55112\times10^{-17}, 0.\},$
$\{1.60583\times10^{-12}, 1.42109\times10^{-12}, -1.7053\times10^{-12}, 1.86162\times10^{-12}\}\}$

$\{\{2.498\times10^{-16}, -1.11022\times10^{-16}, 5.20417\times10^{-16}, -1.38778\times10^{-17}\},$
$\{-3.6593\times10^{-13}, 1.01252\times10^{-13}, -1.16529\times10^{-12}, 8.52651\times10^{-13}\}\}$

$\{\{3.88578\times10^{-16}, -2.22045\times10^{-16}, 6.66134\times10^{-16}, 5.55112\times10^{-17}\},$
$\{-1.64846\times10^{-12}, 4.05009\times10^{-13}, -2.27374\times10^{-13}, 1.98952\times10^{-13}\}\}$

$\{\{-2.77556\times10^{-17}, 1.11022\times10^{-16}, -5.55112\times10^{-17}, -1.11022\times10^{-16}\},$
$\{-1.66267\times10^{-12}, -9.52127\times10^{-13}, 1.7053\times10^{-12}, -2.10321\times10^{-12}\}\}$

$\{\{-2.22045\times10^{-16}, 1.11022\times10^{-16}, -1.66533\times10^{-16}, -1.11022\times10^{-16}\},$
$\{1.56319\times10^{-12}, 1.27898\times10^{-12}, -1.56319\times10^{-12}, 2.00373\times10^{-12}\}\}$

$\{\{-2.77556\times10^{-17}, 2.22045\times10^{-16}, -5.50775\times10^{-16}, -6.93889\times10^{-18}\},$
$\{-4.29878\times10^{-13}, -1.21902\times10^{-13}, -1.10845\times10^{-12}, 8.2423\times10^{-13}\}\}$

$\{\{2.77556\times10^{-17}, 1.11022\times10^{-16}, 2.498\times10^{-16}, 1.38778\times10^{-17}\},$
$\{-1.15463\times10^{-12}, 2.06057\times10^{-13}, -1.42109\times10^{-13}, 5.68434\times10^{-14}\}\}$

$\{\{1.38778\times10^{-16}, 0., 1.66533\times10^{-16}, 5.55112\times10^{-17}\},$
$\{-1.89004\times10^{-12}, -8.95284\times10^{-13}, 1.62004\times10^{-12}, -2.04636\times10^{-12}\}\}$

$\{\{-8.32667\times10^{-17}, -1.11022\times10^{-16}, 5.55112\times10^{-17}, -5.55112\times10^{-17}\},$
$\{1.82609\times10^{-12}, 1.16529\times10^{-12}, -1.59162\times10^{-12}, 1.97531\times10^{-12}\}\}$

$\{\{-1.38778\times10^{-17}, -1.11022\times10^{-16}, 1.17961\times10^{-16}, 3.46945\times10^{-18}\},$
$\{2.18314\times10^{-12}, -4.17444\times10^{-14}, -9.37916\times10^{-13}, 8.95284\times10^{-13}\}\}$

$\{\{2.498\times10^{-16}, 0., 1.66533\times10^{-16}, 1.38778\times10^{-17}\},$
$\{-5.96856\times10^{-13}, 1.27898\times10^{-13}, 0., 1.13687\times10^{-13}\}\}$

$\{\{0., 1.11022\times10^{-16}, 0., 1.11022\times10^{-16}\},$
$\{-2.14584\times10^{-12}, -7.81597\times10^{-13}, 1.47793\times10^{-12}, -1.91847\times10^{-12}\}\}$

$\{\{0., 0., -2.77556\times10^{-16}, 0.\},$
$\{1.93268\times10^{-12}, 9.52127\times10^{-13}, -1.64846\times10^{-12}, 1.9611\times10^{-12}\}\}$

$\{\{-3.60822\times10^{-16}, 0., -3.88578\times10^{-16}, 6.93889\times10^{-18}\},$
$\{-2.54019\times10^{-13}, -3.01981\times10^{-14}, 9.37916\times10^{-13}, -9.23706\times10^{-13}\}\}$

$\{\{4.66294\times10^{-15}, -2.44249\times10^{-15}, 4.85723\times10^{-15}, 2.37657\times10^{-16}\},$
$\{-1.43139\times10^{-11}, 3.18323\times10^{-12}, 1.08002\times10^{-12}, 9.9476\times10^{-13}\}\}$

$\{\{0., -1.11022\times10^{-16}, -5.55112\times10^{-17}, 0.\},$
$\{-2.00373\times10^{-12}, -8.95284\times10^{-13}, 1.62004\times10^{-12}, -1.8332\times10^{-12}\}\}$

$\{\{-1.66533\times10^{-16}, 2.22045\times10^{-16}, -1.66533\times10^{-16}, -5.55112\times10^{-17}\},$
$\{1.77636\times10^{-12}, 1.0516\times10^{-12}, -1.62004\times10^{-12}, 1.9611\times10^{-12}\}\}$

$\{\{2.08167\times10^{-16}, -1.11022\times10^{-16}, 0., 0.\},$
$\{-1.01785\times10^{-12}, -1.56319\times10^{-13}, 8.52651\times10^{-13}, -7.17648\times10^{-13}\}\}$

$\{\{7.77156\times10^{-16}, -3.33067\times10^{-16}, 3.33067\times10^{-16}, 4.16334\times10^{-17}\},$
$\{-1.31806\times10^{-12}, 2.20268\times10^{-13}, 2.84217\times10^{-13}, 1.84741\times10^{-13}\}\}$