Kinematic Model of the Hand using Computer Vision

Edgar Simó Serra

May 12, 2011



Resumen

La biotecnología es una ciencia en auge y en especial el diseño de interfaces humano-máquina. El objetivo de este proyecto es avanzar en dicho campo y en concreto explorar el diseño de exoesqueletos y prótesis de la mano humana.

La metodología utilizada en este proyecto fundamentalmente consta de tres fases. En primer lugar, se ha establecido un modelo teórico de la cinemática de la mano recurriendo a la documentación médica especializada para concretar su anatomía. Posteriormente se ha procedido a sintetizar la mano en sus parámetros simplificados y así definir un modelo robótico.

Para ajustar dicho modelo a una mano real se procede a capturar el movimiento de ésta en una secuencia de imágenes mediante ordenador. Para ello se utilizan unas marcas en las uñas de la mano con una geometría específica de tal manera que permite la estimación de su pose, es decir su posición e orientación, en el espacio. Esta secuencia de poses estimadas permite caracterizar el movimiento completo de la mano.

Por último, mediante la síntesis cinemática dimensional, se definen las ecuaciones de movimiento parametrizadas del modelo teórico de la mano. Estas ecuaciones permiten ajustar el modelo a la secuencia de poses estimadas mediante visión por ordenador y así crear un modelo personalizado de la mano. Gracias a este sistema, se puede realizar un estudio sobre la correspondencia entre señales electomiográficas y los movimientos de la mano y así lograr una mejor funcionalidad de las prótesis.

En definitiva, este proyecto ha logrado diseñar un algoritmo robusto para el seguimiento y estimación de las poses de las uñas de las manos y ha conseguido definir las ecuaciones de movimiento y crear una aplicación para resolverlas. Asimismo, ha encontrado modelos no antropomórficos que podrían ser de utilidad en el diseño de exoesqueletos.



Abstract

Biotechnology is a science that is growing rapidly. The objective of this project is to advance in the field. Specifically it aims to study applications of kinematics in the field of human-machine interfaces namely exoskeleton and prosthesis designs for the human hand.

The methodology used in this project consists of three phases. First a theoretical model of the hand kinematics is defined from medical literature. This is done by synthesizing the hand into its simplified parameters to define a robotic model.

The adjustment of the theoretical model to a hand is then done by capturing the movement using computer vision. This is done by using markers on each nail to be able to estimate their poses which consist of their spatial orientation and position. This sequence permits estimating the movement of the entire hand.

Dimensional kinematic synthesis is finally applied to adapt the theoretical model to the dataset provided by computer vision. This is done by defining the equations of the movement of the theoretical hand model and is then solved by a numerical solver. This allows the creation of a personalized hand model that can then be used for the study of correspondences between electromyography (EMG) and the movements of the hand.

In conclusion, this project has designed a robust algorithm for the tracking and estimation of the poses of the nails of the hand. It has also defined the movement equations and created an application to solve them. This has led to the finding of many non-anthropomorphic models that can be of use in the design of exoskeletons.





Contents

Resumen **1**

Abstract **3**

Table of Contents **5**

List of Figures **11**

List of Tables **13**

Glossary **15**

Nomenclature **17**

Preface **19**

Introduction **21**

1 Overview **23**

1.1 Motivation 23

1.2 Objectives 24

1.3 Scope 25

2 Kinematic Hand Model **27**

2.1 State of the Art . . . 27

2.2 Hand Anatomy . . . 28

2.2.1 Joints . . . 29

2.2.2 Dimensions . . . 30

2.2.3 Movement . . . 30

2.3 Robotic Hand Model . . . 30

2.3.1 Joint model with D-H Parameters . . . 31

2.3.2 Degrees of freedom . . . 33

2.3.3 Model . . . 33

3 Detection of Hand Characteristics **37**

3.1 State of the Art . . . 37

3.2 Approach . . . 38

3.3 Markers . . . 38

3.4 Pose Estimation Problem . . . 39

3.5 Algorithm . . . 40

3.5.1 Acquiring Images . . . 41

3.5.2 Points of Interest . . . 42

3.5.3 Solving 2D-3D Correspondence . . . 42

3.5.4 Descriptors . . . 43

3.5.5 Solving the Pose Estimation Problem . . . 45

3.6 Reliability and Error . . . 46



3.6.1 Marker Simulation . . . 46
3.6.2 Error Analysis . . . 47
3.7 Experimental Dataset . . . 49

4 Kinematic Synthesis 51

4.1 State of the Art . . . 51
4.2 Dimensional Kinematic Synthesis . . . 52
4.3 Kinematic Chain . . . 52
4.4 Screws . . . 53
4.5 Quaternions . . . 54
4.6 Dual Unit . . . 54
4.7 Plücker Coordinates . . . 55
4.8 Clifford Algebra . . . 55
4.8.1 Dual Quaternions . . . 56
4.9 Exponential Map . . . 59
4.10 Forward Kinematics . . . 61
4.11 Design Equations . . . 61

5 Non-linear Solver 63

5.1 State of the Art . . . 63
5.2 System Dimension . . . 63
5.2.1 System Reduction . . . 65
5.2.2 Solution Bound . . . 65
5.3 Numerical Solver . . . 66

5.3.1 Levenberg-Marquadt . . . 67
5.3.2 Genetic Algorithm . . . 67
5.3.3 Hybrid Solver . . . 68
5.4 Implementation . . . 68
5.5 System Solutions . . . 70

6 Results **71**
6.1 Overview . . . 71
6.2 Hand Detection . . . 71
6.3 Kinematic Synthesis . . . 72
6.4 Environmental Impact . . . 74
6.5 Budget . . . 74

Conclusions **75**

Acknowledgements **77**

Bibliography **87**

A MINPACK Solver **89**
A.1 Rotation Error . . . 89
A.2 Translation Error . . . 89
A.3 Angular Error . . . 91
A.4 Full Error . . . 92



B Genetic Algorithm Solver **95**

B.1 Algorithm Implementation . . . 95

B.1.1 Selection Function . . . 95

B.1.2 Fitness Function . . . 96

B.1.3 Crossover Function . . . 96

B.1.4 Mutation Function . . . 96

B.1.5 Parameters . . . 98

B.2 Runtime Information . . . 98

C Budget **101**

C.1 Programming . . . 101

C.2 Software Licensing . . . 102

C.3 Equipment . . . 103

C.4 Total . . . 103

D Solver Manual **105**

D.1 Command Line Arguments . . . 105

D.1.1 Levenberg-Marquadt . . . 105

D.1.2 Genetic algorithm . . . 106

D.2 Input Format . . . 107

D.2.1 Angles . . . 108

D.2.2 Axes . . . 108

D.2.3 Forward Kinematics . . . 108

D.3 Synthetic Data Generator . . . 109



D.3.1 Command Line Arguments 109

D.3.2 Input 109



List of Figures

2.1 Bones of the human hand (Source: Wikipedia). 28

2.2 Visual representation of D-H parameters. 31

2.3 Computer model of the human hand. 33

3.1 Marker template. 38

3.2 A detected marker. 39

3.3 Flux diagram of the detection algorithms 40

3.4 An image captured by a Flea® 2 camera directly in grayscale. 41

3.5 Points of interest detected in an image. 42

3.6 Markers created from points of interest using heuristics. 43

3.7 Reconstructed poses from markers. 45

3.8 Projection of Simulated Marker 47

3.9 Simulation of Translation Error 48

3.10 Simulation of Rotation Error 48

4.1 Serial kinematic chain. 52

4.2 Tree-like kinematic chain. 53

4.3 A screw displacement. 53

5.1 Convergence of a hybrid solver run. 69

5.2 A solution of the general kinematics problem for a hand task. 70

6.1 Motion blur on a captured image. 72

6.2 Another solution of the general kinematics problem for a hand task. 73

A.1 Average rotation error from rotation noise. 90

A.2 Average translation error from rotation noise. 90

A.3 Average rotation error from translation noise. 91

A.4 Average translation error from translation noise. 91

A.5 Average rotation error from angular noise. 92

A.6 Average translation error from angular noise. 92

A.7 Average rotation error for full noise. 93

A.8 Average translation error for full noise. 93



List of Tables

2.1 Degrees of freedom provided by the different joints. 29

2.2 Average dimension of finger bones. 30

2.3 Joint rotation range. 31

2.4 D-H Common Parameters. 33

2.5 D-H Finger Parameters. 34

2.6 D-H Thumb Parameters. 35

4.1 Multiplication table for $Cl^{+}(0,3,1)$ (McArthy bases [51]). 57

5.1 Independent unknowns of the algebraic sets used by the solver. 64

5.2 System of equations parameters for different combinations of fingers considered. . 65

5.3 Run time results for genetic algorithm. 70

6.1 Run time results for genetic algorithm. 73

6.2 Total project budget. 74

B.1 Parameters of the genetic algorithm. 98

B.2 Runtime profile of the genetic algorithm. 99

B.3 Runs of the genetic algorithm. 100



C.1 Programming budget. 101

C.2 Software licensing budget. 102

C.3 Equipment budget. 103

C.4 Total project budget. 103



Glossary

AA adduction-abduction. 29

CMC carpometacarpal. 29, 32, 33

D-H Denavit-Hartenberg. 31, 32

DIP distal interphalangeal. 29

DoF degrees of freedom. 27, 29, 32, 33, 64, 65, 73

EMG electromyography. 3, 21, 23, 24, 27

FE flexion-extension. 29, 33

IP interphalangeal. 29

MCP metacarpophalangeal. 29, 32

MRI magnetic resonance imaging. 29, 37

PIP proximal interphalangeal. 29

PS pronation-supination. 29

ROI region of interest. 37

RU radio-ulna. 29



Nomenclature

$[M]$	A matrix.
$\mathbf{e}_i$	An algebra generator.
$\mathbf{s}$	A vector.
$\cdot$	The dot product of vectors.
ϵ	The dual unit such that $\epsilon^2 = 0$.
$\hat{Q}$	A dual quaternion; $\hat{Q} = \hat{q} + \epsilon\hat{q}^0$.
$\hat{q}$	A quaternion.
$\otimes$	The tensor product of algebras.
S	A dual vector; $\mathsf{S} = \mathbf{s} + \epsilon\mathbf{s}^0$.
$\times$	The cross product of vectors.



Preface

This final thesis is the culmination of the five year degree in Superior Industrial Engineering specialized in Automation for the author. It contains original work done from April 2010 to April 2011 at the Institute of Industrial Robotics (CSIC-IRI), Barcelona.

The topic is the result of a collaboration between Dr. Alba Perez Gracia and Dr. Francesc Moreno Noguer of the Kinematics & Robot design group and Perception & Manipulation group respectively. The project was supervised by Professor Alberto Sanfelu.





Introduction

The hand has always been of special importance to humans. It has allowed man to grasp and manipulate objects and has become an important social symbol [3]. The importance of the hand in human life is astonishing due to its flexibility. Not only can it manipulate and grasp objects, but it can also be used as a sensor or as a way to communicate. This has made individuals with damage to their hands to not function well and has led to the development of the field of prosthetic hand implants and exoskeleton design for augmented performance and rehabilitation.

Technology has recently reached the point at which it can create anthropomorphically correct prosthetic hands. This requires deep understanding of hand anatomy and the kinematics of its movement. It also requires being able to map EMG signals [36] to actual hand movements and the ability to create a small light-weight implant that is functionally equivalent to the human hand.

This project explores the movement of the hand, through the extraction of the exact kinematic model of the hand by means of computer vision. This consists of defining a theoretical hand model and being able to adjust it to match a real hand by using computer vision. One of the goals of the project is to develop an accurate personalized kinematic model that could be used in conjunction with EMG signals from the same hand to further explore and map the relationship between the EMG signals and the hand movements.

This thesis is divided into multiple chapters. An overview of the objectives and motivation of the project is detailed in Chapter 1. Chapter 2 presents an anthropomorphically correct hand model. The detection of characteristics of the hand to be able to construct the model is presented in Chapter 3. The theoretical fundamentals and design equations for hand model kinematics is displayed in Chapter 4. Chapter 5 implements a solver to identify the theoretical model of a real hand. Results of the project are summarized in Chapter 6.





1. Overview

This chapter gives a brief overview of the project presented in this final thesis. It highlights the social interest and motivations that justify the project and states its objectives.

1.1 Motivation

As technology advances, hardware gets smaller and more powerful. This has lead to great advances in robotics and biotechnology and has opened up many previously closed doors. One of the growing new fields in biotechnology is that of robotic prostheses.

As human life expectancy and life style increases in quality, the demand for high-tech prosthesis in society grows. Members of society with damaged limbs are socially accepted and many people are interested in helping their integration in modern life. This has motivated much research in the fields of biotechnology and robotics, which in turn have led to helping us understand more about how we work.

Prosthesis consist of two problems: movement problem and control problem. The movement problem consists in the kinematics and dynamics of the prosthesis to be able to replicate the functionality of the part it replaces and to assist in its implementation. The control problem consists in the translation of the biological signals to motion.

There are two control methods: neuromotive control or neurocognitive control. Neuromotive control consists in translating nerve impulses, which can be done by reading electromyography (EMG) signals; most modern prosthesis use neouromotive control. Neurocognitive control uses higher neural function. An example illustrating the differences would be telling your hand muscles to move to a certain position (neuromotive control) compared to the thought of grasping an object with your hand (neurocognitive control).



For fluid neuromotor prosthesis movement, the neural impulses have to be mapped to muscle movements [40]. This procedure is not simple and involves both being able to process the biological signals like EMG signals and being able to analyze the muscle movement in depth. This project attempts to aid in the mapping of EMG signals to the hand motion by simplifying the analysis of the hand motion.

Another application of the detailed study of hand movements is the identification of primitive hand motions also known as eigen-motions to simplify control strategies [35, 94]. Simplifying control allows for cheaper and more robust prosthesis and exoskeletons to be designed.

1.2 Objectives

The main objective is to be able to adapt a theoretical hand model to a real hand. This can be subdivided into three well defined parts:

- Design of a theoretical kinematic model of the hand.
- Detection of real hand characteristics with computer vision.
- Solver to adjust the theoretical model to the experimental data.

The design of the theoretical kinematic model for the hand is a prerequisite for the other two objectives which are independent among each other. By minimizing cross dependencies in the objectives, the reusability of the parts -in case they need to be changed or adapted to other projects- increases. This is important as the project aims to form part of a larger EMG signal mapping project.

Side objectives of the project also consist in analyzing hand motion for usage in hand exoskeletons, which would be robotic prosthesis mounted on the hand instead of substituting it. This could be useful in the case of individuals with not fully working limbs who do not wish to remove them to implant a prosthesis, who wish to augment normal hand functionality or in the case of rehabilitation after a stroke or other accidents. There is much research going into exoskeleton design for augmenting human capacities. This can be especially useful in manual jobs like construction work.

Overall there are many applications of understanding and being able to work with theoretical hand models that can be related to real world applications.



1.3 Scope

The focus of this thesis is presenting the algorithms, theory, implementation and results of the project. Basics and inner workings of the theory are out of the scope of this thesis. However all information is duly referenced and books to refer to for the theory behind equations are also clearly indicated.





2. Kinematic Hand Model

The hand is a fundamental element of human physiology. It allows humankind to grasp and manipulate any type of object. This chapter will focus on creating an anthropomorphically correct kinematic hand model based on modern anatomical studies.

2.1 State of the Art

The human hand has always been an important research topic in robotics due to its versatility. This has led to the development of many different robotic hand models. The main interest lies in the flexibility of the hand and its ability to grasp and manipulate objects with both power and precision [14]. Many robotic grippers based on simplified hand models have been designed, such as the DIST hand [9], four fingers and 16 degrees of freedom (DoF); the Gifu hand II [39], anthropomorphic with 18 DoF and force sensors and the hand developed at the Keio University [90], anthropomorphic with 20 DoF using elastic elements. There has also been theoretical work done in the area of underactuated hands by Gosselin [7, 41].

As the technology has progressed, more focus has been placed on creating prosthetic hands, which are robotic grippers subject to additional constraints like weight, size and surface finish. Examples of prosthetic hands are the NTU hand [34], the HIT/DLR prosthetic hand [33] and the hand developed at the Doshisha University [81]. These prosthetic hands are controlled by electromyography (EMG) signals. As with the robotic grippers, the focus is more on reproducing the human functionality while approximating anthropomorphism rather than on reproducing the human hand kinematics with high precision.

The thumb is recognized as the reason behind the success of the human hand. The joints of the thumb have a different arrangement, which has been studied to great detail [12, 93]. Work has also been conducted into designing robotic thumbs based on this information [10].



In adition, work has been done in creating anatomically correct hands like the ACT hand [84] which not only strives to replicate the exact kinematics of the hand, but also reproduce the entire bone structure.

2.2 Hand Anatomy

As an important part of human anatomy, the human hand has been subject to much medical study. However most of the medical descriptions do not serve our purpose as they deal with ailments of the hand or soft tissue. The work of Fadi J. Beijjani (1989) [22] shall be used as the reference for this section.

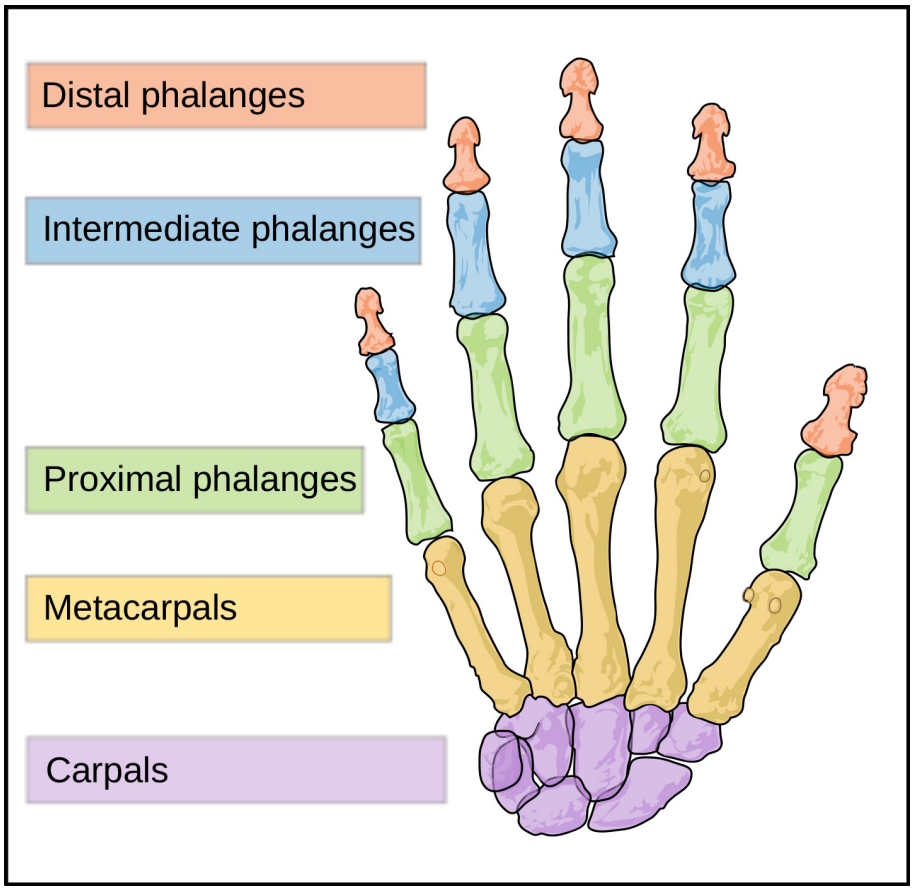


Figure 2.1: Bones of the human hand (Source: Wikipedia).

Figure 2.1 shows an overview of the bones of the hand. The hand has a total of 27 bones which are the best reference for decomposing the hand motion into primitive joints. The four fingers share a same joint structure, while the thumb has a slightly different one. These 27 bones form 14 joints with varying complexity.

For the purpose of this project the hand anatomy will be modelled up to and including the



wrist. This allows the hand model to do the full range of grasping motions.

2.2.1 Joints

While the location of the joints may seem like a trivial task with access to magnetic resonance imaging (MRI) and other high resolution scanners of the bone structure [77], in fact it is an extremely complex task. The joints between two bones in the hand do not generally form a single clean rotation axis, but two axes that may not even intersect. This is specially complex in the case of the thumb [11] and has lead to many studies.

The joints of the fingers are: carpometacarpal (CMC), metacarpophalangeal (MCP), distal interphalangeal (DIP) and proximal interphalangeal (PIP) joints. In the case of the thumb the joints are: CMC, MCP and interphalangeal (IP) joints; however, these joints do not behave exactly like their counterparts in the other fingers. These joints can have two movement types: flexion-extension (FE) and adduction-abduction (AA). The FE movement is on the sagittal plane while the AA movement is on the frontal plane. The wrist radio-ulna (RU) joint provides both FE and AA. For this project we'll also consider that the wrist can provide pronation-supination (PS) which is on the transverse plane.

Table 2.1: Degrees of freedom provided by the different joints.

Joint	DoF	Movement Types	Notes
CMC	1	FE	Finger joints only have 1 DoF.
CMC*	2	AA, FE	Thumb has an extra DoF.
MCP	2	AA, FE	
PIP	1	FE	Not found in thumb.
DIP	1	FE	Not found in thumb.
IP*	1	FE	Only found in thumb.
RU	3	AA, FE, PS	Wrist joint, common for all fingers.

The actual joint movements are more complex than what is depicted in Tab.2.1. However, this approximation will be valid for the purpose of this project.



2.2.2 Dimensions

The dimensions of the hand vary largely between individuals. However, the progression of each finger approximately follows the Fibonacci sequence as seen by the average interarticular lengths of the metacarpals (71 mm), proximal phalanges (46 mm), middle phalanges (28 mm), and distal phalanges (18 mm). As a basis for generating a robot hand model, the results from Tab.2.2 will be used [22].

Table 2.2: Average dimension of finger bones.

Bone	Index	Middle	Third	Fourth	Thumb
Proximal carpal (mm)	15	15	15	15	15
Distal carpal (mm)	13	13	12	8	2.236
Metacarpal bone (mm)	43	43	38	40	25
Proximal phalanx (mm)	30	35	33	24	20
Middle phalanx (mm)	20	26	25	20	-
Distal phalanx (mm)	18	18	16	15	16
Capitate to long axis (mm)	11	0	8	19	13

2.2.3 Movement

One of the most variable features of the hand across human populations and also among the individuals is the range of rotation of each joint. This not only depends on the individual, but also upon the joint laxity. It is most notable beyond the 0°point in the negative direction. For the purpose of this project, exact measurements are not needed. An estimation of the joint rotation range can be found in Tab.2.3.

2.3 Robotic Hand Model

One of the major differences between the robotic hand model and the real model is the lack of soft tissue in the robotic hand model. This simplifies the design greatly, however it can introduce modelling error. Soft kinematics are still at a very early stage of development.



Table 2.3: Joint rotation range.

Joint	Index	Middle	Third	Fourth	Thumb [11]
CMC-AA	-	-	-	-	20°
CMC-FE	-	-	20°	20°	20°
MCP-AA	20°	20°	20°	20°	20°
MCP-FE	70°	80°	90°	95°	90°
PIP-FE	100°	100°	100°	100°	-
DIP-FE	90°	90°	90°	90°	-
IP-FE	-	-	-	-	95°
RU-FE	100°	100°	100°	100°	100°
RU-AA	80°	80°	80°	80°	80°
RU-PS	160°	160°	160°	160°	160°

2.3.1 Joint model with D-H Parameters

There are many kinds of joints, however in the case of the hand model, they can all be modelled by revolute joints. They can be defined by using Denavit-Hartenberg (D-H) parameters [15]. D-H parameters are a way of representing reference frames for an articulated system. This methodology is widely used in robotics and is a minimal representation method.

The D-H convention consists of representing any serial chain as a set of translations and rotations along the X and Z axes as seen in Fig.2.2. These transformations define the reference at each axis. Each transformation always goes through the common normal and has a set of 4 parameters:

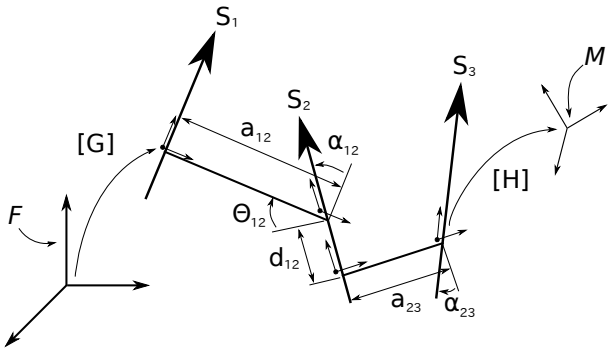


Figure 2.2: Visual representation of D-H parameters.

θ Rotation around Z axis from previous common normal to next common normal.

d Offset along Z axis from previous common normal to next common normal.

α Rotation around X axis from previous common normal to next common normal.

a Offset along X axis from previous common normal to next common normal.

There are various ways of concatenating the rotations and translations. The method used in this project is to first do the Z axis rotation and translation and then do the X axis rotation and translation using homogeneous matrix math,

$$Z(\theta,d)=\left[\begin{array}{ccc|c}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & d\\ \hline 0 & 0 & 0 & 1\end{array}\right]$$

$$X(\alpha,a)=\left[\begin{array}{ccc|c}1 & 0 & 0 & a\\ 0 & \cos\alpha & -\sin\alpha & 0\\ 0 & \sin\alpha & \cos\alpha & 0\\ \hline 0 & 0 & 0 & 1\end{array}\right]$$

$$[T^n]=[Z(\theta_n,d_n)][X(\alpha_n,a_n)] \tag{2.1}$$

A full kinematic chain with n joints can be expressed by an initial transformation $[G]$ followed by the successive transformations from axis to axis using D-H parameters as seen in Eqn.(2.1) complete with a final transformation $[H]$ to the end effector,

$$\begin{aligned}[D] =&[G][T^1][T^2]\cdots[T^n][H]\\ =&[G][Z(\theta_1,d_1)][X(\alpha_1,a_1)]\cdots[Z(\theta_n,d_n)][X(\alpha_n,a_n)][H]\end{aligned} \tag{2.2}$$

Full information on the D-H convention can be found in [29].

By following the bone structure these parameters can be calculated from Tab.2.2. The joints with 2 DoF (CMC*, MCP) shall be considered to have both axes of rotation intersecting. Tables 2.4, 2.5 and 2.6 show the D-H parameters obtained from the bone layout from Fig.2.1 using the dimensions from Tab.2.2. The translations are in mm and the rotations in radians.



Table 2.4: D-H Common Parameters.

D-H	Wrist FE	Wrist AA	Wrist PS
θ	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$
d	0	0	0
α	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
a	0	0	0

2.3.2 Degrees of freedom

In the case of the index and middle finger, the CMC-FE is very small and for the purpose of constructing a robot model of the hand it will not be considered. This leaves the index and middle fingers with only 4 DoF while the middle, third and thumb fingers all have 5 DoF. This simplification lowers the total DoF by 2 and is widely accepted by the scientific community. The wrist provides an additional 3 DoF.

2.3.3 Model

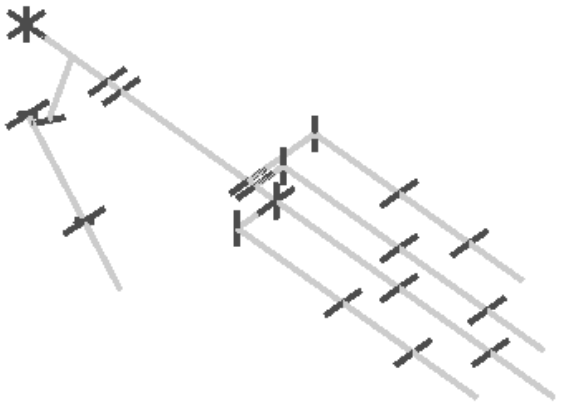


Figure 2.3: Computer model of the human hand.

The complete model can be seen in Fig.2.3 using a full 26 DoF. This model matches with the generally accepted models used for the human hand [62]. It is important to note how the model has a strong similarity with the bone structure from Fig.2.1. The representation of each figure is done by rendering the joint axes and the common normals between the joint axes.

Table 2.5: D-H Finger Parameters.

Joint	D-H	Index	Middle	Third	Fourth
CMC-FE	θ	π	π	π	π
	d	71	71	65	63
	α	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
	a	0	0	0	0
MCP-AA	θ	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
	d	-11	0	8	19
	α	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
	a	0	0	0	0
MCP-FE	θ	0	0	0	0
	d	0	0	0	0
	α	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
	a	30	35	33	24
PIP-FE	θ	0	0	0	0
	d	0	0	0	0
	α	0	0	0	0
	a	20	26	25	20
DIP-FE	θ	0	0	0	0
	d	0	0	0	0
	α	0	0	0	0
	a	0	0	0	0
TCP	θ	0	0	0	0
	d	0	0	0	0
	α	0	0	0	0
	a	18	18	16	15



Table 2.6: D-H Thumb Parameters.

D-H	CMC-AA	CMC-FE	MCP-AA	MCP-FE	IP-FE	TCP
θ	-0.4636	$\frac{\pi}{3}$	0	0	0	0
d	13	5	0	0	0	0
α	$\frac{\pi}{2}$	$\frac{110\pi}{180}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	0
a	15	-0.2236	0	25	0	20



3. Detection of Hand Characteristics

The theoretical hand model developed in Chapter 2 is used as a basis to adjust for a real hand. An accurate hand model, adapted to the dimensions of a particular individual, has two important applications: one is the accurate description of the anatomy of the individual, the other one is to have a correct kinematic structure in order to perform accurate hand motion and joint angle tracking. To perform this sizing of the kinematic skeleton, we use computer vision to obtain the input data. This chapter will focus on obtaining the task poses to create the kinematic model.

3.1 State of the Art

There are two general approaches to hand detection: contact and non-contact [85]. Contact approaches consist in mounting a device to the hand that can capture the poses as it moves. Examples include the AcceleGlove [31], the VPL DataGlove [32] or the Rutgers Master II [8]. However all gloves have at least one major drawback: low portability, high cost, need for calibration or low resolution. The main advantage to contact approaches is that they are well suited to realtime tracking and can provide large amounts of data. A more detailed review on modern contact-based hand detection can be found in [16].

Non-contact technologies are generally vision-based although other devices like magnetic resonance imaging (MRI) scans have been used [77]. Most of the focus in single camera approaches has been with detecting the region of interest (ROI) of the hand [5, 91]. Some have attempted to create simplified hand models through markerless detection [27, 44]. Work has also been done using markers [4] and multi camera systems [80]. There are a variety of different methods. However, many approaches focus on realtime tracking and not precision for application in the growing field of augmented reality. For a more complete survey of human motion capture refer to [56] and [21] for the hand.



3.2 Approach

The hand is covered with skin which is a flexible soft tissue that is constantly being deformed. Deformable materials are an important area of study [71] in computer vision, however they complicate tracking as the deformation must be taken into account. The skin of the hand also generally lacks noticeable features which also makes it harder to track as a deformable surface. The only visible fixed exterior parts of the hand are its fingernails. However fingernails are hard to track as they are contoured surfaces with different shapes for different individuals. It is important for this project to be able to obtain reliable positions and orientations of the fingernails to estimate the kinematic hand model.

3.3 Markers

An important aspect of the computer vision system is the geometry of the markers. The geometry of a marker affects directly its performance and usability in computer vision applications. The markers are used to estimate the pose, which consists of the position and orientation, of each fingernail. Fingernails are not subject to elastic deformation and thus can be used to estimate the kinematics of the hand.

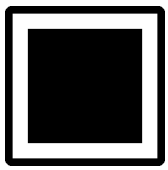


Figure 3.1: Marker template.

There are many designs of markers [61]; however, most are focused on storing data. This makes them more complex and harder to detect. By using more simple plain markers that just express geometry it is easier to have a more robust detection, especially when the marker is visually small in the image. This is important because the markers represent a small part of the hand and have to be detectable when 5 markers are moving around simultaneously.

The design used by this project is a simple white square with a smaller black square inside as seen in Fig.3.1. This gives it 4 sharp visible corners which form a perfect square that can be used to find the 3D pose of the marker. Figure 3.2 shows a properly detected marker. It is important for these markers to be completely rigid for accurate pose estimation.



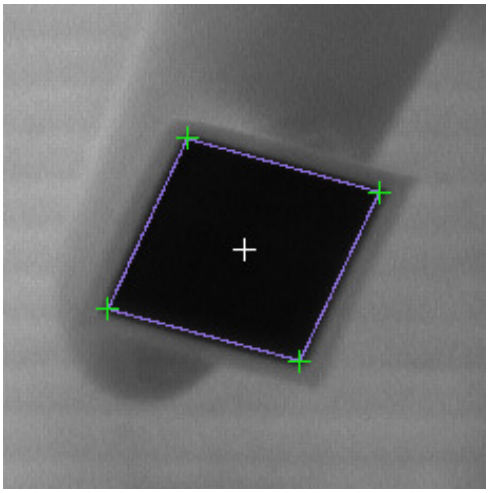


Figure 3.2: A detected marker.

3.4 Pose Estimation Problem

The combination of position and orientation is called a pose. The pose estimation problem or PnP consists in identifying the pose of the camera given n 2D-3D correspondences and the camera internal parameters [46]. This finds the transformation from the camera coordinates to the object. It is an important problem in computer vision [87].

More specifically the full transformation can be written as,

$$\mathbf{u} = [A][R|\mathbf{t}]\mathbf{p} = [P]\mathbf{p} \tag{3.1}$$

where $[A]$ is the 3x3 internal calibration matrix [30] and $[R|\mathbf{t}]$ is the 3x4 transformation matrix composed of a 3x3 rotation matrix $[R]$ and a 3x1 translation vector $\mathbf{t}$. The matrix $[A]$ of internal parameters is known and may be written as,

$$[A] = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where $a_x = f \cdot m_x$ and $a_y = f \cdot m_y$ with f being the focal length and m_x, m_y being the x and y scale factors respectively. The principal point or image center is provided by u_0, v_0.

Given the matrix of internal parameters $[A]$ and a set of 3D-to-2D correspondences $\{\mathbf{u}_i \leftrightarrow \mathbf{p}_i\}$, the goal is to retrieve $[R]$ and $\mathbf{t}$. The generalized problem is very complex [45], however in

the case of the markers chosen it is reduced to a 4 points in coplanar configuration simplifying the problem greatly.

3.5 Algorithm

The objective of the algorithm is to obtain a set of task positions from a video stream of a hand moving with markers on its fingernails. An overview of the algorithm can be seen in Fig.3.3.

The detection algorithm consists of 6 steps:

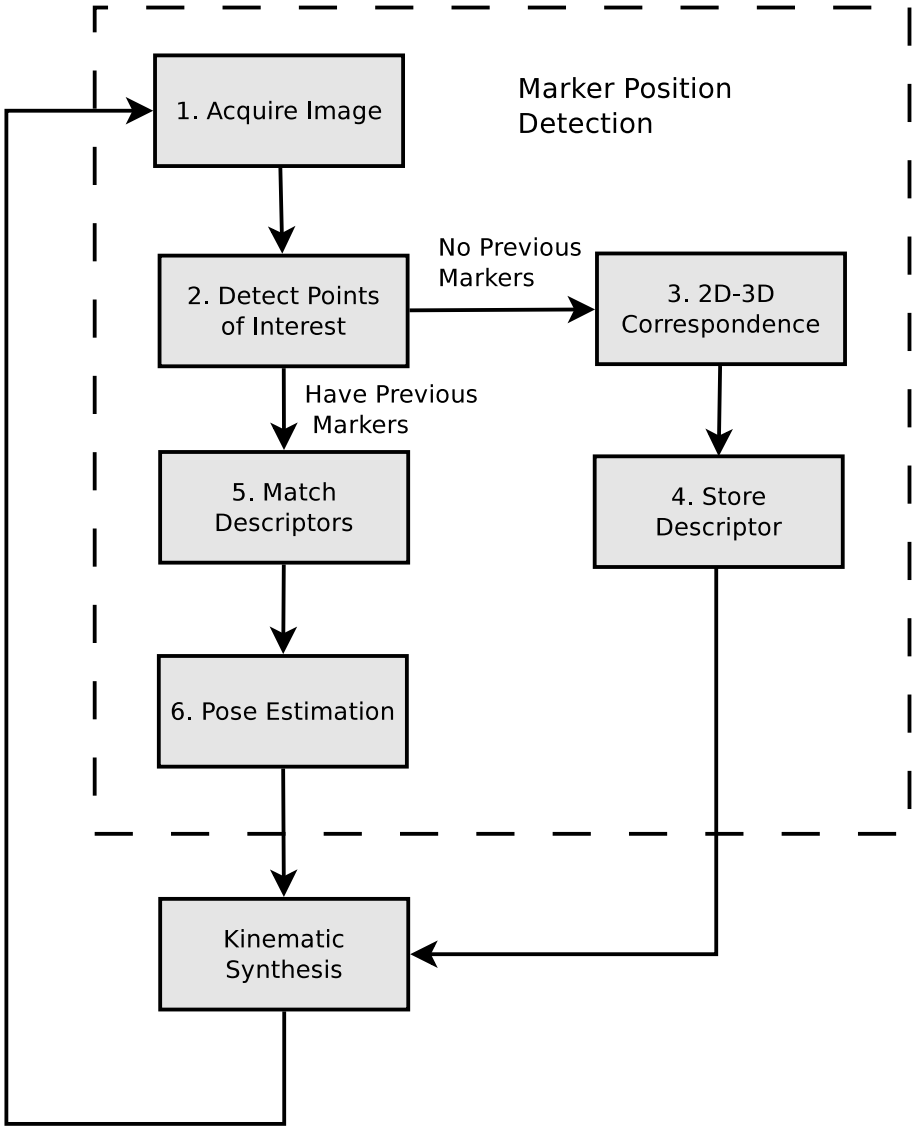


Figure 3.3: Flux diagram of the detection algorithms

1. Acquire Image Obtains an image from the video stream.



2. **Detect Points of Interest** Detects the points which are candidates for belonging to a marker.

3. **2D-3D Correspondence** Tries to form the markers using heuristics.

4. **Store Descriptor** Stores the descriptors of the detected markers.

5. **Match Descriptors** Tries to match the descriptors with the points of interest to find the pose of the markers.

6. **Pose Estimation** Estimates the pose from the 2D marker positions.

3.5.1 Acquiring Images

High resolution and low noise images are fundamental for the precise detection of the markers. Since the estimation is based on arbitrary task positions, movement is not important when acquiring images. However if the camera is very slow, there can be problems with motion blur. It is therefore recommended to do slow movements with high resolution cameras.

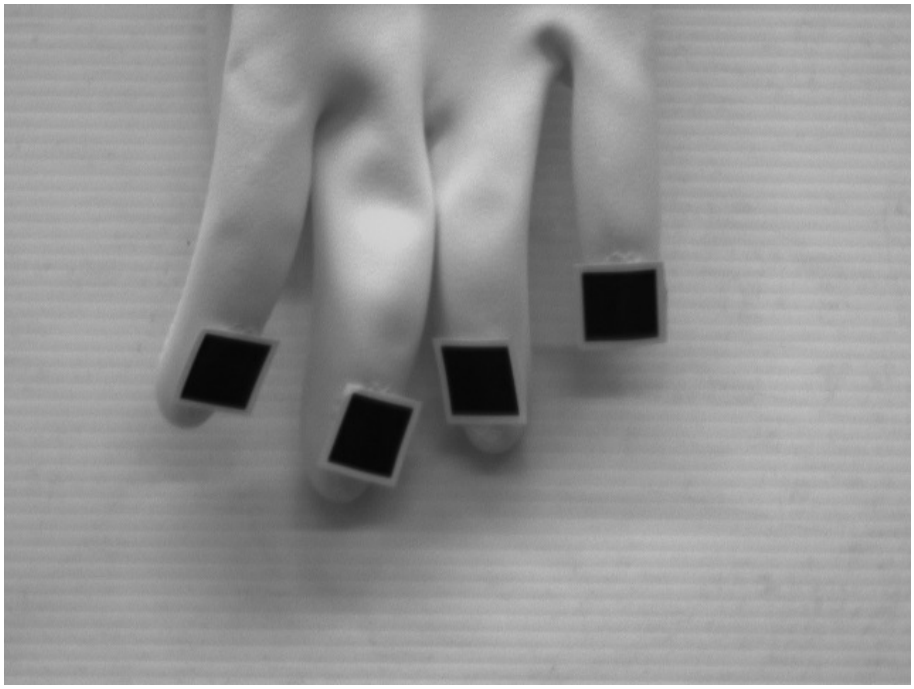


Figure 3.4: An image captured by a Flea® 2 camera directly in grayscale.

The cameras used in this project are the Point Grey Flea® 2 [23] which provide a resolution of 1288x964 and a frame rate of 30 FPS. The image is directly obtained in 8BPP gray scale which lowers the needed bandwidth and helps to speed up further calculations.

3.5.2 Points of Interest

The main goal of the points of interest is to reduce the processing time of each frame. By detecting possible candidates of marker corners, the calculations done later on are simplified. The Harris corner and edge detector [28] is a superb way to detect the candidate points.

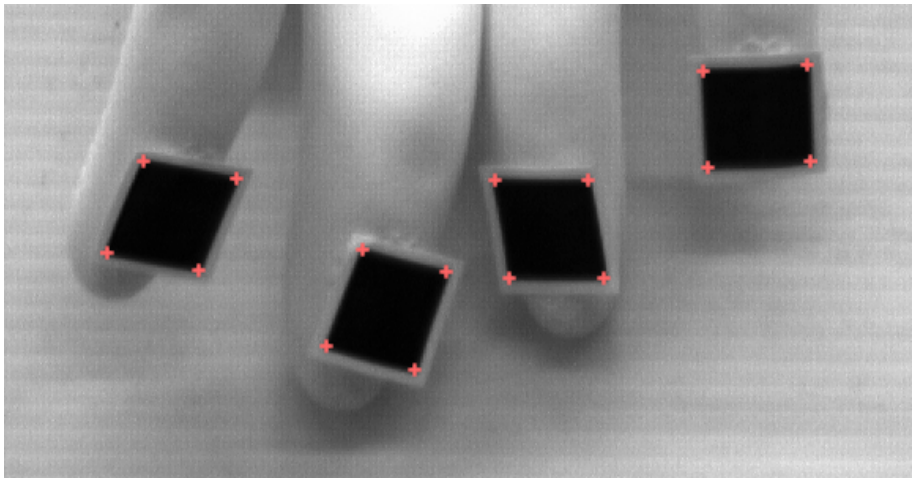


Figure 3.5: Points of interest detected in an image.

The Harris corner and edge detector relies on the fact that, at a corner, the image intensity changes strongly in multiple directions. It is done with an approximation of the first gradients of the intensity to speed up computational time. To make the detector more immune to noise [6], the image is run through a Gaussian filter to smooth the pixel noise and reduce the amount of false positives. Figure 3.5 shows points of interest detected. It is important to have controlled lighting and background to reduce the amount of points of interest found.

3.5.3 Solving 2D-3D Correspondence

Before having descriptors for each marker corner, the markers have to be matched to the 2D points of interest. This is a complex issue because the points of interest have to be grouped into possible markers out of which the best has to be chosen through heuristics.

The heuristic approach to detect markers is slow because it must iterate over many combinations. The number of iterations n_i needed for a set of p points can be calculated by,

$$n_i = \frac{p!}{(p-4)!} \tag{3.3}$$



With $p = 50$ the amount of iterations is $n_i = 5,527,200$ which makes this algorithm very slow due to its complexity $O(2^{n \log n})$. It is therefore important to try to discard as many points as possible when doing corner detection.

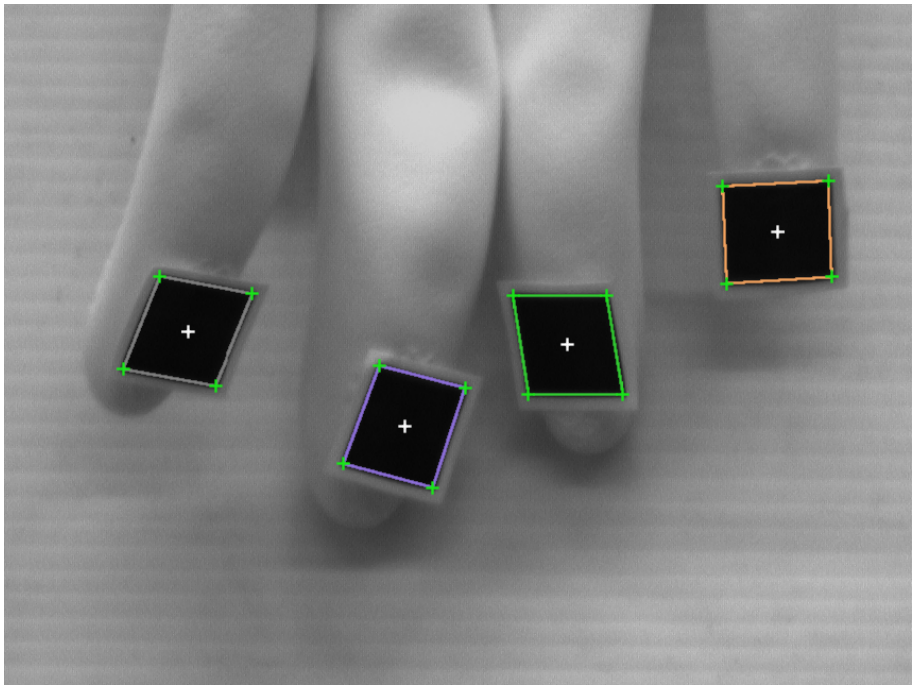


Figure 3.6: Markers created from points of interest using heuristics.

The heuristics are designed to first create optimal sets of 4 points as shown in Alg.1. This algorithm tries to drop as many combinations as possible to simplify posterior calculations. These sets of 4 points are then grouped into a group of 5 sets. Each set of 4 points represents a marker and the group of 5 sets represents all five markers. It is therefore important for the first image to have visible and flat markers. Once the markers are found, it no longer uses a heuristic approach but can rely on the descriptors from Sec.3.5.4.

3.5.4 Descriptors

Descriptors are a way of mapping points on an image to a function that gives them a unique value, allowing them to be compared to points on other images to see if they could be the same point. This allows the comparison of candidate points between frames to track them. There are many different types of descriptors and they all generally use histograms. This project uses the DAISY descriptors [83] which have proven to be very robust and fast.

DAISY descriptors are similar to SIFT [48] and GLOH [54] as they all depend on histograms of gradients; however, they use a Gaussian weighting and circularly symmetrical kernel. The

Input: Set of points of interest

Output: 5 groups of 4 points representing markers

$markers := \emptyset$

forall $m \in permutation(p)$ **do**

if $m \in markers$ **then**
continue

end

$angle_{error} := |angle(m.l[2], m.l[1]) - angle(m.l[2], m.l[4])| + |angle(m.l[4], m.l[1]) -$

$angle(m.l[2], m.l[3])|$

if $|angle_{error}| > \frac{\pi}{3}$ **then**
continue

end

$perimeter := \sum_{i=1}^{4} m.l[i]$

$perimeter_{avg} := \frac{1}{4} \sum_{i=1}^{4} m.l[i]$

$perimeter_{error} := \sum_{i=1}^{4} |m.l[i] - perimeter_{avg}|$

if $perimeter_{error} > 1.5 perimeter_{avg}$ **then**
continue

end

$area := marker_area(m)$

if $area < 50^2$ *or* $area > 150^2$ **then**
continue

end

$roundness := \frac{4\pi area}{perimeter^2}$

if $roundness < 0.436$ **then**
continue

end

$m.fitness := 2roundness + 1.5\frac{perimeter_{error}}{perimeter_{avg}} - 0.5\frac{angle_{error}}{\pi}$

$markers.add(m)$

end

return $best(markers, 4)$

Algorithm 1: Heuristics used for estimating markers for the first time.



shape of the kernel is what gives the descriptor its name. The descriptor is designed for dense matching; however, its performance on individual points is also remarkable.

3.5.5 Solving the Pose Estimation Problem

Once the four corners that form each marker are obtained, the marker pose must be estimated. This forms the pose estimation problem for the specific case of 4 points known as P4P. The P4P problem with coplanar model points has a single unique solution [92]. The solution can be found by using the Orthogonal Iteration (OI) [49] algorithm to approximate the pose. The estimation can be used as a base and be refined by with the Robust Pose Estimation algorithm [72].

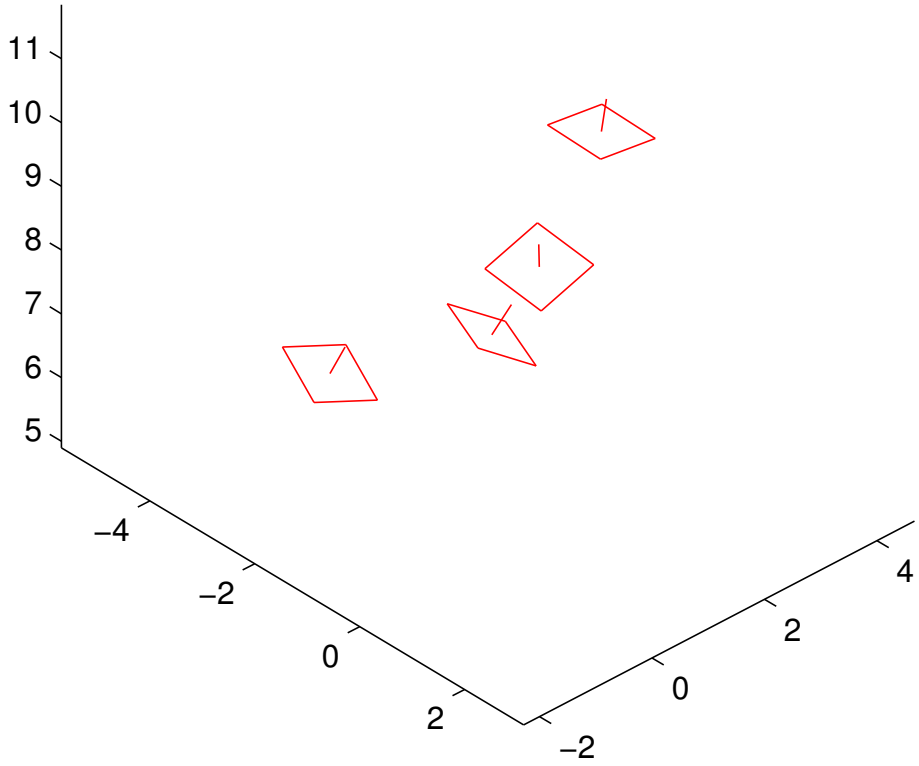


Figure 3.7: Reconstructed poses from markers.

Results of the pose estimation of 4 markers can be seen in Fig.3.7. However, the pose estimation is not fully accurate and generally will have error. The pose estimation error is analyzed in Sec.3.6.

Orthogonal Iteration

The Orthogonal Iteration algorithm is an iterative algorithm for estimation the absolute orientation problem. It is a replacement for older methods using optimizers like the Gauss-Newton



method or the Levenberg-Marquadt method. The algorithm is fast and also converges globally.

Robust Pose Estimation

The Robust Pose Estimation algorithm attempts to improve the estimation by resolving pose ambiguity. This is caused by the fact that there are generally multiple minima when estimating the absolute orientation problem. With an ideal pose there is either one minimum or two local minima; However, when noise from detecting the 2D-3D correspondence is added, there may be even more minima. The correct pose is generally the absolute minima. This algorithm is slower than the Orthogonal Iteration algorithm, but provides better results.

3.6 Reliability and Error

One of the important aspects of computer vision is being able to control and work around the possible errors that can appear from the entire process of taking the image and processing it. Computer vision is not an exact science and therefore there is a lot of variability and error in all stages of the processing. Focus will be given to the error in the pose estimation.

3.6.1 Marker Simulation

The first test is done by projection of a marker pose on to the camera plane. The objective is to see how error propagates from the 2D projection on the camera plane to the estimation of the original pose. The projection positions are then subject to Gaussian noise which affects the pose estimation. The Gaussian noise has the position as its mean and varies in standard deviation which is used as a simulation parameter. Afterwards the projection positions are used to estimate the original object's pose. Finally the estimated pose is compared with the original pose. This can then be used to see what error to expect depending on the environment noise.

The noise represents the accumulation of error from various sources:

- Quantization of the image plane
- Distortion caused by approximating camera parameters
- Motion blur



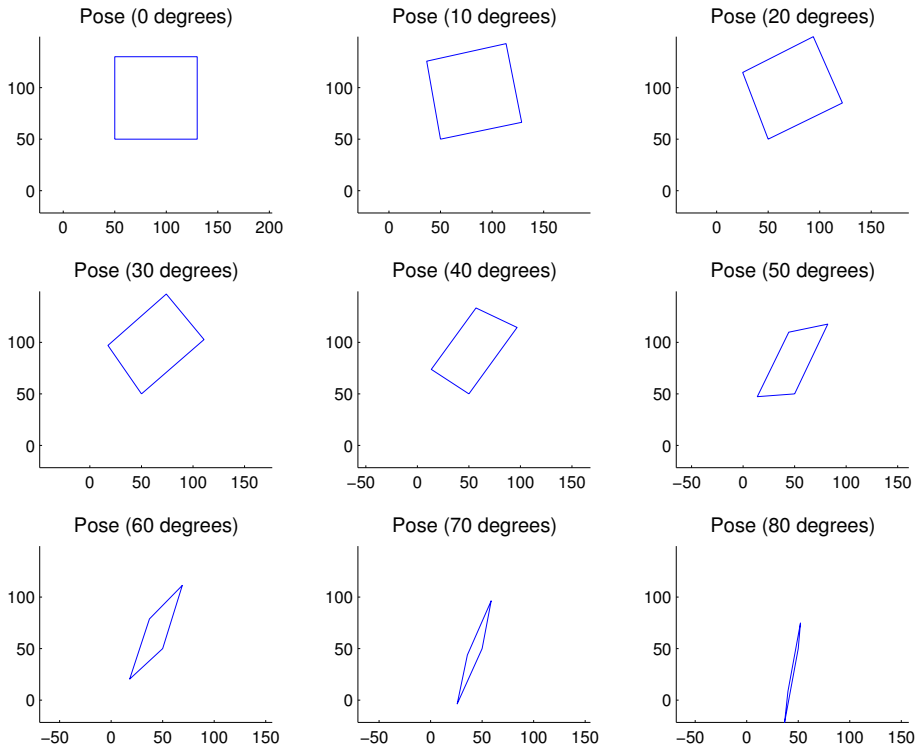


Figure 3.8: Projection of Simulated Marker

- Image noise

As simulating the exact behaviour of the different sources of noise is very complex, they are all treated as a single source of noise that follows the Gaussian distribution.

The test marker poses can be seen in Fig.3.8. They are obtained from the rotation of a marker with the same angle around the three Cartesian axes. It is designed so that the image with a 0° tilt contains a 100x100 pixel marker, similar to what will be seen on the camera. The poses are representative of different situations in which a marker is visible. Noise is added on each corner "pixel" in the image following Gaussian distribution. The standard deviation of the Gaussian noise is used as a simulation parameter.

3.6.2 Error Analysis

Two different errors will be studied: translation error and rotation error. Translation error is the error of the center position of the marker and rotation error is the error of the normal vector of the marker. The rotation error is more dangerous to the system due to the fact that it causes important propagations down the entire kinematic chain. Figure 3.9 shows the translation error while Fig.3.10 shows the rotation error using the combined Orthogonal Iteration and Robust



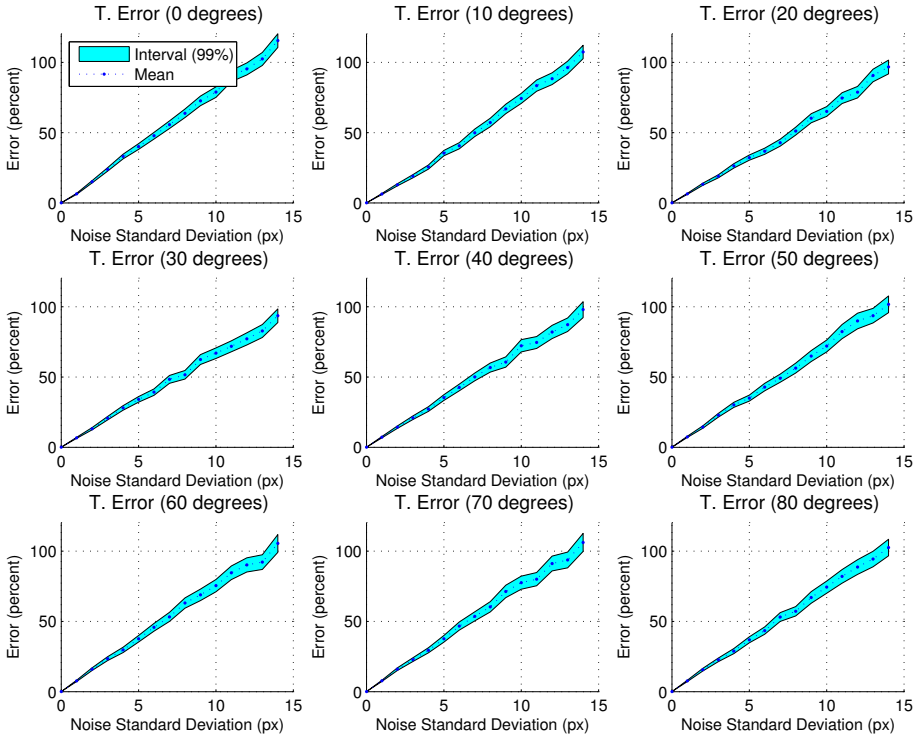


Figure 3.9: Simulation of Translation Error

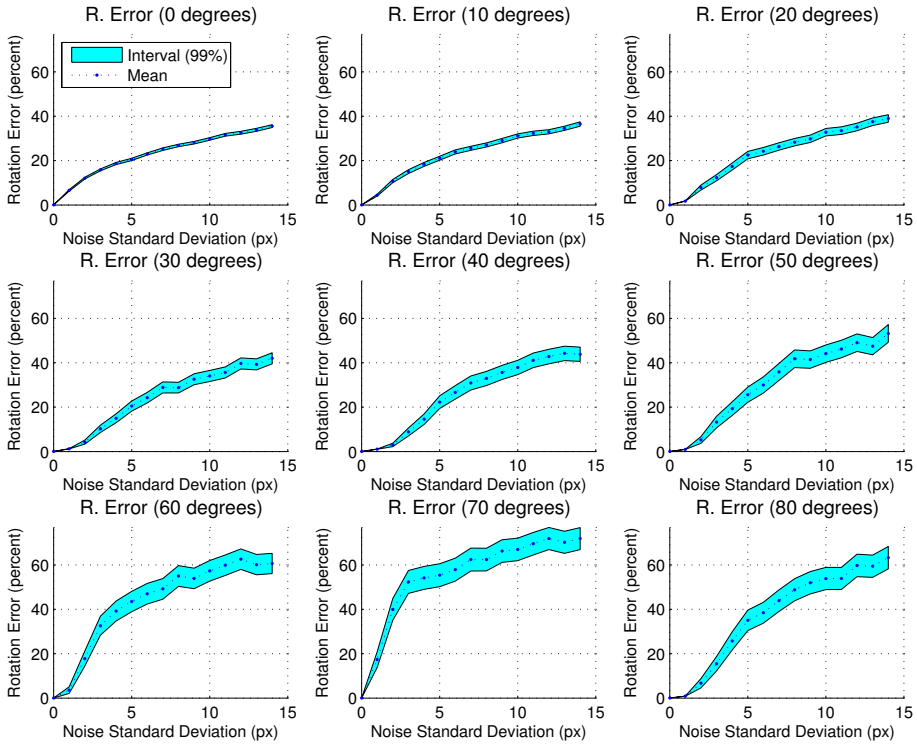


Figure 3.10: Simulation of Rotation Error



Pose Estimation algorithms.

It is important to note that the translation error is nearly linear while the rotational error increases quickly. Therefore it is very important to minimize error by using higher resolution cameras with less noise to get reliable results and minimize rotational error.

For the set up used in this project an error of 0-3 pixels can be expected. This gives an acceptable error except in the case of the rotation error when the marker is close to perpendicular to the camera plane. This can be avoided by filtering the movement of the markers with a Kalman filter [38] or by increasing the number of frames used by the solver.

3.7 Experimental Dataset

The sets of poses obtained from the hand can then be used as an experimental dataset for the solver explained in Chapter 5 to estimate the kinematic hand model of the real hand. The processed frames do not have to form a continuous animation. However, it is important for them to be of as high quality and as noise free as possible so the error does not propagate into the solver. Thus it is important to use for the experimental dataset the best images less likely to be affected by error and that are as different as possible among themselves. All joints should have as much movement as possible to ensure they get properly adjusted and to minimize error.





4. Kinematic Synthesis

The generalized inverse kinematics problem consists in the dimensional fitting of a robot to move through a series of task positions. It involves changing structural parameters of the kinematic models to solve the problem. This includes not only the movements of the joints, but also the position and orientation of the joints. This chapter will deal with the definition of the motion-to-form or dimensional kinematic synthesis problem. It will establish the theoretical base and define the design equations to be able to adjust the theoretical model defined in Chapter 2 to a real hand dataset detected as explained Chapter 3 It is not meant to be an authoritative guide on the subject, for an authoritative guide refer to the works by McCarthy (1990) [51] and Selig (2004) [73].

4.1 State of the Art

The general inverse kinematics problem is generally an unsolved problem. Much research has been done in the past decade on the topic [52]. Methodology has been developed using polynomial homotopy continuation, Gröebner bases, elimination [59] and linear product decomposition [53].

During this time many different spatial serial chains have had their generalized inverse kinematics analyzed. These serial chains are generally formed by revolute (R), spherical (S), prismatic (P) and universal (T) joints. Some examples of chains solved are the RPS [78], PRS [79], RR [68], RRP, RPR and PRR [66].

However more complex serial chains, especially those formed exclusively by R joints, have not been fully analyzed. For example the RRR or 3R joint formed by three revolution joints has been studied only in a predetermined range where 13 real solutions have been found [43]. This is because the complexity increases greatly. More complex systems have been studied [69] but not as complex as the model from Chapter 2 nor as in detail as the simpler systems.



4.2 Dimensional Kinematic Synthesis

Kinematic synthesis deals with the motion-to-form problem. Given a kinematic task, it calculates the set of articulated bodies able to perform that task. In general there are two types of synthesis: type synthesis and dimensional synthesis.

Type synthesis deals with selecting or computing the number and type of joints for the set of articulated bodies. For this project this is already defined by the form of the hand and has been presented in Chapter 2.

Dimensional synthesis performs the sizing of the articulated system. This consists of calculating all the geometric dimensions of the system. This project will focus on dimensional synthesis to fit the hand model to the data provided by computer vision.

4.3 Kinematic Chain

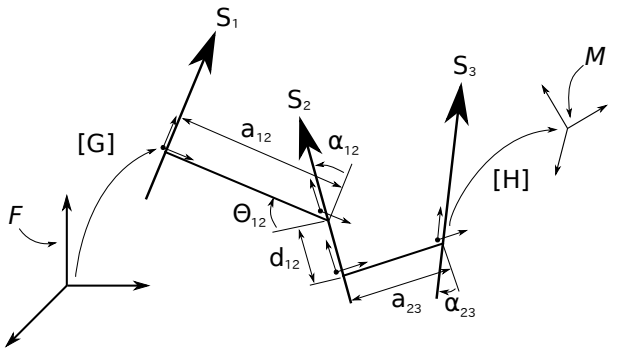


Figure 4.1: Serial kinematic chain.

A kinematic chain is a combination of joints representing an articulated system and the relationships between the joints. A kinematic chain can have various topologies depending on how they're connected. Figure 4.1 shows a simple 3 joint kinematic serial chain.

The topology of the hand is a bit more complicated. It consists of 5 fingers that share 3 common axes. It can be represented with a tree-like kinematic chain consisting of a 3 common axis kinematic serial chain connected to five kinematic serial chains arranged in parallel. A tree-like kinematic chain is represented in Fig.4.2.

If all the joints are revolute joints, they can be expressed as nR where n is the number of joints in the serial kinematic chain.



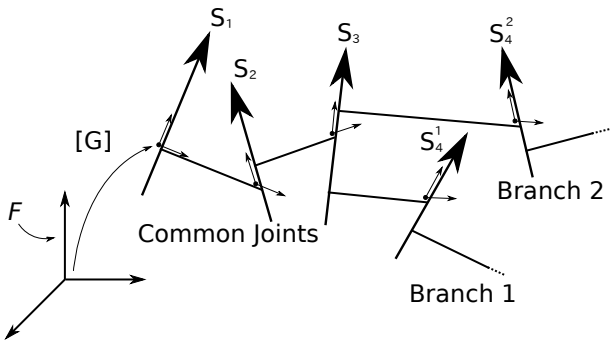


Figure 4.2: Tree-like kinematic chain.

4.4 Screws

Chasles' Theorem 1. *All proper rigid body motions in 3-dimensional space, with the exception of pure translations, are equivalent to a screw motion, that is a rotation about a line together with a translation along the line [73].*

To understand the mathematics in this chapter it is important to know the basics of screw theory. Screw theory is a conceptual framework useful for its application in kinematics. Any rigid body transformation (rotation and translation) can be expressed as a screw displacement: a rotation and translation along an axis as stated by Chasles' Theorem (1830). This axis is called the screw axis and can be seen in Fig.4.3. The combination of two screw displacements gives another screw displacement. This is one of the fundamental concepts of kinematics.

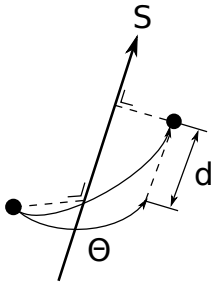


Figure 4.3: A screw displacement.

Screw displacements can be written in many ways. A screw displacement around the vector $\mathbf{v}$, that passes through the origin, with a rotation θ and translation p along the axis can be written as the 4x4 homogeneous matrix,

$$[A(\theta)] = \begin{bmatrix} [R] & \frac{\theta}{2\pi} p\mathbf{v} \\ 0 & 1 \end{bmatrix} \tag{4.1}$$



The rotation matrix $[R]$ is a θ rotation around the axis with the orientation $\mathbf{v}$. In other words $\mathbf{v}$ is an eigenvector of $[R]$ and thus $[R]\mathbf{v} = \mathbf{v}$. In general the screw axis does not go through the origin, but through a point $\mathbf{u}$. This can be written as,

$$\begin{bmatrix} [I] & \mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} [R] & \frac{\theta}{2\pi}p\mathbf{v} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} [I] & -\mathbf{u} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} [R] & \frac{\theta}{2\pi}p\mathbf{v} + ([I] - [R])\mathbf{u} \\ 0 & 1 \end{bmatrix} \tag{4.2}$$

4.5 Quaternions

Quaternions were first described by Sir William Rowan Hamilton in 1843. They can be written as,

$$\hat{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{4.3}$$

The algebra of quaternions is also known as the Hamiltonian algebra $\mathbb{H}$. The fundamental formula of quaternion multiplication can be denoted by,

$$i^2 = j^2 = k^2 = ijk = -1 \tag{4.4}$$

Unit quaternions which comply with $\hat{q}\hat{q}^* = 1$ are isomorphic to the special orthogonal group $SO(3)$ and can be used to represent 3D rotations.

4.6 Dual Unit

The dual unit ϵ is an extension to real numbers that is nilpotent ($\epsilon^2 = 0$). The ring of dual numbers can be represented by $\mathbb{D}$. The dual numbers are an alternate complex plane that complements the ordinary complex plane $\mathbb{C}$. The dual unit can be used to extend other algebras.



4.7 Plücker Coordinates

Lines in geometry are generally represented by a point $\mathbf{c}$ and a unit vector $\mathbf{s}$ multiplied by a value $u \in \mathbb{R}$,

$$\mathbf{c} + u\mathbf{s} \tag{4.5}$$

This leads to infinite representations of the same line as the point $\mathbf{c}$ can be any point contained by the line. The unit vector $\mathbf{s}$ can also have two representations, one for each direction along the same orientation. To reduce the number of representations of a line the moment of the line $\mathbf{s}^0$ can be calculated as,

$$\mathbf{s}^0 = \mathbf{c} \times \mathbf{s} \tag{4.6}$$

For a given point $\mathbf{c}$ and a given unit vector $\mathbf{s}$ there is only a single representation of $\mathbf{s}^0$. This now leaves two representations of the same line, corresponding to both directions of the unit vector $\mathbf{s}$. The representation of a line by its orientation and moment is called a Plücker coordinate and was introduced by Julius Plücker in the late 19th century. The dual unit ϵ can be used to represent the Plücker coordinate as the dual vector,

$$\mathsf{S} = \mathbf{s} + \epsilon\mathbf{s}^0 = \mathbf{s} + \epsilon\mathbf{c} \times \mathbf{s} \tag{4.7}$$

4.8 Clifford Algebra

Clifford algebras are associative algebras characterized by the Clifford product.The Clifford algebra is defined by the generators $\mathbf{e}_i$ used to construct it. Different authors have used different generators which lead to different notations. This Chapter will use the notation used by McCarthy (1990) [51] and Selig (2004) [73]. It is important to note that the bases used in these two books are not exactly the same but with a simple name change they end up working the same.

The Clifford algebras can be written as $Cl(p, q, r)$ for a Clifford algebra with p generators that square to +1, q generators that square to -1 and r generators that square to 0. The simplest Clifford algebra besides the trivial case is $Cl(0, 1, 0)$ that is isomorphic to complex numbers $\mathbb{C}$



with the form $x + y\mathbf{e}_1$. This can be seen easily as $\mathbf{e}_1^2 = -1$, as the only generator in the algebra must square to -1. Another simple Clifford algebra is $Cl(0, 0, 1)$ which by similar arguments is the ring of dual numbers $x + y\mathbf{e}_1 = x + y\epsilon$.

The dimension of the algebra generated by $n = p + q + r$ elements will be 2^n. An example would be the Clifford algebra $Cl(0, 2, 0)$ which is isomorphic to quaternions. An element of this algebra has the form $w + x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3 = w + xi + yj + zk$ which is of dimension 4.

The algebra can also be broken into even and odd degrees subspaces represented by $Cl^+(p, q, r)$ and $Cl^-(p, q, r)$ respectively. The even part forms a subalgebra, as the product of even degree monomials is always even. The monomials of this subalgebra shall be represented by $\mathbf{e}_i\mathbf{e}_j = \mathbf{e}_{ij}$. The dimension of this subalgebra is $2^{p+q+r-1}$ and it is isomorphic to a Clifford algebra with one more generator:

$$Cl(p, q, r) = Cl^+(p, q + 1, r) \tag{4.8}$$

More information on Clifford algebra can be found in [73].

4.8.1 Dual Quaternions

Dual quaternions are elements of the Clifford subalgebra $Cl^+(0, 3, 1) = \mathbb{H} \otimes \mathbb{D}$. Researchers working with dual quaternions have used different basis which may lead to confusion, for this project the basis chosen is the one used by McCarthy (1990) [51],

$$\{1, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}, \mathbf{e}_{41}, \mathbf{e}_{42}, \mathbf{e}_{43}, \mathbf{e}_{1234}\} = \{1, i, j, k, i\epsilon, j\epsilon, k\epsilon, \epsilon\} \tag{4.9}$$

These bases generate the multiplication table Tab.4.1. A dual quaternion can be written as,

$$\begin{aligned} \hat{Q} = \hat{q} + \epsilon\hat{q}^0 &= (q_0 + q_1 i + q_2 j + q_3 k) + \epsilon(q_7 + q_4 i + q_5 j + q_6 k) \\ &= q_0 + \mathbf{q} + \epsilon(q_7 + \mathbf{q}^0) = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_0 \end{Bmatrix} + \epsilon \begin{Bmatrix} q_4 \\ q_5 \\ q_6 \\ q_7 \end{Bmatrix} \end{aligned} \tag{4.10}$$



Table 4.1: Multiplication table for $Cl^{+}(0,3,1)$ (McArthy bases [51]).

$\hat{Q}\hat{P}$	1	i	j	k	ϵi	ϵj	ϵk	ϵ
1	1	i	j	k	ϵi	ϵj	ϵk	ϵ
i	i	-1	k	$-j$	$-\epsilon$	ϵk	$-\epsilon j$	ϵi
j	j	$-k$	-1	i	$-\epsilon k$	$-\epsilon$	ϵi	ϵj
j	k	j	$-i$	-1	ϵj	$-\epsilon i$	$-\epsilon$	ϵk
ϵi	ϵi	$-\epsilon$	ϵk	ϵj	0	0	0	0
ϵj	ϵj	$-\epsilon k$	$-\epsilon$	ϵi	0	0	0	0
ϵk	ϵk	ϵj	$-\epsilon i$	$-\epsilon$	0	0	0	0
ϵ	ϵ	ϵi	ϵj	ϵk	0	0	0	0

Dual quaternion multiplication has the following properties:

- Associative: $a(bc) = (ab)c = abc$.
- Non-commutative: $ab \neq ba$.
- Not all elements have the multiplicative inverse.
- The identity is the scalar 1.

The subalgebra can be used to represent spatial rigid body dynamics [18]. The real component of the dual quaternion is a quaternion that represents orientation. Similar to what was done with Plücker coordinates, the dual component represents displacement or position. This leads to the following identifications:

- Points can be represented by $\hat{P} = 1 + \epsilon\mathbf{p} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} \right\} + \epsilon \left\{ \begin{array}{c} p_x \\ p_y \\ p_z \\ 0 \end{array} \right\}$.
- Lines can be represented by $\hat{L} = \mathbf{s} + \epsilon\mathbf{s}^0 = \left\{ \begin{array}{c} s_x \\ s_y \\ s_z \\ 0 \end{array} \right\} + \epsilon \left\{ \begin{array}{c} s_x^0 \\ s_y^0 \\ s_z^0 \\ 0 \end{array} \right\}, \|\mathbf{s}\| = 1, \mathbf{s} \cdot \mathbf{s}^0 = 0.$



- Rotations can be represented by $\hat{R} = \hat{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(\mathbf{s} + \epsilon\mathbf{s}^0)$
$$= \begin{Bmatrix} \sin\frac{\theta}{2}s_x \\ \sin\frac{\theta}{2}s_y \\ \sin\frac{\theta}{2}s_z \\ \cos\frac{\theta}{2} \end{Bmatrix} + \epsilon \begin{Bmatrix} \sin\frac{\theta}{2}s_x^0 \\ \sin\frac{\theta}{2}s_y^0 \\ \sin\frac{\theta}{2}s_z^0 \\ 0 \end{Bmatrix}, \hat{q}\hat{q}^* = 1.$$

- Translations can be represented by $\hat{T} = 1 + \frac{1}{2}\epsilon\mathbf{t} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix} + \epsilon \begin{Bmatrix} \frac{1}{2}t_x \\ \frac{1}{2}t_y \\ \frac{1}{2}t_z \\ 0 \end{Bmatrix}$

Transformations can be composed by multiplication so that, for example, a translation followed by a rotation becomes $\hat{Q} = \hat{R}\hat{T}$. The first transformation applied is the one furthest on the right.

The manipulation of points or lines must be done by the conjugation action ABA^{-1}. However, there are four conjugations defined for the Clifford algebra, although only two are of practical use for this project. Two matching dual quaternion conjugations can be defined,

$$\hat{Q}^* = q_0 - \mathbf{q} + \epsilon(q_7 - \mathbf{q}^0) = \hat{q}^* + \epsilon\hat{q}^{0*} \tag{4.11}$$

$$\hat{Q}^\dagger = q_0 - \mathbf{q} + \epsilon(\mathbf{q}^0 - q_7) \tag{4.12}$$

To transform lines the action known as the Clifford conjugation f_{2G} is used,

$$\begin{aligned} f_{2G} : Cl(p,q,r) &\longrightarrow Cl(p,q,r) \\ A : B &\longmapsto ABA^* \end{aligned} \tag{4.13}$$

For points the action f_{4G} is used,

$$\begin{aligned} f_{4G} : C(p,q,r) &\longrightarrow Cl(p,q,r) \\ A : B &\longmapsto ABA^\dagger \end{aligned} \tag{4.14}$$



The conjugations also explain why the unit dual quaternion representing a displacement uses half of the rotation or translation. The conjugation action multiplies the transformation quaternion twice. This can be shown with the simple translation of a point,

$$\hat{T}\hat{P}\hat{T}^{\dagger} = (1 + \frac{1}{2}\epsilon\mathbf{t})(1 + \epsilon\mathbf{p})(1 + \frac{1}{2}\epsilon\mathbf{t}) = (1 + \epsilon(\mathbf{p} + \frac{1}{2}\mathbf{t}))(1 + \frac{1}{2}\epsilon\mathbf{t}) = 1 + \epsilon(\mathbf{p} + \mathbf{t}) \tag{4.15}$$

It is also to note that the dual quaternions double cover the group of proper rigid body transformations $SE(3)$, which is to say both $\hat{Q}$ and $-\hat{Q}$ represent the same displacement. This is important to keep in mind when comparing transformations.

4.9 Exponential Map

The Lie algebra of a group can be thought of as the tangent space at the identity element. The Lie algebra elements are a left-invariant vector field on the group. If $[X]$ is a matrix representing a tangent vector at the identity, then the tangent vector at the point $\mathbf{g}$ of the group will be $\mathbf{g}[X]$. Curves that are tangent to the field at each point can be written as the differential equation,

$$\frac{d\mathbf{g}}{dt} = \mathbf{g}[X] \tag{4.16}$$

The analytic solution would be,

$$\mathbf{g}(t) = e^{t[X]} \tag{4.17}$$

The exponential of a matrix can be defined by the series,

$$e^{[X]} = [I] + [X] + \frac{1}{2!}[X]^2 + \frac{1}{3!}[X]^3 + \cdots \tag{4.18}$$

This series can be proven to converge. We can find the Lie algebra of a finite screw motion from Eqn.4.2 by taking the derivative at $\theta = 0$ [74],

$$S = \begin{bmatrix} [\Omega] & \frac{\omega p}{2\pi}\mathbf{v} - [\Omega]\mathbf{u} \\ 0 & 0 \end{bmatrix} \tag{4.19}$$



$[\Omega]$ is the 3x3 anti-symmetric matrix that corresponds to the 3D vector $\boldsymbol{\omega}$,

$$[\Omega] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{4.20}$$

If $[X]$ is a matrix representing an element of a Lie algebra, then $e^{[X]}$ is an element of the corresponding Lie group. The exponential map sends a Lie algebra element to the Lie group. This works with any 1-parameter subgroup, which translate into any 1 degree of freedom joint. The mapping is also independent of the representation. This allows a joint to be written as the 1-parameter screw,

$$S(\hat{\theta}) = e^{\hat{\theta}J} \tag{4.21}$$

where J is the screw and $\hat{\theta} = \theta + \epsilon d$ is the displacement of the screw. The displacement must contain only one parameter meaning that the joint must either be revolute ($d = 0$) or prismatic ($\theta = 0$). Prismatic and revolute joints can be used to form more complicated joints. Using unit dual quaternions they can be written as [70],

$$\hat{S}(\hat{\theta}) = e^{\frac{\hat{\theta}}{2}\mathsf{S}} = \cos\frac{\hat{\theta}}{2} + \sin\frac{\hat{\theta}}{2}\mathsf{S} \tag{4.22}$$

It is important to note that a screw motion of $\hat{\theta} = \theta + \epsilon d$ becomes the dual quaternion motion $\frac{\hat{\theta}}{2} = \frac{\theta}{2} + \epsilon\frac{d}{2}$. This is because of the conjugation action ABA^{-1} used in Clifford algebra to transform coordinates as explained in Sec.4.8.

The exponential map is widely used in kinematics for its versatility and compactness. Although it can use different representations, this project will use the dual quaternion representation shown in Eqn.(4.22). Using the exponential map, the forward kinematics and design equations of the kinematic can be written. For more information on Lie algebra and its application in kinematics refer to [74].



4.10 Forward Kinematics

Given a kinematic serial chain with n joints and the initial transformation $\hat{G}$ from the global reference to the first joint axis, the kinematics equation can be written using the product of exponentials of the screws corresponding to the joint axes [57],

$$\begin{aligned}\hat{T}(\boldsymbol{\theta}) &= \hat{S}_1(\hat{\theta}_1)\hat{S}_2(\hat{\theta}_2)\cdots\hat{S}_n(\hat{\theta}_n)\hat{G} = e^{\frac{\hat{\theta}_1}{2}\mathsf{S}_1}e^{\frac{\hat{\theta}_2}{2}\mathsf{S}_2}\cdots e^{\frac{\hat{\theta}_n}{2}\mathsf{S}_n}\hat{G} \\ &= (\cos\frac{\hat{\theta}_1}{2} + \sin\frac{\hat{\theta}_1}{2}\mathsf{S}_1)\cdots(\cos\frac{\hat{\theta}_n}{2} + \sin\frac{\hat{\theta}_n}{2}\mathsf{S}_n)\hat{G} \end{aligned} \tag{4.23}$$

where $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + \epsilon\mathbf{d}$ is the displacement of the joints. Given an arbitrary reference configuration the forward kinematics can be calculated using the relative displacement from the reference configuration [66],

$$\begin{aligned}\hat{Q}(\boldsymbol{\Delta\hat{\theta}}) &= \hat{S}_1(\Delta\hat{\theta}_1)\hat{S}_2(\Delta\hat{\theta}_2)\cdots\hat{S}_n(\Delta\hat{\theta}_n) = e^{\frac{\Delta\hat{\theta}_1}{2}\mathsf{S}_1}e^{\frac{\Delta\hat{\theta}_2}{2}\mathsf{S}_2}\cdots e^{\frac{\Delta\hat{\theta}_n}{2}\mathsf{S}_n} \\ &= (\cos\frac{\Delta\hat{\theta}_1}{2} + \sin\frac{\Delta\hat{\theta}_1}{2}\mathsf{S}_1)\cdots(\cos\frac{\Delta\hat{\theta}_n}{2} + \sin\frac{\Delta\hat{\theta}_n}{2}\mathsf{S}_n) \end{aligned} \tag{4.24}$$

where $\boldsymbol{\Delta\hat{\theta}} = \boldsymbol{\Delta\theta} + \epsilon\boldsymbol{\Delta d}$ is the relative displacement of the joints from the reference configuration. It is important to note that when doing forward kinematics as relative displacements from a reference configuration we no longer need the initial transformation $\hat{G}$.

4.11 Design Equations

Given $m-1$ relative transformations $\hat{P}_{1j} = \hat{T}_j\hat{T}_1^{-1}, j = 2,\ldots,m$ defining the task, the forward kinematics for each finger can be calculated [19],

$$\begin{aligned}&\hat{S}_{1j}(\Delta\hat{\theta}_{1j})\hat{S}_{2j}(\Delta\hat{\theta}_{2j})\hat{S}_{3j}(\Delta\hat{\theta}_{3j})\hat{S}^k_{4j}(\Delta\hat{\theta}^k_{4j})\cdots\hat{S}^k_{7j}(\Delta\hat{\theta}^k_{7j}) - \hat{P}^k_{1j} = 0, && k \in 1,2, \\ &\hat{S}_{1j}(\Delta\hat{\theta}_{1j})\hat{S}_{2j}(\Delta\hat{\theta}_{2j})\hat{S}_{3j}(\Delta\hat{\theta}_{3j})\hat{S}^k_{4j}(\Delta\hat{\theta}^k_{4j})\cdots\hat{S}^k_{8j}(\Delta\hat{\theta}^k_{8j}) - \hat{P}^k_{1j} = 0, && k \in 3,4,5, \\ &&& j = 2,...,m \end{aligned} \tag{4.25}$$

The index and middle finger when $k = 1$ and $k = 2$ respectively. They have one less joint than the other 3 fingers as explained in Sec.2.2.





5. Non-linear Solver

Once the design equations are defined they must be solved in order to fit the theoretical hand model to the real hand data. This must be done by a non-linear numerical global optimizer also known as a non-linear solver. The amount of equations and the non-linearity of them makes this problem a difficult one to solve. Different approaches will be dealt with in this chapter.

5.1 State of the Art

Non-linear optimization is a field that has grown immensely with the advent of modern computers. This has led to an important growth of algorithms and publications on the topic in the last couple of decades, especially in the field of meta-heuristics. As meta-heuristics are not a precise science, many improvements are still being done on older algorithms like the genetic algorithm [26].

Swarm theory based global optimizers have also been a growing subject of study like the Particle Swarm Optimizer [20] or the Bee Colony Optimizer [82]. This has led to new variants that try to keep diversity high [60] or that use hierarchical structures [50]. Work has also been done on merging different algorithms to create hybrid algorithms like Ant Colony Optimization and Genetic Algorithms [89] or Particle Swarm and Genetic Algorithms [89]. Serial configurations of different solvers also has been experimented with [2].

5.2 System Dimension

The first important step to solving the kinematic model is by proving there is at least a single solution. This can be accomplished by proving that the number of independent unknowns and equations are at least equal. Since the system is non-linear, a single solution is guaranteed but



it may not be unique. The system must be studied to determine the number and distribution of the solutions.

Table 5.1: Independent unknowns of the algebraic sets used by the solver.

Symbol	Set	Components	Independent	Notes
θ	Angle	1	1	Periodic with period 2π
$\mathbf{s}+\epsilon\mathbf{s}^0$	Plücker coordinates	6	4	$\mathbf{s}\cdot\mathbf{s}^0=0$, $\|\mathbf{s}\|=1$
$\widehat{q}+\epsilon\widehat{q}^0$	Dual quaternion	8	6	$\widehat{q}\cdot\widehat{q}^0=0$, $\widehat{q}\widehat{q}^*=1$

The kinematic equations used are built around 3 algebraic sets. It is important to know the properties of these sets as they will determine the final behaviour of the solver. A brief overview of different properties of the sets is seen in Tab.5.1. It is important to note the number independent unknowns when seeing if an equation system is solvable.

The number of independent unknowns will be denoted by n_x^0 and the number of independent equations by n_f^0 and can be calculated by,

$$n_x^0 = r(\underbrace{4}_{Structural} + \underbrace{(m-1)}_{Joint})$$

$$n_x = r(6+(m-1)) \tag{5.1}$$

$$n_f^0 = b6(m-1)$$

$$n_f = b8(m-1) \tag{5.2}$$

The number of branches is denoted by b and m represents the number of frames. The actual number of variables and equations in the equation system are higher and are represented by n_x and n_f respectively. The degrees of freedom (DoF) of the hand are defined by the kinematic model from Sec.2.3.3. The kinematic chains can be solved for a given number of task positions m that can be calculated by,

$$m = \frac{4r}{6b-r} + 1 > 0 \tag{5.3}$$

$$6b - r > 0 \tag{5.4}$$



5.2.1 System Reduction

Table 5.2: System of equations parameters for different combinations of fingers considered.

b	r	$n_x^0 = n_f^0$	n_x	n_f	m	D	Notes
5	26	780	832	1092	27	10^{1058}	Full hand model
4	22	1056	1100	1452	45	10^{1442}	(5R, 5R, 5R, 4R) fingers
4	21	672	714	938	29	10^{908}	(5R, 5R, 4R, 4R) fingers
3	17	1224	1258	1666	69	10^{1666}	(5R, 5R, 4R) fingers
3	16	576	608	800	33	10^{773}	(5R, 4R, 4R) fingers
2	11	528	550	726	45	10^{698}	(4R, 4R) fingers
1	5	120	130	170	21	10^{325}	5R finger, common solved
1	4	48	56	72	9	10^{119}	4R finger, common solved

The system can be solved with fewer kinematic chains at a time if they comply with Eqn.(5.4). Not all combinations are possible; the possible combinations are shown in Tab.5.2. The minimum amount of frames required for the entire system is 27 if solve the 5 kinematic chains are solved at once. However the fastest system to solve is through solving both fingers with 4 DoF (index and middle) and then proceeding to solve the remaining kinematic chains individually.

The complexity of the system is greatly increased by the number of equations. This makes minimizing the maximum dimension of a system of equations to solve an important goal in the design of the solver. The results of the common joints can then be used to solve the remaining serial chains individually.

5.2.2 Solution Bound

The number of solutions is extremely large and finding an exact upper bound on the number of real solutions is very complicated. Approaches have been done using polynomial homotopy continuation [76] with tools like PHCpack [86]. When applied to dimensional kinematic synthesis the complexity increases greatly. Other approaches like Gröebner bases and elimination theory have been tried. The application of most of these algorithms is not obvious and has not been successfully applied to more complicated dimensional synthesis problems including the one dealt within this project. An overview of these approaches can be found in [59].

For an approximate idea of the dimension, the chains from Eqn.(4.25) can be expanded to



dual quaternions [67]. The components can be expanded into a polynomial to analyze the total degree of the equation system.

Bézout's theorem states that the upper bound of solutions for any general polynomial system of equation can be expressed as the product of the degrees d_i of the polynomials $D = d_1 d_2 \cdots d_n$. D is the total degree of the system which is the upper bound of the number of solutions, complex and real, that the system may have. This provides a very rough approximation of the upper bound of solutions due to high internal structure of the kinematic chains which is overestimates by many orders of magnitude [53].

The degree of a serial chain composed by r revolute joints can be approximated with $(3r)^6$. The total degree of Eqn.(4.25) with $m = 27$ positions can be calculated by,

$$\begin{aligned} d_i &= 21^{6(m-1)} && , i \in 1, 2, \\ d_i &= 24^{6(m-1)} && , i \in 3, 4, 5, \\ D &= \prod_{i=1}^{5} d_i \approx 10^{1058} \end{aligned} \tag{5.5}$$

The total degrees of different solvable variants of the synthesis problem can be seen in Tab.5.2. However, the number of real solutions however is generally much lower. A sharper upper bound is out of the scope of this project.

5.3 Numerical Solver

One of the main issues in finding a suitable numerical solver approach for the system of equations defined in Eqn.(4.25) is the fact that only the Plücker Coordinates of the axes can be approximated when initializing the solver. The angles which make up most of the variables must be randomly initialized in the feasible movement range defined by the hand model in Chapter 2. This usually leads to a starting position that is very far off from a global minimum and thus a global solver is necessary.

When using synthetic data sets, at least a single solution is guaranteed corresponding to the model used to generate the synthetic data. However, there have been no detailed studies on the number of solutions that may be encountered, although a generous upper bound has been



calculated in Sec.5.2.2. With experimental data, the data set will contain noise and error that will most likely not have the exact solution. The system must be minimized through sum of squares. This also complicates the numerical solving of the system.

5.3.1 Levenberg-Marquadt

The Levenberg-Marquadt method [47] is a local non-linear least squares optimizer based on the Gauss-Newton algorithm. This solver uses the Jacobian matrix of the non-linear equation system to iteratively converge to a local minima. If the Jacobian matrix can not be calculated, it can be approximated using the finite difference approximation. The software package MINPACK [37] provides an implementation of the Levenberg-Marquadt method and finite difference Jacobian approximation.

The algorithm was tested in the near area of the solution with different variations of Guassian noise to test the reliability in solving the dimensional kinematic synthesis problem defined by Eqn.(4.25). The results in Appendix A show that it is not suitable when the starting position is far off from the global solution. This makes it too unreliable and unsuitable for the problem at hand to use it by itself for this project. The speed of the algorithm depends heavily on the proximity of a solution and is generally impossible to predict.

5.3.2 Genetic Algorithm

Genetic algorithms [25] are meta-heuristic algorithms that allow solving a non-linear system just by being able to evaluate it at a given point. This means there is no need for expensive finite difference Jacobian approximations. The speed allows the algorithm to explore more of the search space and makes the algorithm behave like a global optimizer given a certain set of parameters that must be experimentally tested [55]. A detailed analysis of equation system at hand is needed to be able to use the algorithm optimally.

The inspiration for genetic algorithms comes from Darwin's theory of evolution. The algorithm generates random individuals to form a population. These individuals are reproduced among each other and are mutated. This reproduction is called crossover. The crossover and mutation, if adjusted properly, cause the algorithm to converge on a system minima.

The choice of selection function, to choose what individuals to crossover; fitness function, to evaluate each individual in the population; crossover function, to determine how individuals



create offspring and mutation function, to determine how individuals mutate are fundamental to the behaviour of the algorithm. They must be chosen carefully: there is no analytical method to determine them. Like most of the configuration of meta-heuristic algorithms, these functions are generally chosen experimentally.

Genetic algorithms have three basic parameters: population, crossover rate and mutation rate. Through modification of these parameters, the speed of convergence can be modified. Due to the large dimension of the system of equations, a slow convergence was needed to explore most of the search space and not converge on local minima. After many tests, the result was that a pure genetic algorithm was neither reliable nor suitable for the problem at hand. The parameters and functions used by the genetic algorithm implementation can be seen in Appendix B.

5.3.3 Hybrid Solver

The solution proposed to the convergence problem was combining both, the genetic algorithm and the Levenberg-Marquedt local solver, to form a more reliable global solver. The local solver would be used to reduce the search space from the entire space formed by the kinematic synthesis from Sec.4.2 to only the local minima in the same search space. By reducing the search space to local minima, the genetic algorithm is much more efficient at finding solutions and does not generally get stuck at local minima. The drawback is that the computation time gets dramatically increased.

5.4 Implementation

The implementation of the Levenberg-Marquardt was provided by the MINPACK software suite [37] while the genetic algorithm was written specifically for this project in order to have a tight integration with the Levenberg-Marquadt solver and is written in C. The entire dual quaternion implementation is provided by libdq [75] developed specifically for this project. Static analysis tools like Cppcheck [13] and Clang [42] in conjunction with dynamic analysis tools like valgrind [58] were used in verifying the quality of the code.

Genetic algorithms are very easy to parallelize, allowing the solver to run on multiple CPU cores to optimize the runtime. The code was profiled to try to optimize it. However, 90% of



the computing time is spent calculating doing QR decomposition of the system as part of the Levenberg-Marquadt method provided by MINPACK [37] as seen in Appendix B.

It is extremely difficult to extrapolate data due to the fact that not much study has been done in the dimensional kinematics synthesis problem with such a complex model and topology. From experimentation, it has been seen that the results of the solver (both in quality and in computation time) vary greatly with the input data sets used. Generally the more movement and difference between poses in the input data set provides a much better result. However due to the computation time not much has been studied in this regard.

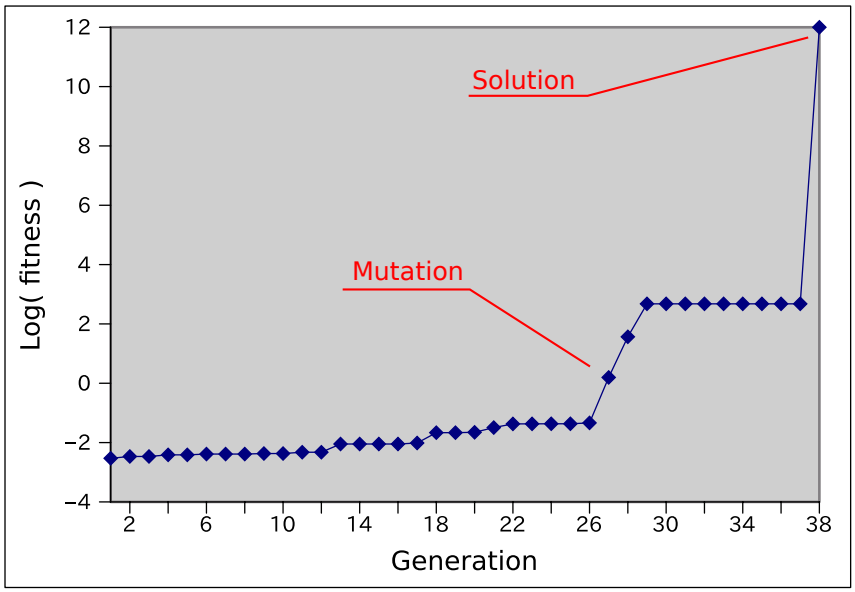


Figure 5.1: Convergence of a hybrid solver run.

The fitness function used is the inverse of the sum of all the error obtained from the design equations. This function is defined positive so that a fitness-biased selection scheme for crossover can be used. A solution is considered to be found when the fitness surpasses 10^{12}. At this point the error can be attributed to floating point imprecision. An example of the convergence can be seen in Fig.5.1.

Average run times can be seen in Tab.5.3. The high standard deviations is indicative of the runtime variability of the algorithm. The full manual for using the solver can be found in Appendix D.

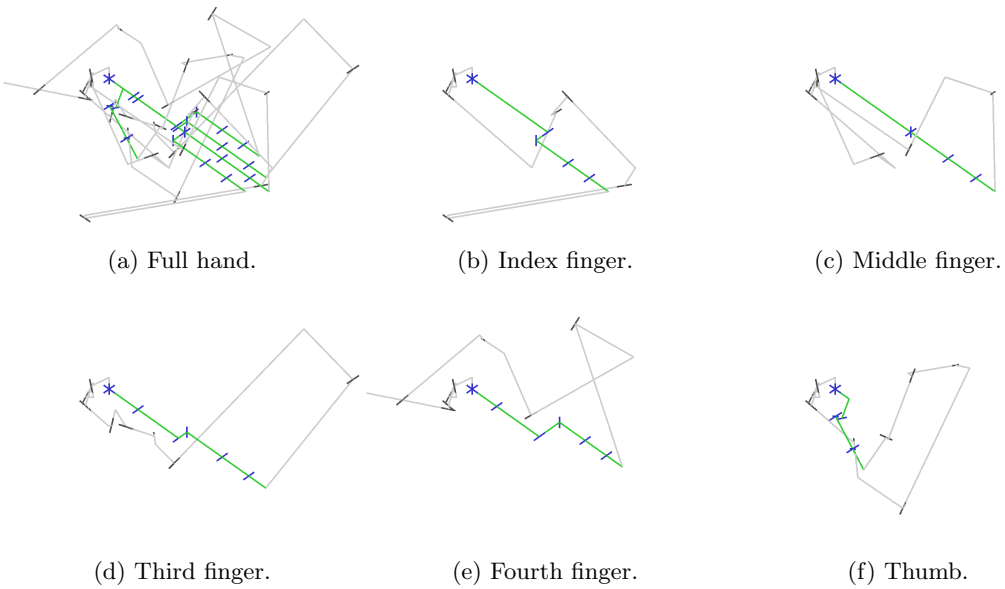


Table 5.3: Run time results for genetic algorithm.

Generation mean	Generation stddev	Run time mean (hours)	Run time stddev (hours)
38.1071	13.4862	59.7065	25.6962

5.5 System Solutions

To study the behaviour of the solver, an application was created to generate synthetic data sets based on the hand model from Sec.2.3.3. The solver was then run against the synthetic data and results were compared. This led to the finding of many alternate solutions to the dimensional kinematic synthesis problem of the hand as seen in Fig.5.2. These solutions move exactly like a hand, but have an extremely non-anthropomorphic form.



(a) Full hand. (b) Index finger. (c) Middle finger.

(d) Third finger. (e) Fourth finger. (f) Thumb.

Figure 5.2: A solution of the general kinematics problem for a hand task.

An attempt to solve this issue was done by adding constraints to all the variables, however this nearly doubled the number of equations in the system. The increased complexity does not make it feasible to attempt to solve the constrained system. The desired solution was never found.



6. Results

This chapter details the results of the different objectives and experiments done throughout the project. The results more or less match the maturity of the research in each of the fields they relate to.

6.1 Overview

During the course of this project, two major problems were defined:

- PnP or pose estimation problem (Chapter 3).
- Motion-to-form or dimensional kinematic synthesis problem (Chapter 4).

The first deals with detecting the fingernail poses of the hand and the latter deals with sizing a kinematic model that is able to move along the given poses. Both problems are very important in their respective fields. However, while the PnP is generally considered a mature problem and is being optimized, the motion-to-form problem is still very undeveloped, with only simple cases algebraically solved to date.

6.2 Hand Detection

The results of tracking the hand fingernail markers are very satisfactory. Within a controlled set up, the markers are detected and tracked very well. Computational time is also low by the usage of the DAISY descriptors. The only problems were with motion blur as seen in Fig.6.1. However, by slowing down the hand movements, they could be avoided. Other proposals to



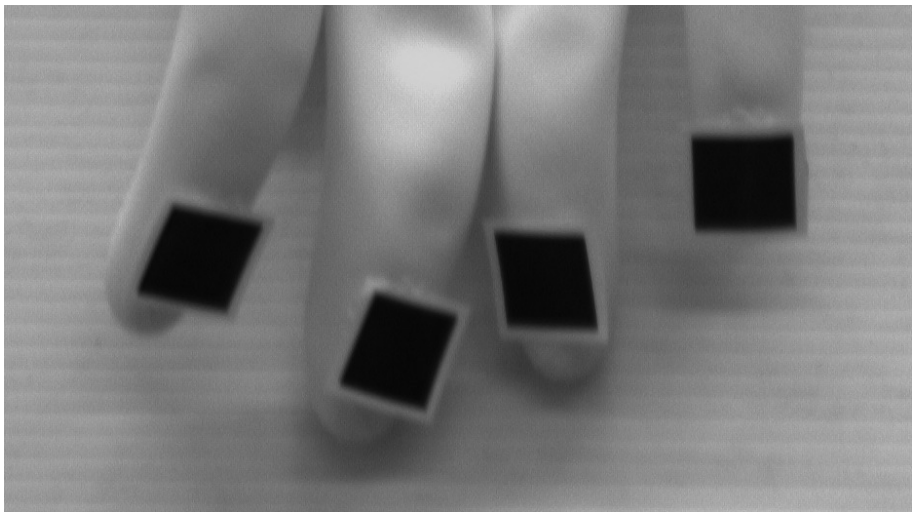


Figure 6.1: Motion blur on a captured image.

solve motion blur are to use a faster camera, use strobe lighting or use Kalman filters to smooth the movement.

The error when solving the PnP or pose estimation problem is also small with few pixels of offset as seen in Chapter 3. The error is non-linear depending primarily on the orientation of the markers and the noise. However, such error can be minimized by manually selecting the frames or using high resolution cameras. Manually selecting frames can also aid the solver in convergence by choosing frames representing very different hand poses.

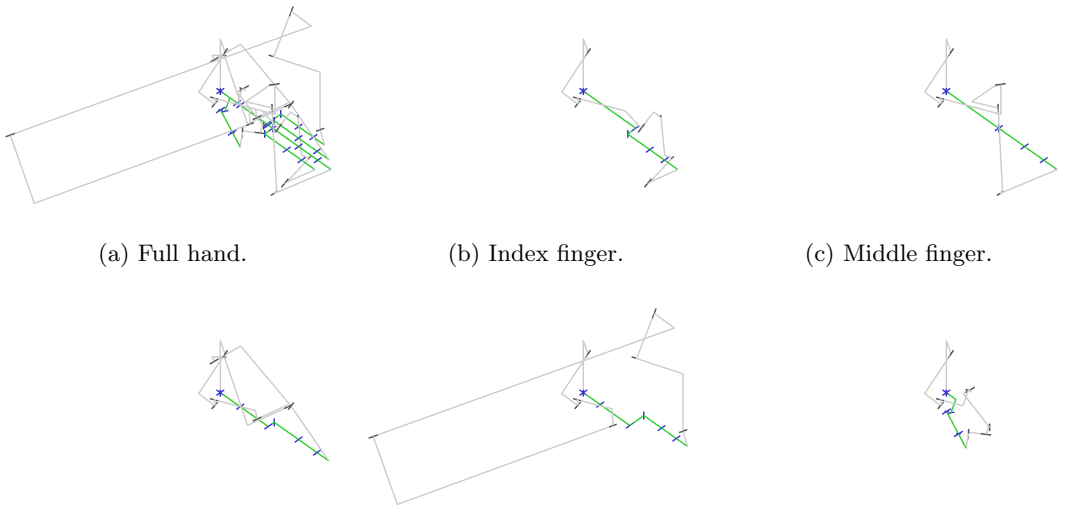
6.3 Kinematic Synthesis

The motion-to-form problem is very complex and currently can be considered an unsolved problem. There are no generic solutions and not much research has been done in complex topologies such as the one described in Chapter 4.

The problem was defined and a solver, using a genetic algorithm and Levenberg-Marquadt optimizer, was written and was able to find solutions. However, none of the solutions matched the synthetic solution used to create the synthetic dataset. This can be explained due to the complexity of the problem studied in the same chapter and found to have a Bézout bound of 10^{1058}.

Without using dual quaternions, the actual number of variables and equations would be much larger than what is discussed in Chapter 5 and the design equations would be unfeasibly large [24]. This shows the importance of modern mathematics in the field of kinematic synthesis.





(a) Full hand. (b) Index finger. (c) Middle finger.

(d) Third finger. (e) Fourth finger. (f) Thumb.

Figure 6.2: Another solution of the general kinematics problem for a hand task.

Various approaches on the solving problems were proposed by splitting the complete hand model into submodels that could be solved. The fastest submodel was using only the index and middle fingers.

Table 6.1: Run time results for genetic algorithm.

Generation mean	Generation stddev	Run time mean (hours)	Run time stddev (hours)
38.1071	13.4862	59.7065	25.6962

Due to the dimension of the problem it takes an average of 59 hours and 38 generations to solve a synthetic dataset as seen in Tab.6.1. This may seem like a long time, however profiling was done and the bottleneck in the code was identified to be the QR decomposition that is part of the MINPACK Levenberg-Marquadt solver with over 90% of the CPU time. The dual quaternion implementation of the design equation is the slowest part excluding MINPACK and represents only 1% of the CPU time.

The solutions found are also an interesting object of study by themselves. They are a set of non-anthropomorphic robots with the same amount of degrees of freedom (DoF) as the human hand model from Chapter 2 that move exactly in the same way as a human hand. An example solution that is compared to the theoretical hand model can be seen in Fig.6.2. These robots can be designed to perform any task, that is a set of poses of fingers that a human hand can do, while having an aspect that greatly differs from the human hand. This can have other applications



like the design of exoskeletons that mount on the hand and can move exactly as such.

6.4 Environmental Impact

There is no real environmental impact associated to this project. By both being of low scale deployment and not needing any hardware fancier than a desktop computer, there is no more impact than the CO_2 emitted while running the desktop computer and the paper and cardboard used for the markers which is less than half an A4 paper. To all practical effects, such impact can be considered negligible.

However successful implantation of the results shown in this project could be used to avoid MRI or x-ray scans on patients by doing joint analysis with computer vision. This could be used for detection of joint problems in patients, analyzing athlete performance and many other cases. X-ray scans produce toxic silver thiosulfate which needs special treatment to avoid pollution [1]. X-rays also pose a health risk due to exposure to radiation. MRI scans use a lot of energy and may require special dyes. Both of these solutions while, providing more anatomical detail, may not be necessary if ailments can be previously detected with computer vision.

6.5 Budget

Table 6.2: Total project budget.

Budget	Percent of Total	Cost (€)
Programming	94.8%	€128,591
Software Licenses	4.3%	€5,834
Equipment	0.9%	€1,238
Total	100%	€135,663

Being almost entirely a software-based research project, most of the cost derives from the development of the applications as seen in Tab.6.2. The focus of the budget is the cost of the development of the project, actual usage costs highly depend on the exact usage given to it. Thus the usage costs will more or less amount to the cost of the technician to handle the software. A full overview of the budget can be seen in Appendix C.



Conclusions

The project has the goal of adjusting a theoretical model to experimental data obtained through computer vision. This can be split into 3 parts: creation of a theoretical model, computer vision and kinematic solver.

A widely-accepted theoretical model was defined in Chapter 2. The computer vision part has met the expectations as it is capable of tracking and following separate finger markers and reconstructing their poses from 2D images. However, the kinematic solver was not as successful in the strict sense. These results are proportional to the research done in each field.

The marker detection was shown to be a variant of the pose estimation problem or PnP problem as exposed in Chapter 3. The PnP problem is a mature and solved problem. This can be seen in the results of the computer detection, which are of accurate marker tracking with reliable pose estimations.

The kinematic solver was solving the motion-to-form or dimensional kinematic synthesis problem. This problem, as exposed in Chapter 4, is an open problem for complicated kinematic models like the one used in this project. Before the start of the project, little was known about the exact behaviour and equation complexity of the kinematics of the human hand. In Chapter 5 the dimension and behaviour was both theoretically approximated and experimentally tested and found to be of enormous complexity. The superior limit of the number of solutions to the generic hand kinematic synthesis was found to be 10^{1058} solutions, which is an extremely large amount. The trouble of numerically solving the design equations could not be entirely predicted.

The solver was able to solve the design equations despite the complexity and enormous bound of solutions. However, none of the solutions found matched the desired solution. These solutions also have a direct application in the design of exoskeletons for the hand. The solutions can represent non-anthropomorphic robotic mechanisms that have the same range of movements of a human hand. Therefore, a robotic exoskeleton could be personalized for a specific hand.



This could augment the capabilities of the individual or help compensate a partial disability.

From a more theoretical point of view, it is interesting to note the fact that using a tree-like topology of kinematic chains allowed performing dimensional kinematic synthesis with many degrees of freedom. Each individual kinematic chain can not be individually solved as they have infinite solutions. However, when adding the tree-topology constraints the entire system has a finite number of solutions. This is pioneer work in the field, as it is the first project to perform dimensional kinematic synthesis to such topology.

This project has defined and solved many of the hurdles for the completion of the original ambitious goal. Future work can focus on improving the solver by using alternative techniques or improving the current solver by adding more realistic constraints to limit the search space to only pure anthropomorphic movements. Another line of work could be to research different set of constraints for the design of personalized exoskeletons.

Overall, the research done shows great promise and as the research in this topic matures, especially in the field of kinematic synthesis, the techniques presented in this project will be able to be refined and adjusted to achieve further results.



Acknowledgements

I would like to thank both of my project directors: Dr. Alba Perez Gracia and Dr. Francesc Moreno Noguer for putting up with my endless questions and doubts during the entire project. I would also like to thank the support of Professor Alberto Sanfeliu as my supervisor.

I would also like to thank my cats: Rasputin and Nuryev, as without them my thesis would have never been completed. My family has also proven invaluable in keeping me alive while working on this project.

In memory of Misi.



Bibliography

[1] Minnesota Pollution Control Agency. "Managing Photographic and X-ray Waste". In: (Oct. 2003).

[2] G. Alizadeh et al. "Serial configuration of genetic algorithm and particle swarm optimization to increase the convergence speed and accuracy". In: *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*. 2010, pp. 272 –277. DOI: 10.1109/ISDA.2010.5687252.

[3] Ethel J. Alpenfels. "The Anthropology and Social Significance of the Human Hand". In: *Artificial Limbs* (May 1955), pp. 4 –22.

[4] A. Aristidou and J. Lasenby. "Motion capture with constrained inverse kinematics for real-time hand tracking". In: *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*. 2010, pp. 1 –5. DOI: 10.1109/ISCCSP.2010.5463419.

[5] M.S.M. Asaari and S.A. Suandi. "Hand gesture tracking system using Adaptive Kalman Filter". In: *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*. 2010, pp. 166 –171. DOI: 10.1109/ISDA.2010.5687273.

[6] M. Bertrand, F. Bouchara, and S. Ramdani. "Estimation of uncertainty for Harris corner detector". In: *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*. 2010, pp. 249 –252. DOI: 10.1109/IPTA.2010.5586794.

[7] L. Birglen and C.M. Gosselin. "Kinetostatic analysis of underactuated fingers". In: *Robotics and Automation, IEEE Transactions on* 20.2 (Apr. 2004), pp. 211 –221. ISSN: 1042-296X. DOI: 10.1109/TRA.2004.824641.

[8] M. Bouzit et al. "The Rutgers Master II-new design force-feedback glove". In: *Mechatronics, IEEE/ASME Transactions on* 7.2 (June 2002), pp. 256 –263. ISSN: 1083-4435. DOI: 10.1109/TMECH.2002.1011262.



[9] A. Caffaz and G. Cannata. "The design and development of the DIST-Hand dextrous gripper". In: *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*. Vol. 3. 1998, 2075 –2080 vol.3. DOI: 10.1109/ROBOT.1998.680623.

[10] M. Chalon et al. "The thumb: guidelines for a robotic design". In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. 2010, pp. 5886 –5893. DOI: 10.1109/IROS.2010.5650454.

[11] L.Y. Chang and Y. Matsuoka. "A kinematic thumb model for the ACT hand". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 2006, pp. 1000 –1005. DOI: 10.1109/ROBOT.2006.1641840.

[12] L.Y. Chang and N.S. Pollard. "Method for Determining Kinematic Parameters of the In Vivo Thumb Carpometacarpal Joint". In: *Biomedical Engineering, IEEE Transactions on* 55.7 (July 2008), pp. 1897 –1906. ISSN: 0018-9294. DOI: 10.1109/TBME.2008.919854.

[13] *Cppcheck*. http://cppcheck.sourceforge.net/.

[14] M.R. Cutkosky. "On grasp choice, grasp models, and the design of hands for manufacturing tasks". In: *Robotics and Automation, IEEE Transactions on* 5.3 (June 1989), pp. 269 –279. ISSN: 1042-296X. DOI: 10.1109/70.34763.

[15] J Denavit and R S Hartenberg. "A kinematic notation for lower-pair mechanisms based on matrices". In: *Journal of Applied Mechanics, (pp. 215–221). APPENDIX D* (1955).

[16] L. Dipietro, A.M. Sabatini, and P. Dario. "A Survey of Glove-Based Systems and Their Applications". In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 38.4 (July 2008), pp. 461 –482. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2008.923862.

[17] C. Greenough D.J. Worth and L.S. Chin. "A Survey of C and C++ Software Tools for Computational Science". In: (Dec. 2009).

[18] J.R. Dooley and J.M. McCarthy. "Spatial rigid body dynamics using dual quaternion components". In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. 1991, 90 –95 vol.1. DOI: 10.1109/ROBOT.1991.131559.

[19] K. Duraisamy et al. "Kinematic Synthesis for Smart Hand Prosthesis". In: *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*. 2006, pp. 1135 –1140. DOI: 10.1109/BIOROB.2006.1639245.

[20] R. Eberhart and J. Kennedy. "A new optimizer using particle swarm theory". In: *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*. 1995, pp. 39 –43. DOI: 10.1109/MHS.1995.494215.



[21] A. Erol et al. “A Review on Vision-Based Full DOF Hand Motion Estimation”. In: *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. 2005, p. 75. DOI: 10.1109/CVPR.2005.395.

[22] Johan M. F. Landsmeer Fadi J. Beijjani. “Basic Biomechanics of the Musculoskeletal System”. In: ed. by M Nordin and VH Frankel. Lea & Febiger, 1989. Chap. Biomechanics of the hand, pp. 275–304.

[23] *Flea® 2*. Point Grey. Richmond, BC, Canada.

[24] J. Funda and R.P. Paul. “A computational analysis of screw transformations in robotics”. In: *Robotics and Automation, IEEE Transactions on* 6.3 (June 1990), pp. 348 –356. ISSN: 1042-296X. DOI: 10.1109/70.56653.

[25] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201157675.

[26] Pengfei Guo, Xuezhi Wang, and Yingshi Han. “The enhanced genetic algorithms for the optimization design”. In: *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*. Vol. 7. 2010, pp. 2990 –2994. DOI: 10.1109/BMEI.2010.5639829.

[27] Zhou Hang, Ruan Qiuqi, and Chen Houjin. “A new approach of hand tracking based on integrated optical flow analyse”. In: *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*. 2010, pp. 1194 –1197. DOI: 10.1109/ICOSP.2010.5656105.

[28] C. Harris and M. Stephens. “A Combined Corner and Edge Detector”. In: *Proceedings of the 4th Alvey Vision Conference*. 1988, pp. 147–151.

[29] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. New York: McGraw-Hill Book co., 1964.

[30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[31] Jose L. Hernandez-Rebollar, Nicholas Kyriakopoulos, and Robert W. Lindeman. “The AcceleGlove: a whole-hand input device for virtual reality”. In: *ACM SIGGRAPH 2002 conference abstracts and applications*. SIGGRAPH ’02. New York, NY, USA: ACM, 2002, pp. 259–259. ISBN: 1-58113-525-4. DOI: http://doi.acm.org/10.1145/1242073.1242272. URL: http://doi.acm.org/10.1145/1242073.1242272.

[32] J. Hong and X. Tan. “Calibrating a VPL DataGlove for teleoperating the Utah/MIT hand”. In: *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. 1989, 1752 –1757 vol.3. DOI: 10.1109/ROBOT.1989.100228.



[33] Hai Huang et al. “The Mechanical Design and Experiments of HIT/DLR Prosthetic Hand”. In: *Robotics and Biomimetics, 2006. ROBIO ’06. IEEE International Conference on*. 2006, pp. 896 –901. DOI: 10.1109/ROBIO.2006.340339.

[34] Han-Pang Huang and Chun-Yen Chen. “Development of a myoelectric discrimination system for a multi-degree prosthetic hand”. In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 3. 1999, 2392 –2397 vol.3. DOI: 10.1109/ROBOT.1999.770463.

[35] Chang-Soon Hwang and K. Sasaki. “Control program of two-fingered dexterous manipulation with primitive motions”. In: *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 3. 2003, 2902 –2907 vol.3. DOI: 10.1109/IROS.2003.1249311.

[36] T. Iberall et al. “On the development of EMG control for a prosthesis using a robotic hand”. In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. 1994, 1753 –1758 vol.2. DOI: 10.1109/ROBOT.1994.351339.

[37] J. J. Moré and B. S. Garbow and K. E. Hillstrom. *User Guide for MINPACK-1, Report ANL-80-74*. See http://www-fp.mcs.anl.gov/otc/Guide/softwareGuide/Blurbs/minpack.html. Argonne, Illinois, USA 1980.

[38] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[39] H. Kawasaki, T. Komatsu, and K. Uchiyama. “Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II”. In: *Mechatronics, IEEE/ASME Transactions on* 7.3 (Sept. 2002), pp. 296 –303. ISSN: 1083-4435. DOI: 10.1109/TMECH.2002.802720.

[40] A. Kiso and H. Seki. “Optimal mapping of torus self-organizing map for forearm motion discrimination based on EMG”. In: *SICE Annual Conference 2010, Proceedings of*. 2010, pp. 80 –83.

[41] Thierry Laliberté and Clément M. Gosselin. “Simulation and design of underactuated mechanical hands”. In: *Mechanism and Machine Theory* 33.1-2 (1998), pp. 39 –57. ISSN: 0094-114X. DOI: DOI:10.1016/S0094-114X(97)00020-7. URL: http://www.sciencedirect.com/science/article/B6V46-3SYPRG8-T/2/bbb8e20738204d6f5aa498311733387a.

[42] Chris Lattner and Vikram Adve. “LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation”. In: *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*. CGO ’04.



Washington, DC, USA: IEEE Computer Society, 2004, pp. 75–. ISBN: 0-7695-2102-9. URL: http://portal.acm.org/citation.cfm?id=977395.977673.

[43] E. Lee, C. Mavroidis, and J. P. Merlet. "Five Precision Point Synthesis of Spatial RRR Manipulators Using Interval Analysis". In: *ASME Journal of Mechanical Design* 126(5) (2004), pp. 842–850.

[44] Sung Uk Lee and I. Cohen. "3D hand reconstruction from a monocular view". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 3. 2004, 310 –313 Vol.3. DOI: 10.1109/ICPR.2004.1334529.

[45] Dawei Leng and Weidong Sun. "Finding all the solutions of PnP problem". In: *Imaging Systems and Techniques, 2009. IST '09. IEEE International Workshop on*. 2009, pp. 348 –352. DOI: 10.1109/IST.2009.5071663.

[46] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "EPnP: An Accurate O(n) Solution to the PnP Problem". In: *Int. J. Comput. Vision* 81 (2 2009), pp. 155–166. ISSN: 0920-5691. DOI: 10.1007/s11263-008-0152-6. URL: http://portal.acm.org/citation.cfm?id=1487388.1487412.

[47] K. Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly Journal of Applied Mathmatics* II.2 (1944), pp. 164–168.

[48] D.G. Lowe. "Object recognition from local scale-invariant features". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. 1999, 1150 –1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[49] Chien-Ping Lu, Gregory D. Hager, and Eric Mjolsness. "Fast and Globally Convergent Pose Estimation from Video Images". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.6 (2000), pp. 610–622. ISSN: 0162-8828. DOI: http://dx.doi.org/10.1109/34.862199.

[50] Lin Lu et al. "An improved particle swarm optimization algorithm". In: *Granular Computing, 2008. GrC 2008. IEEE International Conference on*. 2008, pp. 486 –490. DOI: 10.1109/GRC.2008.4664694.

[51] J. M. McCarthy. *Introduction to theoretical kinematics*. Cambridge, MA, USA: MIT Press, 1990. ISBN: 0-262-13252-4.

[52] J. Michael McCarthy. "21st Century Kinematics - Synthesis, Compliance and Tensegrity". In: (May 2011).

[53] J. Michael McCarthy and Gim Song Soh. "Geometric Design of Linkages". In: New York: Springer, 2010. Chap. 14. ISBN: 078-1-4419-7892-9.



[54] Krystian Mikolajczyk and Cordelia Schmid. "An affine invariant interest point detector". In: *In Proceedings of the 7th European Conference on Computer Vision*. 2002, pp. 0–7.

[55] Liang Ming, Yuping Wang, and Yiu-Ming Cheung. "On Convergence Rate of a Class of Genetic Algorithms". In: *Automation Congress, 2006. WAC '06. World*. 2006, pp. 1 –6. DOI: 10.1109/WAC.2006.376051.

[56] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. "A survey of advances in vision-based human motion capture and analysis". In: *Comput. Vis. Image Underst.* 104 (2 2006), pp. 90–126. ISSN: 1077-3142. DOI: http://dx.doi.org/10.1016/j.cviu.2006.08.002. URL: http://dx.doi.org/10.1016/j.cviu.2006.08.002.

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

[58] Nicholas Nethercote and Julian Seward. "Valgrind: a framework for heavyweight dynamic binary instrumentation". In: *SIGPLAN Not.* 42 (6 2007), pp. 89–100. ISSN: 0362-1340. DOI: http://doi.acm.org/10.1145/1273442.1250746. URL: http://doi.acm.org/10.1145/1273442.1250746.

[59] James Nielsen and Bernard Roth. "On the Kinematic Analysis of Robotic Mechanisms". In: *The International Journal of Robotics Research* 18.12 (1999), pp. 1147–1160. DOI: 10.1177/02783649922067771. eprint: http://ijr.sagepub.com/content/18/12/1147.full.pdf+html. URL: http://ijr.sagepub.com/content/18/12/1147.abstract.

[60] Guochao Niu, Baodi Chen, and Jianchao Zeng. "Repulsive Particle Swarm Optimization based on new diversity". In: *Control and Decision Conference (CCDC), 2010 Chinese*. 2010, pp. 815 –819. DOI: 10.1109/CCDC.2010.5498113.

[61] C.B. Owen, Fan Xiao, and P. Middlin. "What is the best fiducial?" In: *Augmented Reality Toolkit, The First IEEE International Workshop*. 2002, 8 pp. DOI: 10.1109/ART.2002.1107021.

[62] S. Parasuraman and Kee Chew Yee. "Bio-Mechanical Analysis of Human Hand". In: *Computer and Automation Engineering, 2009. ICCAE '09. International Conference on*. 2009, pp. 93 –97. DOI: 10.1109/ICCAE.2009.68.

[63] *PayScale*. http://www.payscale.com/. 2011.

[64] A. Perez. "Kinematics of Robots". Class Notes. 2011.

[65] A. Perez. "Literature Review and Proposed Research". Internal Review. 2006.



[66] A. Perez and J.M. McCarthy. "Sizing a Serial Chain to Fit a Task Trajectory Using Clifford Algebra Exponentials". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005, pp. 4709 –4715. DOI: 10.1109/ROBOT.2005.1570847.

[67] Alba Perez. "Use of dual quaternions for the dimensional synthesis of serial chains with less than 6 degrees of freedom". PhD thesis. University of California - Irvine, 2003.

[68] Alba Perez and J. M. McCarthy. "Dimensional Synthesis of Bennett Linkages". In: *Journal of Mechanical Design* 125.1 (2003), pp. 98–104. DOI: 10.1115/1.1539507. URL: http://link.aip.org/link/?JMD/125/98/1.

[69] Alba Perez and J. M. McCarthy. "Dual Quaternion Synthesis of Constrained Robotic Systems". In: *Journal of Mechanical Design* 126.3 (2004), pp. 425–435. DOI: 10.1115/1.1737378. URL: http://link.aip.org/link/?JMD/126/425/1.

[70] Alba Perez-Gracia and J. M. McCarthy. "Kinematic Synthesis of Spatial Serial Chains Using Clifford Algebra Exponentials". In: *Proc. of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science* 220.7 (2006), pp. 953–968.

[71] J. Sánnchez-Riera et al. "Simultaneous pose, correspondence and non-rigid shape". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010, pp. 1189 –1196. DOI: 10.1109/CVPR.2010.5539831.

[72] Gerald Schweighofer and Axel Pinz. "Robust Pose Estimation from a Planar Target". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.12 (2006), pp. 2024–2030. ISSN: 0162-8828. DOI: http://dx.doi.org/10.1109/TPAMI.2006.252.

[73] J. M. Selig. *Geometric Fundamentals of Robotics (Monographs in Computer Science)*. SpringerVerlag, 2004. ISBN: 0387208747.

[74] J. M. Selig. *Lie groups and Lie Algebras in Robotics*.

[75] E. Simo-Serra. *libdq: Dual Quaternion Library*. https://github.com/bobbens/libdq. 2011.

[76] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. "Advances in Polynomial Continuation for Solving Problems in Kinematics". In: *Journal of Mechanical Design* 126.2 (2004), pp. 262–268. DOI: 10.1115/1.1649965. URL: http://link.aip.org/link/?JMD/126/262/1.

[77] Smagt P van der Stillfried G. "Movement model of a human hand based on magnetic resonance imaging (MRI)". In: *International Conference on Applied Bionics and Biomechanics (ICABB)*. 2010.



[78] Hai-Jun Su and J. Michael McCarthy. "Kinematic Synthesis of RPS Serial Chains". In: *ASME Conference Proceedings* 2003.37009 (2003), pp. 1041–1047. DOI: 10.1115/DETC2003/DAC-48813. URL: http://link.aip.org/link/abstract/ASMECP/v2003/i37009/p1041/s1.

[79] Hai-Jun Su, Charles W. Wampler, and J. Michael McCarthy. "Geometric Design of Cylindric PRS Serial Chains". In: *Journal of Mechanical Design* 126.2 (2004), pp. 269–277. DOI: 10.1115/1.1667965. URL: http://link.aip.org/link/?JMD/126/269/1.

[80] K. Takahashi et al. "Remarks on robust extraction of hand's silhouette for 3D hand motion capture system". In: *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*. 2009, pp. 2326 –2331. DOI: 10.1109/IECON.2009.5415283.

[81] H. Takeda et al. "Development of prosthetic arm with pneumatic prosthetic hand and tendon-driven wrist". In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. 2009, pp. 5048 –5051. DOI: 10.1109/IEMBS.2009.5333668.

[82] Dusan Teodorovic et al. "Bee Colony Optimization: Principles and Applications". In: *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*. 2006, pp. 151 –156. DOI: 10.1109/NEUREL.2006.341200.

[83] E. Tola, V. Lepetit, and P. Fua. "Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo". In: vol. 32. 5. 2010, pp. 815–830.

[84] M. Vande Weghe et al. "The ACT Hand: design of the skeletal structure". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 4. 2004, 3375 –3379 Vol.4. DOI: 10.1109/ROBOT.2004.1308775.

[85] E. Varga et al. "Survey and Investigation of Hand Motion Processsing Technologies for Compliance with Shape Conceptualization". In: *ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2004).

[86] Jan Verschelde. "Polynomial homotopy continuation with PHCpack". In: *SIGSAM Bull.* 44 (3/4 2011), pp. 217–220. ISSN: 0163-5824. DOI: http://doi.acm.org/10.1145/1940475.1940524. URL: http://doi.acm.org/10.1145/1940475.1940524.

[87] Xu Wenli and Zhang Lihua. "Pose estimation problem in computer vision". In: *TENCON '93. Proceedings. Computer, Communication, Control and Power Engineering.1993 IEEE Region 10 Conference on*. 0. 1993, 1138 –1141 vol.2. DOI: 10.1109/TENCON.1993.320206.

[88] David Wheeler. *SLOCcount*. 2009. URL: http://www.dwheeler.com/sloccount/.



[89] Jui-Yu Wu. "Real-coded genetic algorithm-based particle swarm optimization method for solving unconstrained optimization problems". In: *Electronics and Information Engineering (ICEIE), 2010 International Conference On*. Vol. 1. 2010, pp. V1–194 –V1–198. DOI: 10.1109/ICEIE.2010.5559892.

[90] I. Yamano and T. Maeno. "Five-fingered Robot Hand using Ultrasonic Motors and Elastic Elements". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005, pp. 2673 –2678. DOI: 10.1109/ROBOT.2005.1570517.

[91] Hanxuan Yang, Zhan Song, and Runen Chen. "An improved unscented particle filter for visual hand tracking". In: *Image and Signal Processing (CISP), 2010 3rd International Congress on*. Vol. 1. 2010, pp. 427 –431. DOI: 10.1109/CISP.2010.5647988.

[92] J.S.-C. Yuan. "A general photogrammetric method for determining object position and orientation". In: *Robotics and Automation, IEEE Transactions on* 5.2 (Apr. 1989), pp. 129 –142. ISSN: 1042-296X. DOI: 10.1109/70.88034.

[93] Xudong Zhang et al. "A Normative Database of Thumb Circumduction In Vivo: Center of Rotation and Range of Motion". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 47.3 (Fall 2005), pp. 550–561. DOI: 10.1518/001872005774860069. eprint: http://hfs.sagepub.com/content/47/3/550.full.pdf+html. URL: http://hfs.sagepub.com/content/47/3/550.abstract.

[94] Hanning Zhou and T.S. Huang. "Tracking articulated hand motion with eigen dynamics analysis". In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. 2003, 1102 –1109 vol.2. DOI: 10.1109/ICCV.2003.1238472.





A. MINPACK Solver

The MINPACK [37] solver implements the Levenberg-Marquadt method [47] for non-linear least squares local optimization. When applied to kinematics problems, there are 3 types of unknowns:

- Axis orientation unknowns that represent the orientation of each joint axis.
- Axis position unknowns that represent location of each joint axis.
- Angle unknowns that represent the angle of each joint for each frame.

These will be studied by using Gaussian noise with varying degrees of standard deviation centered around the solution to determine the behaviour of the solver. The average error of the dual quaternions forms the design equations from Sec.4.11 on both, the real component and the dual component of the dual quaternion representing the accumulative rotation and accumulative translation error respectively.

A.1 Rotation Error

Represents the error created by deviation in the joint axis orientation from the solution. Figure A.1 represents the average rotation error and Fig.A.2 represents the average translation error on both, the real and the dual components of the dual quaternions, when using the MINPACK solver with varying degrees of joint axis orientation Gaussian noise.

A.2 Translation Error

Represents the error created by deviation in the joint axis location from the solution. Figure A.3 represents the average rotation error and Fig.A.4 represents the average translation error



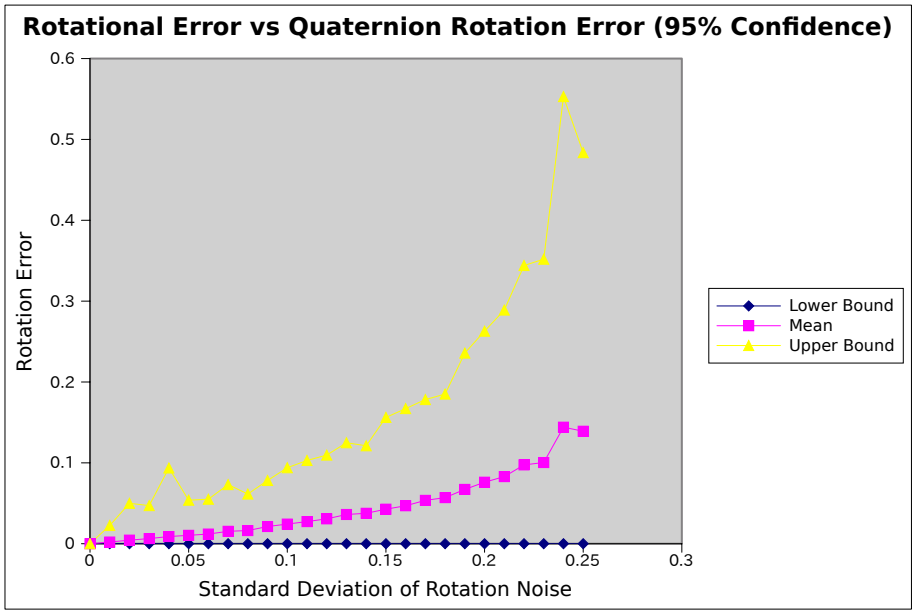


Figure A.1: Average rotation error from rotation noise.

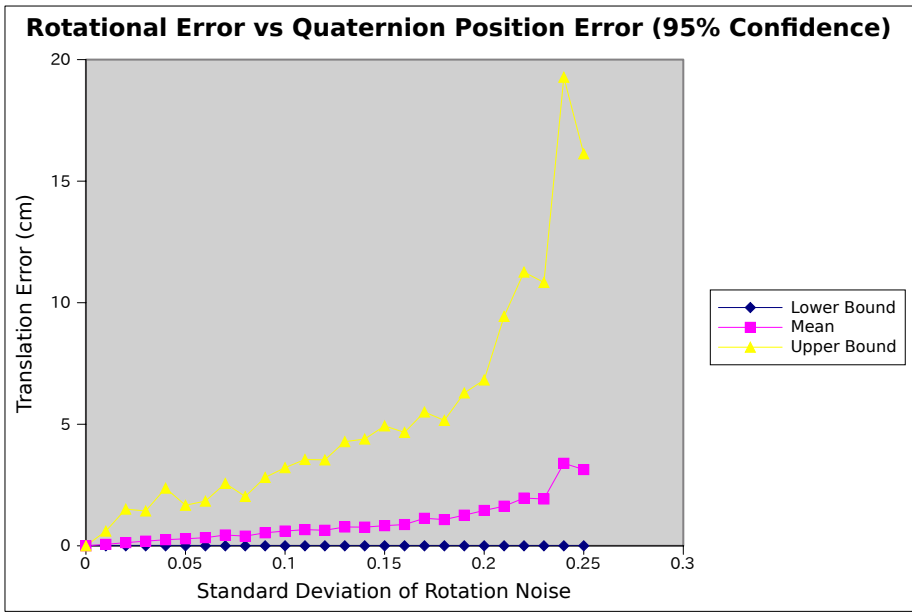


Figure A.2: Average translation error from rotation noise.

on both, the real and the dual components of the dual quaternions, when using the MINPACK solver with varying degrees of joint axis location Gaussian noise.

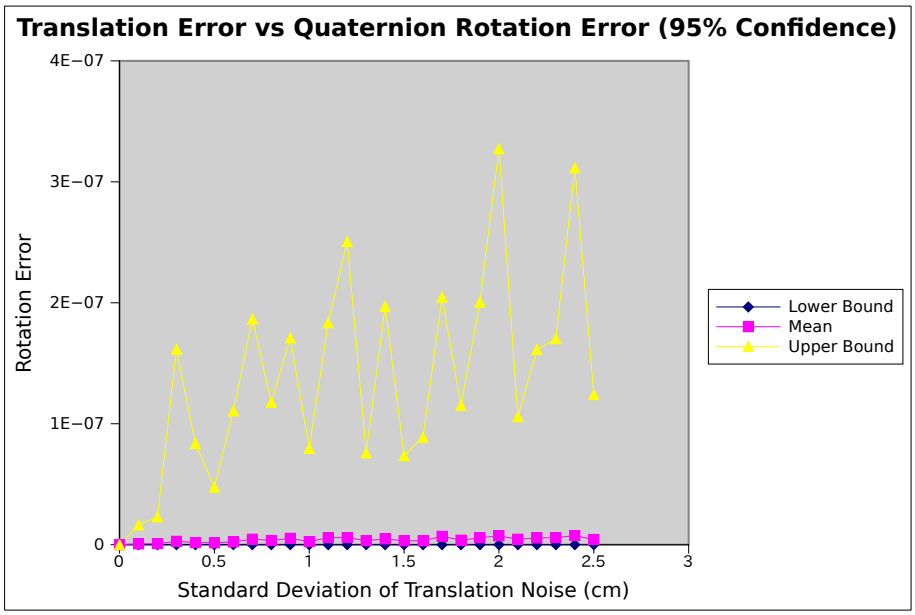


Figure A.3: Average rotation error from translation noise.

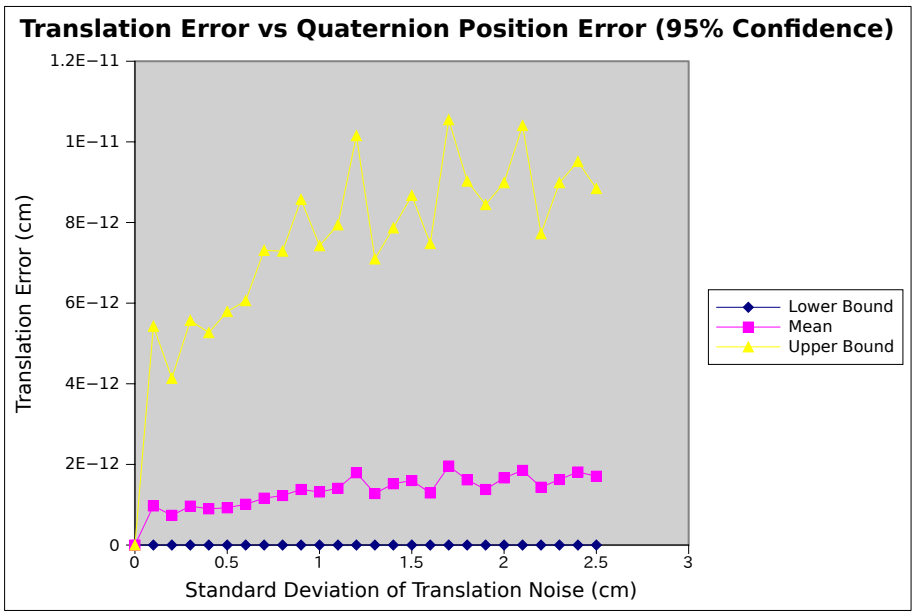


Figure A.4: Average translation error from translation noise.

A.3 Angular Error

Represents the error created by the deviation in the joint angle positions from the solution. Figure A.5 represents the average rotation error and Fig.A.6 represents the average translation error on both, the real and the dual components of the dual quaternions, when using the MINPACK

solver with varying degrees of joint angle Gaussian noise.

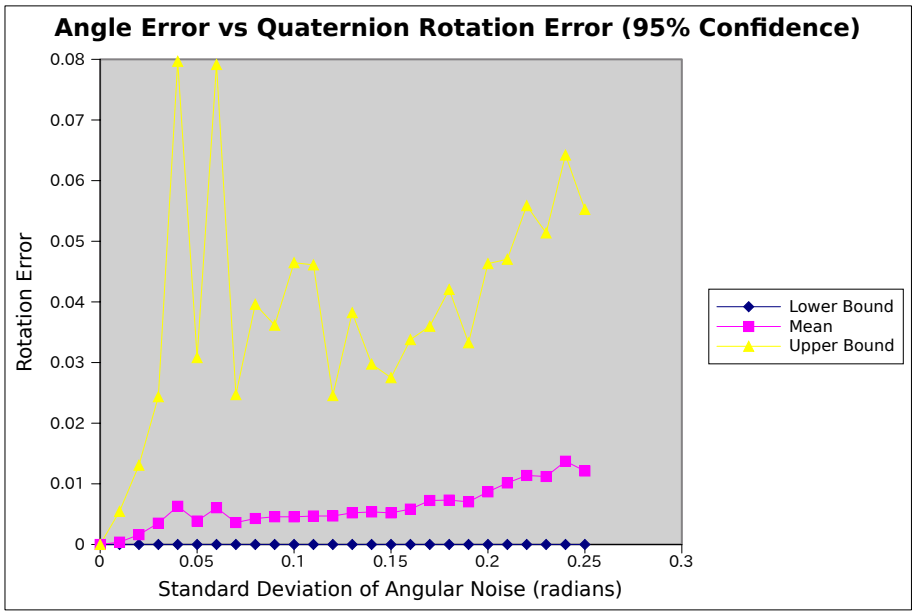


Figure A.5: Average rotation error from angular noise.

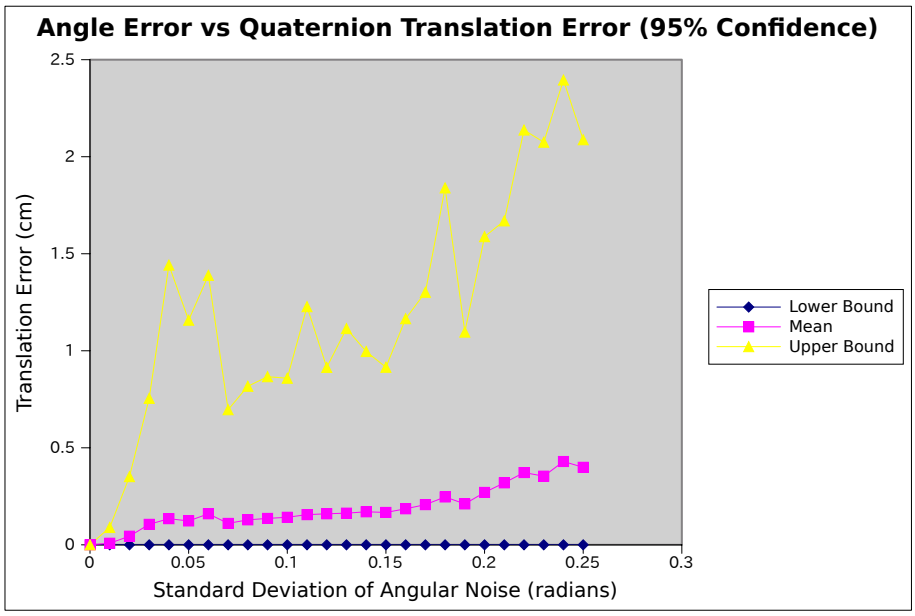


Figure A.6: Average translation error from angular noise.

A.4 Full Error

Full error refers to the composition of rotation, translation and angular error, to give an idea of possible real world behaviour of the solver. A level of standard deviation refers to a 0.1 cm translation standard deviation and a 0.01 rotation standard deviation. Figure A.7 represents the average rotation error and Fig.A.8 represents the average translation error on both the real



and the dual components of the dual quaternions when using the MINPACK solver with varying degrees of global Gaussian noise.

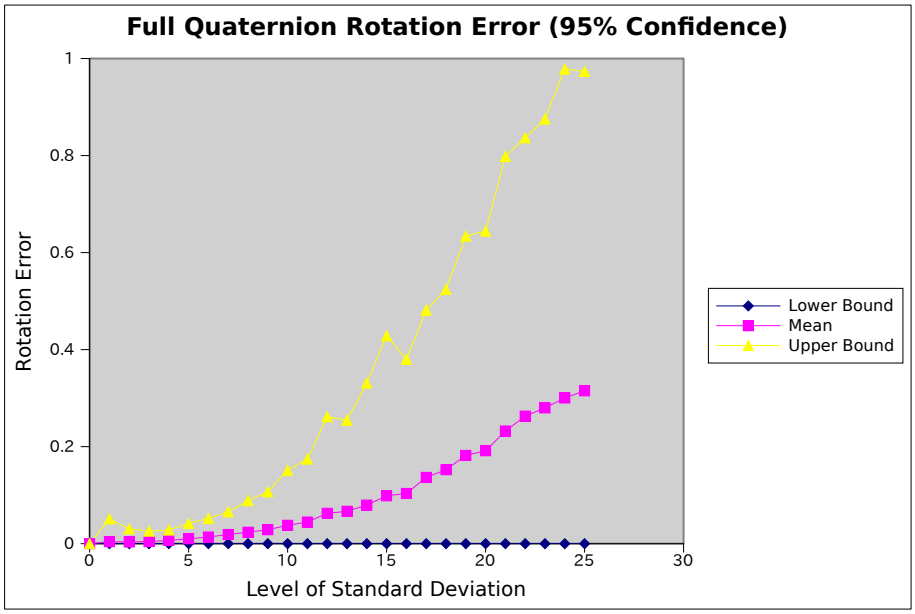


Figure A.7: Average rotation error for full noise.

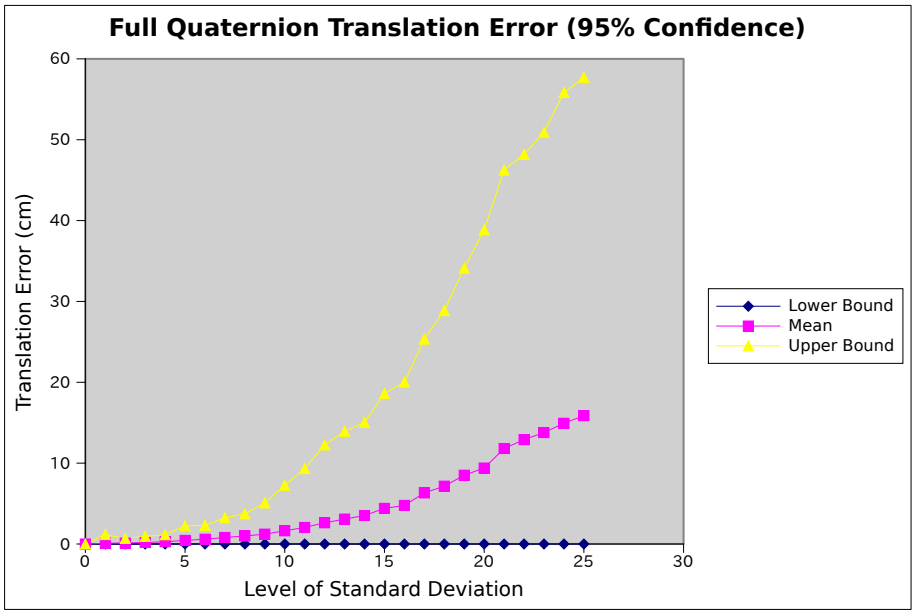


Figure A.8: Average translation error for full noise.



B. Genetic Algorithm Solver

As all meta-heuristic algorithms, a genetic algorithm must be adjusted experimentally. There are generally no analytical methods for setting parameters and the best set up depends entirely on the problem at hand.

B.1 Algorithm Implementation

Genetic algorithms have four important functions that must be designed for the problem:

- Selection function.
- Fitness function.
- Crossover function.
- Mutation function.

These are complemented by parameters that adjust the behaviour.

B.1.1 Selection Function

The selection function, Algorithm 2, determines how to select an individual from the population. It is biased towards higher fitness.



Input: A population of individuals

Output: An individual

$total_fitness := \sum_{individual \in population} individual.fitness$

$number := random_real(0, total_fitness)$

$accum := 0$

for $individual \in population$ **do**
    $fitness := get_fitness(individual)$
    if $number < accum$ **then**
        return $individual$
    end
    $accum := accum + fitness$
end

Algorithm 2: Genetic algorithm selection function.

B.1.2 Fitness Function

The fitness function, Algorithm 3, evaluates a single individual. It is considered to be a solution if the fitness is over 10^{12}.

Input: An individual

Output: Fitness of the individual

$error := evaluate(individual)$ **return** $\frac{1}{\sum_{i=0}^{N} error_i}$

Algorithm 3: Genetic algorithm fitness function.

B.1.3 Crossover Function

The crossover function, Algorithm 4, determines how two individuals produce offspring.

B.1.4 Mutation Function

The mutation function, Algorithm 5, controls how the individual is mutated. This consists of modifying different genes.



Input: A mother and a father.

Output: A daughter and a son.

$daughter := newindividual$

$son := newindividual$

forall $genes \in mother$ **do**

  if $random_boolean()$ **then**

    $a := get_gene(mother, gene)$

    $b := get_gene(father, gene)$

  end

  else

    $a := get_gene(father, gene)$

    $b := get_gene(mother, gene)$

  end

  $add_gene(daugher, a)$

  $add_gene(son, b)$

end

return $daughter, son$

Algorithm 4: Genetic algorithm crossover function.

Input: An individual

Output: A mutated individual

if $random_boolean()$ **then**
  return $mutate_angles(individual)$

end

if $random_boolean()$ **then**

  if $random_boolean()$ **then**

    return $mutate_joint_orientation(individual)$

  end

  else

    return $mutate_joint_position(individual)$

  end

end

return $individual$

Algorithm 5: Genetic algorithm mutation function.



B.1.5 Parameters

The default parameters used by the genetic algorithm can be seen in Tab.B.1. It is important to adjust and tune these parameters everytime the equation system is changed or modified.

Table B.1: Parameters of the genetic algorithm.

Name	Value
Population	2500
Generations	1000
Eliteness	0.05
Crossover	0.10
Mutation	0.50

B.2 Runtime Information

A Intel®Core^{TM}i7 CPU 870 @ 2.93GHz running Ubuntu GNU/Linux 10.04 was used to generate the runtime information in this appendix. The code was compiled with gcc 4.4.3.

The two slowest functions seen in Tab.B.2 are from the MINPACK algorithm. They are used to compute the QR decomposition of the matrix to solve the system and are provided by the CMINPACK library.



Table B.2: Runtime profile of the genetic algorithm.

CPU Usage (%)	Cumulative (s)	Duration (s)	Calls	Speed (ms/call)	Name
46.39	15335.60	15335.60			qrfac
43.74	29794.47	14458.87			qrsolv
2.86	30740.75	946.28			lmdif
2.50	31567.70	826.95			enorm
2.16	32281.91	714.21			lmpar
1.31	32716.52	434.61			fdjac2
0.99	33045.31	328.80	234512297	0.00	syn_calc_branch
0.03	33054.61	9.30	117247734	0.00	syn_map_vec_from_x
0.02	33061.02	6.41	117313781	0.00	syn_map_claim_to_vec
0.00	33062.41	1.39	117238497	0.00	minpack_eqns
0.00	33062.67	0.26	2	130.03	kin_obj_tcp_save
0.00	33062.81	0.14	117276759	0.00	syn_map_vec_to_fvec
0.00	33062.93	0.12	10279	0.01	syn_solve_minpack
0.00	33062.99	0.06	3	20.00	kin_obj_chain_save
0.00	33063.05	0.06			dpmpar
0.00	33063.07	0.02	9859	0.00	ga_ent_evaluate_pthread
0.00	33063.08	0.02	359704	0.00	syn_claim_add
0.00	33063.09	0.01	430003	0.00	rand_double
0.00	33063.10	0.01	113129	0.00	kin_joint_dupInit
0.00	33063.11	0.01	30906	0.00	kin_obj_destroy
0.00	33063.12	0.01	30847	0.00	kin_obj_chain_dup
0.00	33063.13	0.01	20562	0.00	syn_branch_iter_walk
0.00	33063.14	0.01	10302	0.00	syn_free
0.00	33063.15	0.01	295	0.03	ga_ent_seed_pthread
0.00	33063.16	0.01	1	10.00	syn_solve_ga
0.00	33063.17	0.01			ga_ent_pthread
0.00	33063.18	0.01	113073	0.00	kin_joint_claim



Table B.3: Runs of the genetic algorithm.

Generations	Run times (hours)
32	40.0500
24	96.6667
33	33.3667
57	88.8500
73	58.7333
24	35.1833
31	40.6167
20	32.6500
67	65.4167
39	42.2000
52	48.0167
47	45.1167
46	62.0167
34	46.2833
25	25.0000
38	53.9167
22	21.6500
33	45.2833
41	40.0333
45	37.0000
31	81.8333
29	79.0667
30	81.5167
59	119.9667
31	84.4167
36	86.3833
40	90.2000
28	90.3500



C. Budget

Being an engineering project, a budget can be attributed to the project. The budget will focus on the development of the tools developed in this project to detect the hand characteristics and size the kinematic hand model. The budget can be split into three parts: programming, software licensing and equipment.

C.1 Programming

Table C.1: Programming budget.

Application Name	Physical Source Lines of Code (SLOC)	Cost (€)
Solver	7,112	€119,126
Data Generator	577	€8,384
Support Utilities	82	€1,081
Total	7,771	€128,591

The complexity of the code and thus the development value can be calculated with SLOCcount [88]. The average salary of a computer engineer of €31,115 [63] in Spain is used to generate the approximation. The global output is shown below:

```
Total Physical Source Lines of Code (SLOC)                = 7,771
Development Effort Estimate, Person-Years (Person-Months) = 1.72 (20.66)
   (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.66 (7.90)
   (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 2.62
Total Estimated Cost to Develop                           = €128,591
```



(average salary = €31,115/year, overhead = 2.40).

The values give a good estimation at the amount of people and cost to develop the code used and developed for the project. The development cost related to the actual software created can be seen in Tab.C.1.

C.2 Software Licensing

Table C.2: Software licensing budget.

Application Name	Yearly (€/year)	Usage	Cost (€)
Matlab®	€25,000	2 months	€4,167
Matlab®Symbolic Toolbox	€10,000	2 months	€1,667
Total			€5,834

Most of the software used by this project is open source and thus can not be attributed a cost. A list of open source software used by this project is:

- Ubuntu 10.04 GNU/Linux
- VIM
- GNU Screen
- GCC
- Valgrind
- GDB
- cppcheck
- LLVM
- Python
- Bash



However some software does require licensing like in the case of Matlab®. An overview of the full software licensing costs can be seen in Tab.C.2. Open source software used with no associated cost is not listed.

C.3 Equipment

Table C.3: Equipment budget.

Component Name	Product	Unit Cost (€)	Life Expectancy	Usage	Cost (€)
Camera	Flea®2	€1,195	24 months	2 months	€100
Workstation	OptiPlex 980	€1,804	24 months	12 months	€1,033
Misc. Consumables		€100	12 months	12 months	€105
Total					€1,238

The equipment cost is calculated based on the following equation,

$$C = C_0 \frac{U}{E}(1+i)^{U/12} \tag{C.1}$$

where C_0 is the original unit cost (€), U is usage (months), E is life expectancy (months) and i is the inflation rate considered to be 5%.

C.4 Total

Table C.4: Total project budget.

Budget	Percent of Total	Cost (€)
Programming	94.8%	€128,591
Software Licenses	4.3%	€5,834
Equipment	0.9%	€1,238
Total	100%	€135,663

The total budget can be seen in Tab.C.4. It is the sum of the three different parts: code, software licenses and hardware. It can be seen that the majority of the budget is for creating



the applications (97.1%).



D. Solver Manual

The solver is a command line application that provides both a pure Levenberg-Marquadt algorithm and the hybrid genetic algorithm and Levenberg-Marquadt solver. The solver is open source using the GPLv3 license and is called "solver". It can be run by:

```
$ ./solver [OPTION]... DIRECTORY [OUTPUT_AVERAGE OUTPUT_DATA]
```

D.1 Command Line Arguments

These are general command line arguments that can be used regardless of the solver.

```
-S, --solver=SOLVER        Chooses the solver. (default: minpack)
-c, --common=DOF           Common degrees of freedom for all fingers.
-f, --finger=DOFLIST       List of comma seperated degrees of freedom
    for each finger.
-s, --step[=STEPS]         List of comma seperated steps to take.
    Defaults to minimum with no parameters.
-d, --dump=FILE            Sets the file to dump raw data to.
-D, --soldump=DIR          Sets ouptut directory for the complete
    solution information dumper.
-V, --visualize            Enables opengl visualization of results.
```

D.1.1 Levenberg-Marquadt

These are command line arguments specific to the Levenberg-Marquadt solver also called the MINPACK solver. This solver is accessible by passing the "-Sminpack" flag to the application.



```
-i, --init[=FILE]        Initialize the data using the end results or
   FILE if specified.
-t, --tolerance=TOL      Sets the tolerance of the solver.
-I, --iterate=ITER       Sets the maximum amount of iterations.
-n, --noise_rot=VALUE    Sets the standard deviation of the gaussian
   noise to use for rotational error.
-N, --noise_trans=VALUE Sets the standard deviation of the gaussian
   noise to use for translational error.
-m, --noise_angle=VALUE Sets the standard deviation of the guassian
   noise to use for angular error.
-r, --repetitions=VALUE Sets the number of repititions to do (only
   makes sense with noise set).
-h, --help               Displays this message.
```

D.1.2 Genetic algorithm

These are command line arguments specific to the hybrid genetic algorithm and Levenberg-Marquadt solver also called the GA solver. This solver is accessible by passing the "-Sga" flag to the application.

```
-c, --common=DOF         Common degrees of freedom for all fingers.
-f, --finger=DOFLIST     List of comma seperated degrees of freedom
   for each finger.
-s, --step=STEPS         List of steps to take. Defaults to minimum
   with no parameters.
-t, --threaded           Enable threading. (default: false)
-N, --minpack            Uses minpack so that each point is a local
   minima. (default: false)
-i, --inverse            Uses the inverse fitness function (1/fitness)
    instead of linear maximization. (default: false)
-E, --differential       Uses differential evolution. (default: false)
-p, --popsize=SIZE       Sets the population size. (default: 1000)
-C, --crossover=CR       Sets the crossover factor. (default: 0.1)
-m, --mutation=MUT       Sets the mutation factor. (default: 0.5)
-M, --migration=MIG      Sets the migration factor. (default: 0.01)
```



```
-A, --archipelago=NUM    Sets the number of islands in the archipelago
    . (default: 1)
-g, --generations=GEN     Sets the number of generations. (default:
    1000)
-F, --fitness=FIT         Sets the fitness target. (default: 0. [off])
-T, --time_limit=TIME     Sets the maximum time to run in hours. (
    default: off)
-h, --help                Displays this message.
```

D.2 Input Format

The solver uses input files to define the model to be used. These files can be created from synthetic data using the "gen_data" application provided by the solver. For a complete hand model description the following files are needed:

- anglesTest.txt
- axes.txt
- fourthFK.txt
- indexFK.txt
- middleFK.txt
- thirdFK.txt
- thumbFK.txt

The following symbols will aid in legibility:

r Number of revolute joints.

r_i Number of revolute joints in finger i.

m Number of frames.

It is important for the degrees of freedom and joints set in the solver match the ones in the dataset provided.



D.2.1 Angles

The angles are defined in *anglesTest.txt* as rotations of each joint in radians and are formatted as such:

```
{joint 0 angle 0, joint 0 angle 1 ... joint 0 angle m}
{joint 1 angle 0, joint 1 angle 1 ... joint 1 angle m}
...
{joint r angle 0, joint r angle 1 ... joint r angle m}
```

D.2.2 Axes

The axes are expressed in Plücker coordinates in *axes.txt* and are formatted as such:

```
{{axis 0 orientation}, {axis 0 moment}}
{{axis 1 orientation}, {axis 1 moment}}
...
{{axis r orientation}, {axis r moment}}
```

D.2.3 Forward Kinematics

The poses are represented as 4x4 homogeneous transformation matrix in 5 files, one for each finger:

- fourthFK.txt
- indexFK.txt
- middleFK.txt
- thirdFK.txt
- thumbFK.txt

Each file is formatted as



{joint 0 pose 0}
{joint 0 pose 1}
...
{joint 0 pose m}
{joint 1 pose 0}
{joint 1 pose 1}
...
{joint 1 pose m}
{joint r_i pose 0}
{joint r_i pose 1}
...
{joint r_i pose m}

D.3 Synthetic Data Generator

The synthetic data generator is a command line application called "gen_data" and can be called with:

```
$ ./gen_data
```

D.3.1 Command Line Arguments

```
-p, --poses=POSES        Number of poses to generate. (default: 501)
-z, --zero               Initializes first pose to zero angle value. (
   default: false)
-n, --nolimits           Disables explicit limit setting and uses full
    range. (default: false)
-s, --smooth             Does a smooth movement from lower limit to
   upper limit instead of random. (default: false)
-h, --help               Displays this message.
```

D.3.2 Input

The model of the hand is coded directly into the engine and can be easily modified. It uses Denavit-Hartenberg parameters. The following definition is given for a joint in a joint:



```
/**
 * @brief Represents a link in a serial kinematic chain.
 *
 * Each link is done as:
 *
 * Zdisp( theta, d ) . Xdisp( alpha, a )
 *
 * Where theta is the rotation theta * displacement theta.
 */
typedef struct kin_link_s {
    double theta; /**< Rotation along XY plane respect the last. */
    double d;     /**< Distance along the Z axis based on the last. */
    double alpha; /**< Rotation along ZY plane respect the last. */
    double a;     /**< Distance along the X axis based on the last. */
} kin_link_t;
```

A kinematic serial chain is just an array of r joints.

