

Inter-Finger Coordination in Robot Hands via Mechanical Implementation of Principal Components Analysis

by

Christopher Yeates Brown

Bachelor of Science, Mechanical Engineering
University of Maryland, College Park, 2004

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2007

© 2007 Massachusetts Institute of Technology
All rights reserved

1

Signature of Author

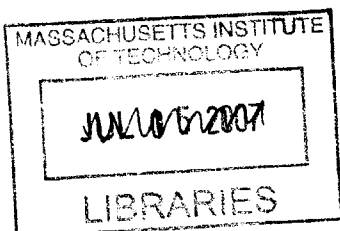
Department of Mechanical Engineering
May 21, 2007

Certified by

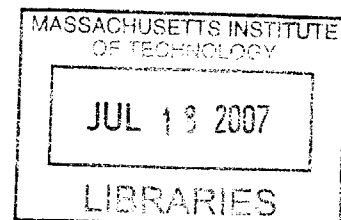
✓ H. Harry Asada
End Professor of Mechanical Engineering
Supervisor

Accepted by

.....
Illit Anand
Chairman, Department Committee on Graduate Students



PARKER



Inter-Finger Coordination in Robot Hands via Mechanical Implementation of Principal Components Analysis

by

Christopher Yeates Brown

Submitted to the Department of Mechanical Engineering
on May 21, 2007 in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Mechanical Engineering

ABSTRACT

Postural synergies describe characteristic patterns of actuation in human hands arising from biomechanical constraints, physical tendon coupling, and neurological control schemes. Often, a small number of synergies contain much of the information required to describe an entire human hand posture, with 80% or more of the total information encoded in only two component values. Synergies have commonly been used to identify hand shapes with minimal processing power. However, they can also be used to recreate postures in robot hands, by allowing a mechanical implementation of inter-finger coordination. This can provide benefits of reduced cost, compact size, and decreased actuator count.

In this paper, a novel mechanism is proposed to drive a dexterous, versatile, 17 degree-of-freedom robot hand using only two DC motors. Posture data was collected with a dataglove, and analyzed using principal components analysis to determine the postural synergies. The synergies are then mechanically hardwired into the driving mechanism, resulting in a concept dubbed eigenpostures. Two eigenpostures effectively recreate the entire posture set.

Several observations and suggestions are presented on tendon-drive robotic hand design in general, and also specifically targeted towards synergy- or eigenposture-based design. Avenues for further research into synergy mechanism design are proposed, including a powerful concept incorporating k-means clustering with principal components analysis to distinguish between high-precision and low-precision tasks, and greatly reduce overall error.

Thesis Supervisor: H. Harry Asada
Title: Professor of Mechanical Engineering

Contents

1. INTRODUCTION.....	13
1.1. Considerations in Robot Hand Design.....	13
1.2. Hand Design in Home Robotics	14
1.3. Synergies as Design Inspiration.....	14
1.4. References.....	16
2. SYNERGIES.....	17
2.1. Introduction to Synergies.....	17
2.2. Applications for Synergies.....	19
2.3. References.....	20
3. PRINCIPAL COMPONENTS ANALYSIS FOR POSTURAL SYNERGIES	23
3.1. Overview	23
3.2. PCA in Robot Hands.....	23
3.3. Principal Components as Eigenpostures	24
3.4. References.....	25
4. MECHANISM DESIGN.....	27
4.1. Concepts.....	27
4.2. Design Implications of Eigenposture Mechanisms.....	31
5. DATA COLLECTION AND ANALYSIS	35
5.1. Collecting Posture Data	35
5.2. Principal Components Analysis.....	40
5.3. Converting to Tendon-Space	40
5.4. References.....	44
6. EIGENPOSTURE ANALYSIS AND POSTURE RECONSTRUCTION.....	45
6.1. Eigenposture Interpretation.....	45
6.2. Posture Reconstruction	47
6.3. Maximum Reconstruction Errors.....	50
6.4. References.....	51
7. PROTOTYPE DESIGN.....	53
7.1. Initial Design Observations.....	53

7.2.	Design for Manufacturing, Design for Assembly	54
7.3.	First Prototype.....	55
7.4.	Second Prototype	56
7.5.	Final Prototype.....	59
7.6.	Prototype Performance.....	63
7.7.	References	64
8.	IMPROVEMENTS TO THE EIGENPOSTURE DRIVE.....	65
8.1.	Introduction.....	65
8.2.	Multiple Eigenpostures	65
8.3.	Compliant Tendons.....	65
8.4.	Nonlinear Eigenposture Pulleys.....	66
8.5.	Reconfigurable Posture Groups Using Clustering Analysis	67
8.6.	References.....	71
9.	CONCLUSIONS.....	73
	APPENDIX A – DERIVATION OF PRINCIPAL COMPONENTS ANALYSIS	77
	APPENDIX B – SELECTED MATLAB FILES	79

List of Figures

Figure 2.1: Branching, coupled tendon structure of human hands.	17
Figure 2.2: Joint coordination in making a fist.	20
Figure 4.1: Traditional bevel-gear differential.....	27
Figure 4.2: Mechanism to actuate any multiple of a vector with tendons.	28
Figure 4.3: Mechanism to add two linear displacements.....	30
Figure 4.4: Mechanism for implementing principal components analysis in robot hands.	31
Figure 4.5: Simplified tendon-drive finger schematic.	31
Figure 5.1: Cyberglove in use to capture posture data.....	35
Figure 5.2: Definitions of joint angles.	36
Figure 5.3: Cyberglove captures of the 14 posture subset.	38
Figure 5.4: Principal components analysis of the posture matrix shows that over 80% of the total information can be explained using only two eigenpostures.	40
Figure 5.5: 3-D solid model of robot hand. Adapted from [2].	41
Figure 6.1. Eigenposture interpretation.	46
Figure 6.2: Average posture.....	47
Figure 6.3: Average error (across all 15 postures) for each joint angle, using only 2 eigenpostures for actuation.	47
Figure 6.4: Postures reconstructed from 2 eigenpostures.	48
Figure 6.5: Postures reconstructed from 2 eigenpostures.	49
Figure 6.6: Postures reconstructed from 2 eigenpostures.	49
Figure 7.1: The eigenposture mechanism is shown again here for convenience.	53
Figure 7.2: First prototype assembly.	55
Figure 7.3: Second prototype assembly.	56
Figure 7.4: Cross-sectional view of prototype 2 configuration.	57
Figure 7.5: 3-D solid model of prototype 2 base.	58
Figure 7.6: Final Prototype Design.....	59
Figure 7.7: Close-up of final prototype base, showing sliding pulleys and adjustment mechanism.	60

Figure 7.8: Front view of final prototype.....	61
Figure 7.9: Close-up of sliding pulley details and tendon routing.....	62
Figure 7.10: Another view of the prototype, showing both eigenposture shafts simultaneously.	63
Figure 8.1: Clustering of eigenpostures for specific task subsets.....	68
Figure 8.2: Grouping results from k-means clustering.	69
Figure 8.3: Effect of grouping on average posture reconstruction error.....	70
Figure 8.4: Mechanism for reconfigurable eigenpostures.	71

List of Tables

Table 5.1: Definitions of joint angles used.	37
Table 5.2: Posture angle data.	39
Table 5.3: Tendon descriptions.	42
Table 5.4: Tendon-space posture matrix.	43
Table 6.1: Tendon-space eigenpostures.	45
Table 6.2: Maximum joint angle errors and associated postures.	51

List of Symbols

n	Number of degrees of freedom, in tendon-space
P_i	$n \times 1$ posture vector, i^{th} posture
$z_{i,j}$	Scalar tendon displacement for i^{th} posture, j^{th} tendon
N	Number of posture vectors
\bar{P}	$N \times n$ posture matrix, with rows consisting of P_i
\bar{P}_0	Posture matrix \bar{P} , with columns re-centered at a mean of 0
$\hat{\bar{P}}$	Approximation of the posture matrix \bar{P}
$q_{i,k}$	Scalar weight for posture i , eigenposture k
e_k	Eigenposture k
\bar{z}_j	Mean across all values of i for $z_{i,j}$
\bar{z}	Average posture vector, in tendon-space
$d_{k,j}$	Element j of eigenposture k
$\varphi_{i,k}$	Input rotation for i^{th} posture, k^{th} eigenposture shaft
$y_{i,k,j}$	Tendon displacement for i^{th} posture, k^{th} eigenposture shaft, j^{th} tendon
c	Scaling factor for eigenposture shaft
$l_{0,p}$	Unstretched length of tendon at joint p
$l_{i,p}$	Actuated tendon length at joint p , for posture i
r_p	Tendon connection radius at joint p
d_{max}	Maximum value of $d_{i,k}$ across all i 's
d_{min}	Minimum value of $d_{i,k}$ across all i 's
τ_f	Frictional torque at an arbitrary finger joint
IPIP	Proximal interphalangeal joint angle at index finger
IMIP	Middle interphalangeal joint angle at index finger
IDIP	Distal interphalangeal joint angle at index finger
IAB	Index finger abduction angle
MPIP	Proximal interphalangeal joint angle at middle finger
MMIP	Middle interphalangeal joint angle at middle finger
MDIP	Distal interphalangeal joint angle at middle finger
RPIP	Proximal interphalangeal joint angle at ring finger

RMIP	Middle interphalangeal joint angle at ring finger
RDIP	Distal interphalangeal joint angle at ring finger
PPIP	Proximal interphalangeal joint angle at pinky finger
PMIP	Middle interphalangeal joint angle at pinky finger
PDIP	Distal interphalangeal joint angle at pinky finger
TPIP	Proximal interphalangeal joint angle at thumb
TDIP	Distal interphalangeal joint angle at thumb
TAB	Thumb abduction angle
TROT	Thumb rotation angle
T1	1 st thumb tendon displacement, controlling TDIP and TPIP
T2	Thumb abduction tendon
I1	Index finger abduction tendon
T3	Thumb rotation tendon
I2	IPIP, IMIP, IDIP tendon
I3	Index finger proximal joint tendon
M1	MPIP MMIP MDIP tendon
M2	Middle finger proximal joint tendon
R1	Ring finger tendon
P1	Pinky finger tendon
f_k	Vector output function for k^{th} eigenposture shaft. Takes scalar input $\phi_{i,k}$
$C_{k,s}$	s^{th} parameter for function f_k

1. INTRODUCTION

1.1. Considerations in Robot Hand Design

The field of robotics research has seen a huge number of robot hand designs over its history. A wide variety of anthropomorphic hands has been created, as well as many other graspers that are distinctly non-humanlike. Although it seems at times that the subject has been exhausted, there remain still certain niche areas for robot hands in which creative and novel design are desired. In addition, the research required in the development of truly anthropomorphic designs serves to increase our knowledge of biomechanics, and the often complicated control schemes can provide useful insight into the function of the human brain.

Although a number of sophisticated biomimetic designs have been created, none of them has succeeded in duplicated the versatility of the human hand. The current reality of robotic hand design is that the hand must be targeted to a specific application or task set. The task set may be quite large, depending on the complexity of the design, but few schemes come close to matching the dexterity of human hands. Many hands, for example, are primarily intended as grippers, and may focus on force control or grasp stability. Others have been created specifically to recreate abstract gestures, such as the sign language alphabet. High bandwidth designs are seldom required, and it is a rare hand indeed that can play piano or perform complicated manipulation tasks.

The most sophisticated designs, while purporting to be able to replicate any movement of a human hand, have their own unique drawbacks. Common issues with these hands are their bulk and complexity. Two famous designs illustrate this point clearly. The Utah/MIT hand, developed in the 1980's, houses an impressive collection of gears, tendons, and pulleys. While versatile, its form factor makes it impractical for incorporating into a full humanoid robot, and it is most useful as a stand-alone device. In addition, the huge number of moving parts and complicated components can result in issues of reliability [1]. A more recent and advanced design is the Shadow Robot Hand, although it demonstrates some of the same issues. The hand itself is extremely dexterous,

but hidden away out of view in many photographs are the multiple pneumatic tubes, which make up most of the bulk of the machine. In both of the cases above we see that dexterous and versatile hand design can easily lead to an increase in size, number of parts, and cost. If these are important factors in the final intended application, then some tradeoffs in hand function will be required. With this in mind, we will examine an area where such tradeoffs are often essential – home robotics.

1.2. Hand Design in Home Robotics

Science fiction envisions a future with fully functional robotic assistants taking care of all the unpleasant duties of everyday life. While this world is far off, some companies are taking steps towards that goal. In order to perform the tasks of humans, a robot must often have a humanlike form, and robots such as Honda's ASIMO and Mitsubishi's Wakamaru are moving in this direction. However, most robots in the home robotics niche currently have little or no physical contact with objects in their environment, interacting primarily through sound or visual means. Some physical interaction is presently achieved with the use of primitive grippers, but a major challenge in this area is to develop a self-contained hand design capable of performing a wide variety of everyday tasks. Such tasks may include traditional grasping, but also manipulation tasks and interactive gestures to enhance the anthropomorphic effect. To achieve these goals, a versatile, compact, self-contained, and affordable hand design is required.

The research presented in this paper suggests one path for realizing such a design. Since we seek to duplicate human functionality, a reasonable approach is to seek inspiration from biomechanics. Although duplicating biological muscle and all the intricate mechanisms of the body is currently beyond our technological grasp, there are certain aspects of control and design which may improve our own schemes. One such feature, which is the focus of this paper, is the concept of synergies.

1.3. Synergies as Design Inspiration

Synergies will be covered in detail in chapter 2, but a brief introduction is given here. In the most basic sense, synergies refer to coordinated movements and control signals to accomplish a given task. The concept has a long history in physiology and medicine, and

was coined by Bernstein around 1967 [2]. In hands in particular, synergies refer to the fact that the joints of the human hand are not independent; they are coupled in many ways, and at many levels – not only mechanical, but also muscular and neurological. The coupling results in coordination of finger movement which can be exploited in the design of robot hands.

One way we can take advantage of this coupling is to use a dimensional reduction scheme to decrease the complexity of the control reference signals – for example, we may only have to tell the key synergies to turn on or off. Alternatively, and perhaps more importantly to our design goals, we can reduce the number of *actuators* required, thus creating a significant decrease in overall size and cost. Previous studies have shown that as much as 80% of the information in everyday hand tasks can be captured with only 2 variables, so using a small number of actuators could still allow us to perform a variety of simple functions [3]. Synergies have often been used to identify hand shapes in data input devices, but the novel concept of incorporating them into *actuation* schemes has not been fully explored. Prior work at MIT’s d’Arbeloff lab implemented a dimensional reduction algorithm to drive a 12 degree-of-freedom (DOF) robot hand with only 8 independent control signals [4], but the full potential of synergies remains untapped.

In this paper, we use the mathematical technique of principal components analysis to identify synergies in common hand postures, and create a set of independent, combinable hand shapes, which we call eigenpostures. The eigenpostures form the basis of a novel mechanism, which uses only two DC motors to operate a 10 DOF multi-fingered robot hand. The research also includes an analysis of tendon-drive robot hand design, and specifically how it is influenced by synergistic actuation schemes. A prototype of the mechanism and hand is presented, along with several design recommendations for future work. Finally, we investigate additional mathematical techniques, such as clustering algorithms, that may significantly improve the device’s accuracy while keeping the size and cost low, and we examine other applications where synergies can enhance current design. Overall, it is shown that mechanical coupling motivated by biological synergies

represents a promising path to affordable, compact, and versatile hand design for home robotics and a variety of other applications.

1.4. References

- [1] S.Jacobsen, E.Iversen, D.Knutti, R.Johnson, K.Biggers. "Design of the Utah/M.I.T. Dextrous Hand," in Proceedings of the 1986 IEE Conference on Robotics and Automation, pp. 1520-32. Apr. 1986.
- [2] A.Bernstein. "The Coordination and Regulation of Movements," Oxford Pergamon, Vol. 6: 77-92, 1967.
- [3] E.J.Weiss, M.Flanders, "Muscular and postural synergies of the human hand," Journal of Neurophysiology. Vol. 92: 523-535, Feb. 18 2004.
- [4] K.J.Cho, J. Rosemarin, H.H.Asada, "Design of vast DOP artificial muscle actuators with a cellular array structure and its application to a five-fingered robotic hand," in Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp.2214-19. May 15-19, 2006.

2. SYNERGIES

2.1. Introduction to Synergies

As established in Chapter 1, the human hand exhibits significant joint coupling and inter-finger coordination. Some of this behavior arises from biomechanical coupling – see Figure 2.1, for example, which shows a simplified anatomy of the human hand. Unlike many tendon-drive robot hand designs, which commonly utilize one tendon per joint, the tendons of the human hand have a complex, branching structure. In some cases, multiple tendons are controlled by a single muscle, and in others, multiple muscles may alternately control a single tendon. Some robot hands incorporate simple coupling, such as between the finger tips and middle phalanges [1], but none of them duplicate the intricate coordination shown here.

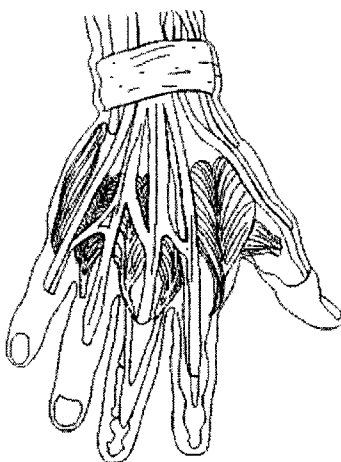


Figure 2.1: Branching, coupled tendon structure of human hands.

The physical coupling of tendons gives rise to common patterns of joint movement, but it is only partially responsible for this phenomenon. In addition, the repetition of many day-to-day tasks can lead the brain to activate muscles in predictable patterns [2]. The combination of tendon coupling and muscle activation patterns leads to so-called muscular and postural synergies, and they occur in other animals as well. For example, an important work on muscular synergies found that just a few common modes of movement could accurately describe the complex kicking of a frog's legs [3]. Muscular synergies traditionally refer to co-activation of muscle groups to achieve a desired motion or posture, and they may be time-varying or static. Muscular synergies have been

observed to develop over time as a task is repeated, and as such may be evidence of a simplified control scheme occurring at the neurological level, so they are a primary interest in brain research [4].

This research, on the other hand, is concerned primarily with *postural* synergies, which describe patterns occurring at the joint displacement level. Postural synergies can be thought of as directions in the coordinate space of a system along which static postures occur; for example, if we define the coordinate space of a hand to be the 21 angles of the finger joints, it has been observed that many of the most common hand shapes can be described with a linear combination of just a few shared vectors in that space [5]. In other words, a coordinate transformation to those vectors can greatly reduce the dimensionality of the set of hand postures. The existence of postural synergies can be attributed to many sources. One obvious cause is biomechanical constraints – the set of *possible* hand postures does not span the entire coordinate space, either because of interference with other joints, or because the tendons or connecting structures themselves do not allow such movement. In addition, the branching tendon structure shown in Figure 2.1 causes coupling between joints, so that often a finger cannot move without repositioning other joints in the process. Neurological control may also be responsible for patterns of joint movement [2,4,5].

One reason human hand movement may use synergies is the similarity of many everyday tasks. Many objects require similar grips, which can lead to developing synergies to simplify this entire set of tasks. On the other hand, many products are also influenced by the ergonomics of the hand, so the objects and tasks we use everyday are tailored to the natural shape of the human hand. In any case, while it may be unclear whether synergies are caused by the similarity of tasks, or whether everyday tasks are similar in response to the observed synergies, the end result is that to mimic the human hand, we may only need to duplicate a small set of coordinated joint movements.

2.2. Applications for Synergies

Since the brain does not directly control joint movement, but rather influences it by muscle displacement, postural synergies are sometimes considered to be less biologically significant [2]. While they thus may not be as interesting for researchers in neuroscience, they can prove extremely useful in computer science and robotics. The most common method of identifying synergies uses principal components analysis (PCA), which is covered in more detail in chapter 3. PCA must begin with a set of vectors to evaluate, which means that any analysis of synergies must start with a subset of desired postures. Depending upon the set of postures in question, different synergies will be used. Complex manipulation tasks, such as spinning a pencil through the fingers, can be recreated with their own unique synergies. If we broaden the scope of tasks to all everyday hand postures, then another set of synergies will be calculated. However, it has been observed that the calculated synergies for different *people*, and similar *tasks* are themselves similar. That is, the synergies are primarily task-specific, and not unique to individual persons. This is promising for robotic hand design, because it means that synergies can be used to recreate common tasks performed by anyone. One very common synergy that occurs in many task sets is the so-called power grip – a co-contraction of all the joints of the hand, as in forming a fist. Figure 2.2 (from a study in [6]) shows a plot of joint movement during this action, which clearly shows that most of the joints are moving in a similar or even identical pattern.

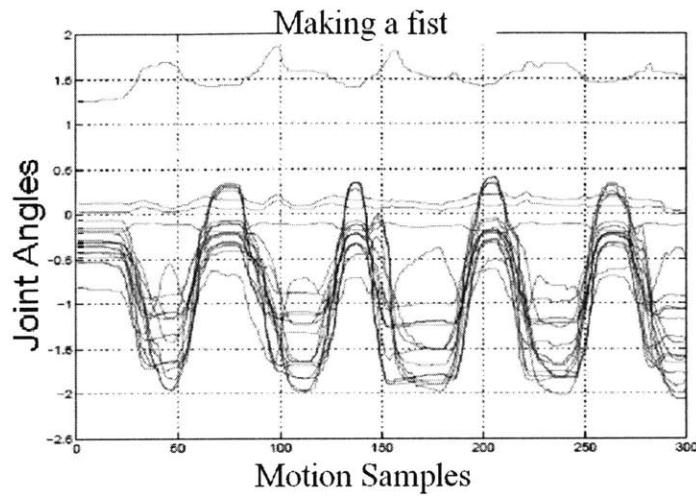


Figure 2.2: Joint coordination in making a fist. Each line on the plot shows a different joint angle, during the process of opening and closing the fist. The alignment of the movements shows that inter-finger coordination is occurring [6].

The commonality of synergies across different people has historically been used only in gesture *recognition*. A prominent study found that up to 80% of information about a posture can be described with only two variables, and used this to identify hand shapes [5]. Using four synergies for identification yielded a better than 90% success rate. In addition, a research group using a data glove as an input device found that their motion tracking algorithms performed significantly better when analyzing only a small set of synergies, rather than all 21 joint angles [1]. However, no major studies have examined the effect of using synergies as the body does – to physically recreate hand postures. Since only a few variables need to be adjusted – specifically, the relative contributions of each synergy – this method can significantly reduce actuator count, leading to lower cost, size, and power requirements. The following chapters present a detailed plan for implementing such a design.

2.3. References

- [1] Y.Wu, J.Y.Lin, T.S.Huang. "Capturing natural hand articulation," Eighth International Conference on Computer Vision (ICCV'01). Vol.2: 426, 2001.
- [2] E.J.Weiss, M.Flanders. "Muscular and postural synergies of the human hand," Journal of Neurophysiology. Vol. 92: 523-535, Feb. 18 2004.

- [3] A.d'Avella, P.Saltiel, E.Bizzi. "Combinations of muscle synergies in the construction of a natural motor behavior," *Nature Neuroscience*. Vol. 6(3): 300-8, Mar. 2003
- [4] J.Pelz, M.Hayhoe, R.Loeber. "The coordination of eye, head, and hand movements in a natural task," *Experimental Brain Research*. Vol. 139(3): 266-77. Aug. 2001.
- [5] M.Santello, M.Flanders, J.F.Soechting. "Postural synergies for tool use," *Journal of Neuroscience*. Vol. 18: 10105-10115, Dec. 1 1998.
- [6] J.Lin, Y.Wu, T.S.Huang. "Modeling the constraints of human hand motion," in *Proceedings of the 2000 IEEE Workshop on Human Motion*. pp. 121-6. Dec. 2000.

3. PRINCIPAL COMPONENTS ANALYSIS FOR POSTURAL SYNERGIES

3.1. Overview

Principal Components Analysis, or PCA, is a dimensional reduction scheme which expresses vectors as linear combinations of a small number of basis vectors. It can be thought of as a coordinate transformation from a large degree-of-freedom (DOF) space to a low DOF space. It is optimal in the sense of reducing the mean squared error in reconstructing the original set of vectors, or in preserving maximal variance in the reproduced vectors [1]. PCA is also useful for determining the relative importance of the basis vectors; the first basis vector, or principal component, is the most significant in terms of variance, and variance decreases with each subsequent basis vector calculated. Thus, by using only the first two or three principal components to reconstruct a given set of vectors, one often finds that the resulting error is sufficiently small for many purposes. The resulting equations are the direct inspiration for much of the material presented in this thesis, and a proof for these equations of PCA is given in Appendix A. In addition, [1] and [2] contain excellent introductions to dimensional reduction in general, and PCA specifically.

3.2. PCA in Robot Hands

As mentioned in chapter 2, our analysis of robotic hand postures must begin with a set of tasks we wish to accomplish. For an n -DOF hand, each task is represented by a posture vector:

$$P_i = [z_{i,1} \quad \dots \quad z_{i,j} \quad \dots \quad z_{i,n}]^T \quad (3.1)$$

Since we are designing a tendon-drive hand, the n elements $z_{i,j}$ of the posture vector have been given as linear tendon displacements. Another type of posture vector could use joint angles as elements instead, but because we will be directly actuating the tendon displacements, here we remain in tendon-space for simplicity. Later, we discuss the implications of transforming to joint-space.

Given a set of N posture vectors, we define the posture matrix:

$$\bar{P} = \begin{bmatrix} P_1^T \\ \vdots \\ P_i^T \\ \vdots \\ P_N^T \end{bmatrix} \quad (3.2)$$

Thus, each row of the posture matrix corresponds to a posture vector, and each column corresponds to a tendon coordinate. PCA allows us to rewrite the posture matrix as the product of two smaller matrices, one consisting of the principal component vectors, and one consisting of the weights for those vectors. The idea is similar to singular value decomposition. The full procedure is detailed in Appendix A, but briefly, the algorithm is as follows: first, we must center the vectors at the origin, so we subtract the mean of each column from each element in that column, to obtain \bar{P}_0 . Next, we calculate $\text{cov}(\bar{P}_0)$, the covariance matrix of the re-centered posture matrix. The principal components are the eigenvectors of this covariance matrix, and their associated eigenvalues represent the relative amount of variance described by each. To calculate the necessary weighting matrix to reconstruct the original posture set, a least squares approach can be used.

3.3. Principal Components as Eigenpostures

The principal components of the posture matrix are the key to reconstructing the entire original set of postures. Because of this, and because they are determined by calculating eigenvectors, we have named them *eigenpostures* in the context of robot hand design (in fact this nomenclature is not entirely original, as a very similar scheme is used in facial recognition algorithms using *eigenfaces* [3]). If we choose to use only a few of the eigenpostures, then we often can approximate the posture matrix with an acceptably low error. As stated earlier, for example, a prominent study has shown that for a wide variety of hand tasks, 80% of the variance is retained using only two eigenpostures. Using four eigenpostures increased the rate to 90%. With this in mind, the bulk of the research in this thesis experimented with methods of implementing a scheme using two eigenpostures. Thus, our posture matrix is given as:

$$\bar{P} \approx \hat{\bar{P}} = \begin{bmatrix} q_{1,1} & q_{1,2} \\ \vdots & \vdots \\ q_{i,1} & q_{i,2} \\ \vdots & \vdots \\ q_{N,1} & q_{N,2} \end{bmatrix} \begin{bmatrix} e_1^T \\ e_2^T \end{bmatrix} + \begin{bmatrix} \bar{z}_1 & \cdots & \bar{z}_n \\ \bar{z}_1 & \cdots & \bar{z}_n \\ \vdots & & \vdots \\ \bar{z}_1 & \cdots & \bar{z}_n \end{bmatrix} \quad (3.3)$$

The vectors e_1 and e_2 are the eigenpostures, constant for all reconstructed postures, and the scalar values $q_{i,1}$ and $q_{i,2}$ are the necessary weights for the eigenpostures in order to reconstruct posture P_i . In addition, (3.3) shows a separate term that we must add to the product. Since we initially subtracted off the column-wise means of the posture matrix during the PCA algorithm, we must add this constant vector $\bar{z} = [\bar{z}_1 \quad \cdots \quad \bar{z}_j \quad \cdots \quad \bar{z}_n]$ to each reconstructed posture, where:

$$\bar{z}_j = \frac{1}{N} \sum_{i=1}^N z_{i,j} \quad (3.4)$$

Comparing (3.2) and (3.3), we reach an important result. The effect of principal components analysis, and our choice of two using only two eigenpostures, leads to the following form for each posture vector:

$$P_i \approx q_{i,1}e_1 + q_{i,2}e_2 + \bar{z} \quad (3.5)$$

It is (3.5) that is the inspiration for a novel robotic hand design, because this simple linear equation is relatively easy to implement mechanically, as we will see in the following chapter.

3.4. References

- [1] L.I.Smith. (Feb. 26, 2002). A tutorial on principal components analysis. Available:
http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
- [2] J.Shlens. (Dec. 10, 2005). A tutorial on principal component analysis. Available:
<http://www.cs.cmu.edu/~elaw/papers/pca.pdf>
- [3] M.A.Turk, A.P.Pentland. "Face recognition using eigenfaces," in Proceedings of the 1991 IEEE Conference on Computer Vision and Pattern Recognition. pp. 586-91. Jun. 3-6, 1991.

4. MECHANISM DESIGN

4.1. Concepts

In the previous chapter, we reached a simple linear equation for reconstructing robotic hand postures:

$$\begin{aligned} P_i &\approx q_{i,1}e_1 + q_{i,2}e_2 + \bar{z} \\ \bar{z} &= [\bar{z}_1 \quad \cdots \quad \bar{z}_n] \end{aligned} \quad (4.1)$$

This equation can be implemented purely mechanically in a straightforward process. By creating a mechanical vector calculator, we let gears and pulleys combine synergies for us, and reduce requirements for control signals and actuators. Notice that the basic mechanical idea of (4.1) is a linear differential, with the inputs $q_{i,1}$ and $q_{i,2}$, and a zero offset of \bar{z} . Traditionally, differentials outputs have been accomplished with bevel gear configurations, as shown in Figure 4.1. However, since such a design can only produce a single output, we would require n such mechanisms for our n -DOF hand, for a total of at least 40 different gears, each having different ratios to reproduce the vectors e_k . This configuration would likely be both bulky and expensive, as well as introducing problems of backlash and the need for precision manufacturing.

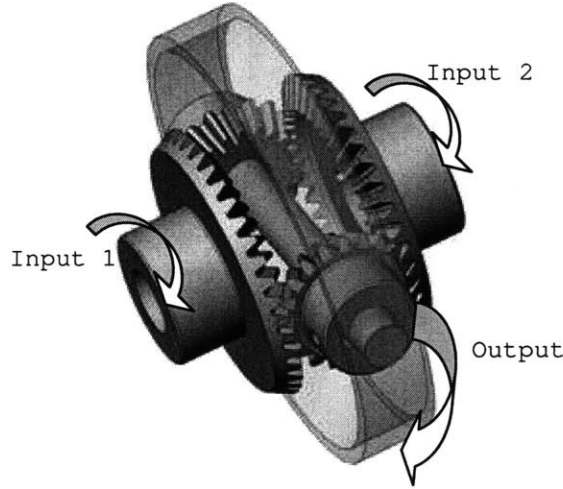


Figure 4.1: Traditional bevel-gear differential.

We require a simpler mechanism which can actuate an entire vector differential at once. Since we are designing a tendon-drive hand, the output must also be linear, unlike the rotary differential in Figure 4.1. Here, the implementation is shown in pieces. The first

step is to actuate the vector multiple $q_{i,k}e_k$. Viewing the arbitrary eigenpostures e_k in its individual elements, we have:

$$e_k = [d_{k,1} \quad \cdots \quad d_{k,j} \quad \cdots \quad d_{k,n}] \quad (4.2)$$

The symbol d has been chosen for the elements because they can be imagined as the diameters of pulleys fixed to a rotating shaft, as shown in Figure 4.2. In the figure, $q_{i,k}$ is represented in the angle of rotation of the shaft, $\phi_{i,k} = 2 \cdot q_{i,k}$. By setting up the mechanism parameters in this fashion, we obtain the tendon displacement

$$y_{i,k,j} = q_{i,k} \cdot d_{k,j}, \quad (4.3)$$

as desired. An important feature of this mechanism is that it can support either sign for $d_{k,j}$ – if any of the components are negative, this is accounted for by wrapping tendons *in the opposite direction*. Assuming some arbitrary tension on the tendons, this system ensures that as the shaft turns, some tendons extend in length, while others shorten.

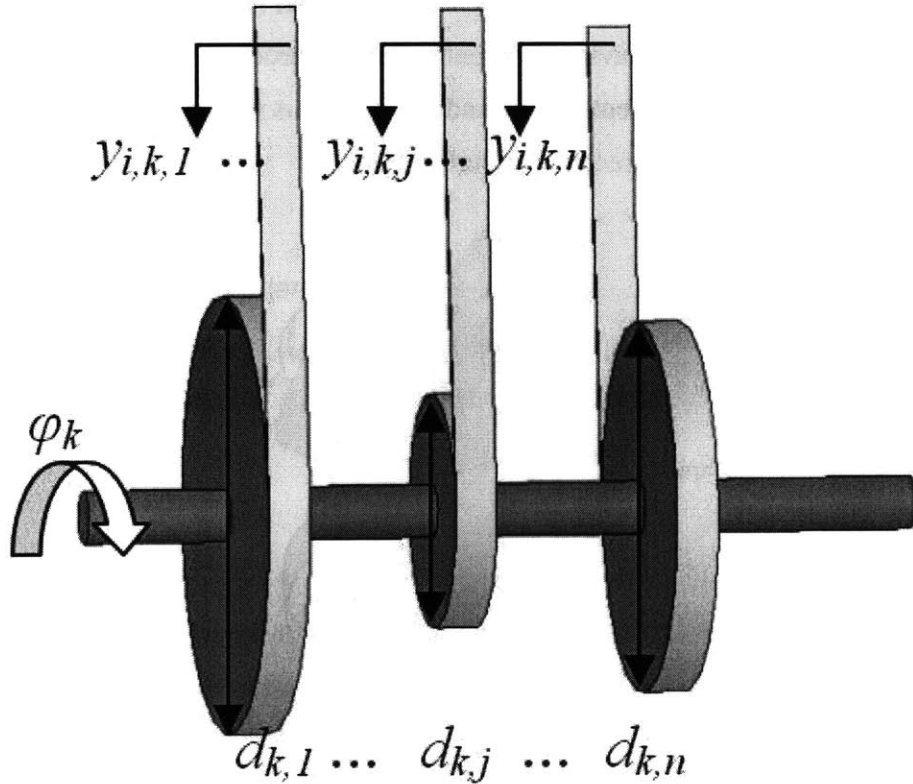


Figure 4.2: Mechanism to actuate any multiple of a vector with tendons. The vector elements are represented by the diameters of the pulleys, which are fixed to the rotating shaft. Note that here, the last vector

element is negative, so the pulley on the right is wrapped in the opposite direction.

More importantly, the mechanism in Figure 4.2 can be scaled by any multiple c . If we want to change some diameter of the shaft, then we multiply the entire vector e_k by c , and multiply $q_{i,k}$ by $1/c$. This ensures that the vector multiple remains constant at $q_{i,k}e_k$. Thus, it is the *ratios* of the elements of e_k that are important, and not their absolute values. Of course, this is unsurprising, since e_k is really an eigenvector, which has no set magnitude, but it is an important consideration when transforming mathematical equations into mechanical structures. Although in pure mathematical thinking we almost never favor the form $\frac{1}{c}q_{i,k} \cdot c \cdot e_k$ over the simplified version, in our mechanical design it allows us great flexibility in form factor (along with strength, force output, and other mechanical properties), by trading off pulley size for angular displacement.

With the mechanism in Figure 4.2, we can implement a vector multiple, but to fully recreate (4.1), we must also be able to add linear vector displacements mechanically. Several linear adders are possible, most using a sliding element, and the proposed design is no exception. Figure 4.3 shows a schematic form for a simple tendon-drive mechanism to add two scalar values.

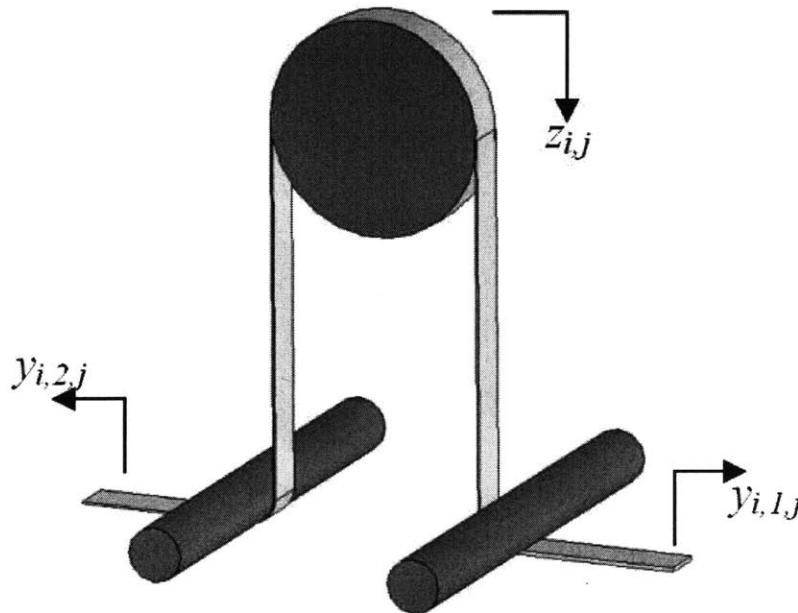


Figure 4.3: Mechanism to add two linear displacements. The lower guide rods are fixed in space. This figure shows the schematic form only. In a real mechanism, output pulley must be constrained to translate in the vertical direction.

The pulley in the figure is free to translate in the vertical direction. The mechanism also winds up scaling the output, so that:

$$z_{i,j} = \frac{1}{2}(y_{i,1,j} + y_{i,2,j}). \quad (4.4)$$

If we attach one of these adding mechanisms to each of the outputs $y_{i,k,jj}$ from Figure 4.2, then we obtain the vector output:

$$\begin{aligned} [z_{i,1} \quad \cdots \quad z_{i,n}] &= \frac{1}{2}([y_{i,1,1} \quad \cdots \quad y_{i,1,n}] + [y_{i,2,1} \quad \cdots \quad y_{i,2,n}]) \\ [z_{i,1} \quad \cdots \quad z_{i,n}] &= \frac{1}{2}(q_{i,1}e_1 + q_{i,2}e_2) \end{aligned} \quad (4.5)$$

If we simply double the right-hand side, by doubling the angular shaft displacements in Figure 4.2, then we will have nearly reconstructed (4.1). All that remains is to account for the zero offset value \bar{z} . This is straightforward, as we simply adjust the tendon lengths so that $[z_{i,1} \cdots z_{i,n}] = \bar{z}$ when the shafts are in their $\phi_1 = \phi_2 = 0$ positions. The complete mechanism schematic is shown in Figure 4.4.

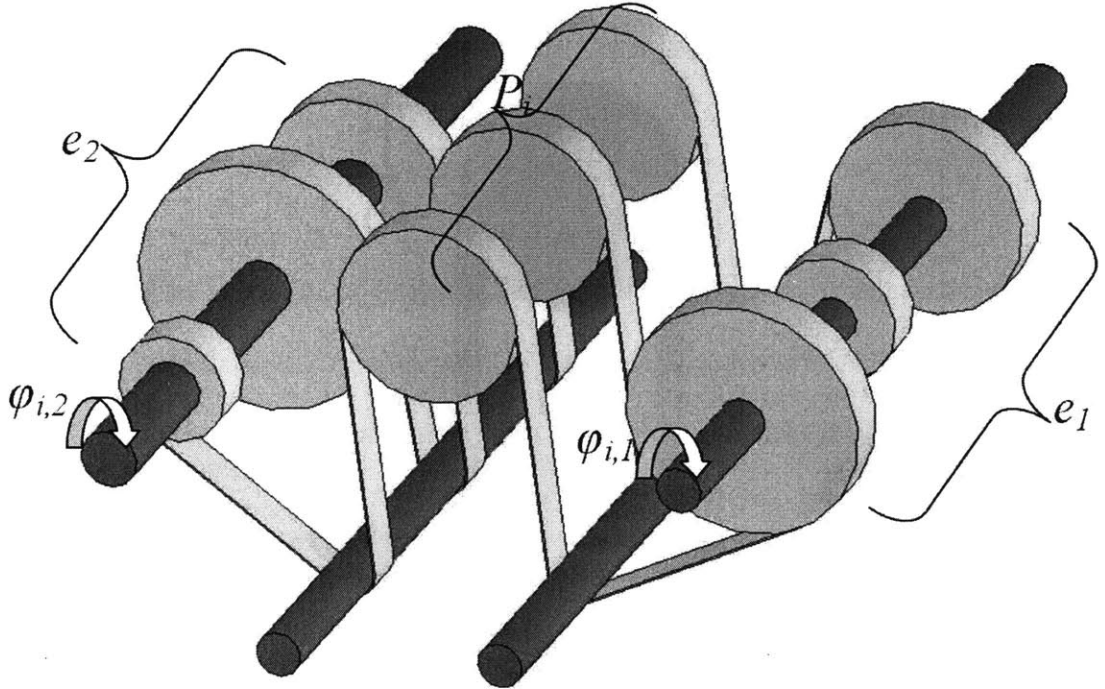


Figure 4.4: Mechanism for implementing principal components analysis in robot hands. Notice the positive and negative component values.

4.2. Design Implications of Eigenposture Mechanisms

In the previous section, we saw a way to mechanically implement the basic reconstruction equation of principal components analysis. Here, we see some ways to refine the method. The most important point is that in the above analysis, we used *tendon-space posture vectors*, as indicated in eq. (3.1). In other words, the posture vectors were defined as a set of tendon displacements. In reality, robotic hand design normally focuses on *joint-space posture vectors*, consisting of the various joint angles of the hand in question. In a tendon drive hand, the joint angles are then converted to tendon-space, using a straightforward kinematic analysis. However, this transformation from joint-space to tendon-space is highly nonlinear, and has a direct impact on the resulting eigenpostures. Consider Figure 4.5, a simplified view of a tendon-drive finger.

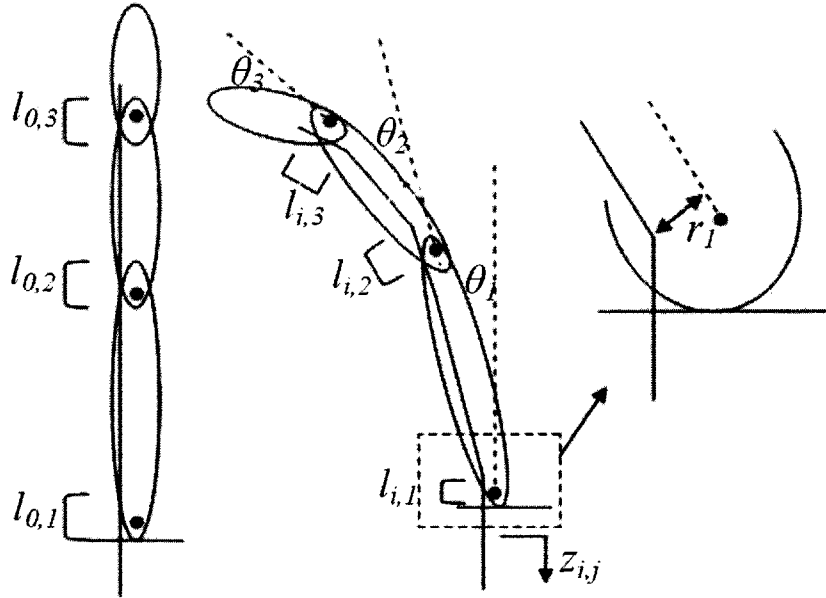


Figure 4.5: Simplified tendon-drive finger schematic. Far left shows the fully extended finger, while the middle shows the flexed finger in a given posture. We calculate the total tendon displacement by determining the changes in the distances l . The parameter r will figure prominently in our design.

The total tendon displacement $z_{i,j}$ required to achieve the middle configuration is given by the sum of the changes in length of l :

$$\begin{aligned} z_{i,j} &= (l_{0,1} - l_{i,1}) + (l_{0,2} - l_{i,2}) + (l_{0,3} - l_{i,3}) \\ z_{i,j} &= f(\theta_1, \theta_2, \theta_3; r_1, r_2, r_3) \end{aligned} \tag{4.6}$$

Here, $l_{0,p}$ represents the unstretched length of tendon at joint p . The tendon displacement $z_{i,j}$ is a nonlinear function of θ , parameterized by the various tendon connection radii r . Thus, by changing the values of r , we can alter the resulting eigenpostures. The main reason this is desirable is to change the form factor of the vector multiplier mechanism in Figure 4.2. For example, generally speaking, decreasing r at a certain joint will decrease the tendon stroke length required for a given value of θ . This means that the pulley from Figure 4.2 associated with that tendon will also decrease in diameter. However, since multiple tendons are running through the same finger, this change in r will affect other pulleys as well. Because of this coupling between tendons, the effect on the pulley ratios is often difficult to anticipate. However, we can run a search algorithm to consider the effect of changes in all the different radii r to find the best form factor for the eigenposture pulley shafts. In general, our goal is to minimize d_{\max}/d_{\min} , since d_{\min} will be determined by strength and torque requirements. The procedure for finding acceptable values of r is detailed in a later section.

We may not wish to focus all our efforts on optimizing the ratio d_{\max}/d_{\min} , because besides their effect on the pulley shafts, the values of r affect other design considerations as well. Most obviously, a large increase in r will lead to an increase in finger width, so there is an upper limit on r to maintain the anthropomorphic look (for our purposes, this limit is about 13mm). There is a lower limit on r as well, because it affects our actuator torque requirements. Specifically, if the resistive torque due to friction at joint p is given by τ_f , then a tendon connected to that joint will have to exert at least $\frac{\tau_f}{r_p}$ to move the joint.

Since some tendons may move multiple joints, several such terms may be summed to determine the total minimum tendon force required, and due to our mechanism design, the actuators must exert enough torque to move all the tendons simultaneously. Thus, decreasing r at a given joint can significantly impact the actuator torque required. In practice, τ_f is determined experimentally, and actuator parameters set accordingly.

The above basic analysis gives us bounds on r , but other factors may weigh in to our selection of a value as well. Since a larger value of r generally leads to a longer stroke

length for a given posture, we can expect a decrease in error sensitivity as well. That is, we will be limited to a certain resolution in tendon displacement based on our selection of actuators and position measurement system, so a larger stroke length will allow us to achieve a smaller error in displacement.

It is evident that the connection radii r have a significant impact on our design. Other finger geometry, such as joint length and width, affect kinematics and design parameters as well. With so many factors affecting our results, it can be difficult to select a starting point. However, since our basic goal is to design an anthropomorphic hand, a reasonable place to begin is in the overall aesthetic quality. After a basic geometry is laid out, we can use mathematical analysis to tweak the design to our specifications. In many cases, we require merely an acceptable solution, not an optimum. The most important point is to remain aware of all the factors which could affect our primary goals of compact, affordable, human-like design. With these factors in mind, we can begin the detailed design process of a robot hand with mechanical implementation of PCA.

5. DATA COLLECTION AND ANALYSIS

5.1. Collecting Posture Data

Before we can begin principal components analysis, we must start with a set of desired postures for our robot hand to perform. For a home robotics application, such a set would normally consist of everyday tasks. A good starting point for such a set could be the postures of the Sollerman Hand Function Test [1], which is made up of 8 postures making up over 80% of daily tasks. For this thesis, however, a set of tasks was provided by the sponsor, Mitsubishi Heavy Industries. These postures consist mainly of grips for common objects, but unsurprisingly, several of them closely resemble postures from the Sollerman set, so we can be confident that they represent a wide spectrum of possible chores for a home robot.

We calculated the joint angles for these postures using an Immersion Corporation Cyberglove. The data glove, shown in Figure 5.1, measures 22 joint angles (including 2 wrist angles) using embedded strain gauges.

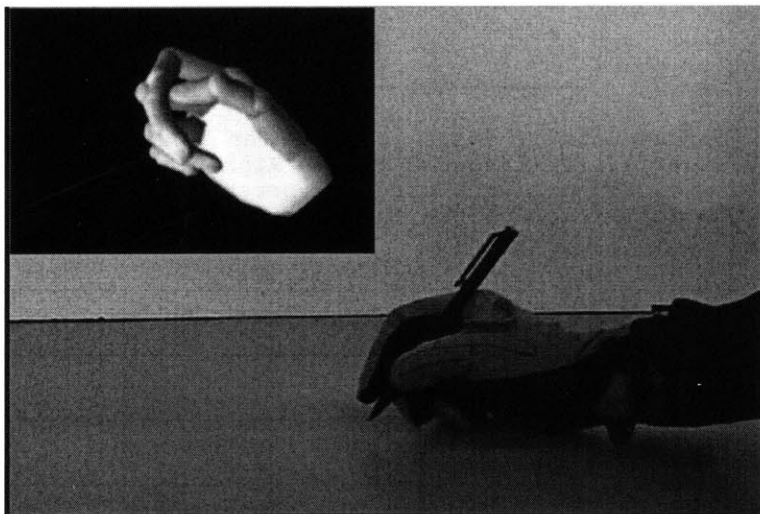


Figure 5.1: Cyberglove in use to capture posture data.

For simplicity, we planned on using only 17 of the 20 finger joint angles in our hand design. The 3 unused angles are the middle, ring, and pinky abduction/adduction. Figure 5.2 defines the nomenclature for the joint angles, and Table 5.1 defines the joint angles

that were used. Figure 5.3 shows the 14 tasks captured by the Cyberglove, with the resulting joint angles in Table 5.2.

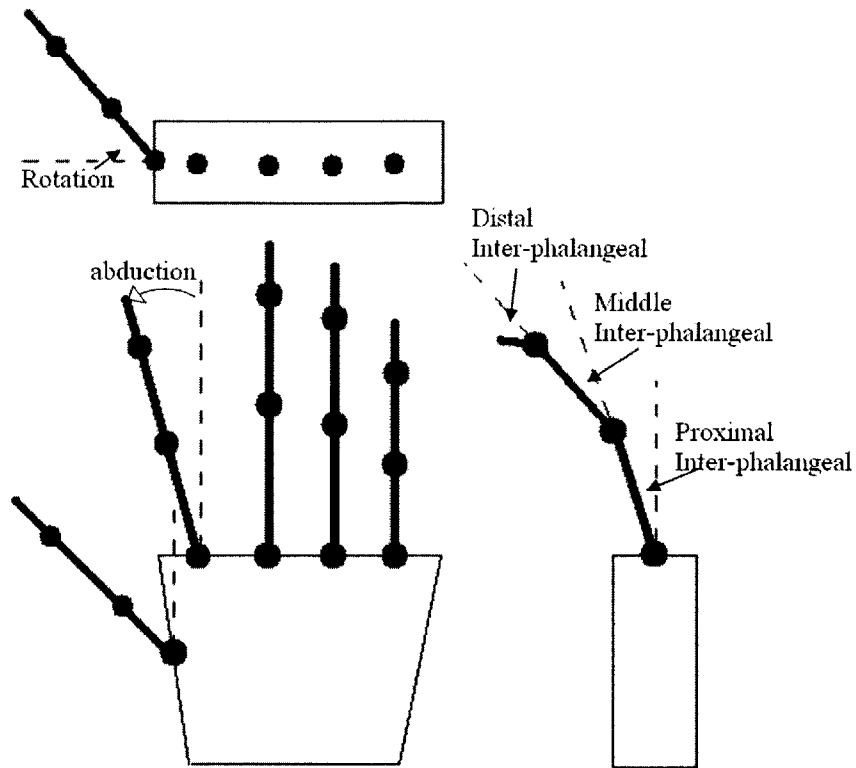


Figure 5.2: Definitions of joint angles. The lower-left figure shows a simplified view of a five-fingered hand. Right image shows a side view of a single flexed finger, with three important joint angles. The top image shows the definition of the thumb rotation angle.

Angle Name	Angle Description
IPIP	Index finger proximal inter-phalangeal
IMIP	Index finger middle inter-phalangeal
IDIP	Index finger distal inter-phalangeal
IAB	Index finger abduction
MPIP	Middle finger proximal inter-phalangeal
MMIP	Middle finger middle inter-phalangeal
MDIP	Middle finger distal inter-phalangeal

Angle Name	Angle Description
RPIP	Ring finger proximal inter-phalangeal
RMIP	Ring finger middle inter-phalangeal
RDIP	Ring finger distal inter-phalangeal
PPIP	Pinky finger proximal inter-phalangeal
PMIP	Pinky finger middle inter-phalangeal
PDIP	Pinky finger distal inter-phalangeal
TPIP	Thumb proximal inter-phalangeal
TDIP	Thumb distal inter-phalangeal
TAB	Thumb abduction
TROT	Thumb rotation

Table 5.1: Definitions of joint angles used. Notice that abduction for only the index finger and thumb was considered, and that the thumb has no middle inter-phalangeal joint.

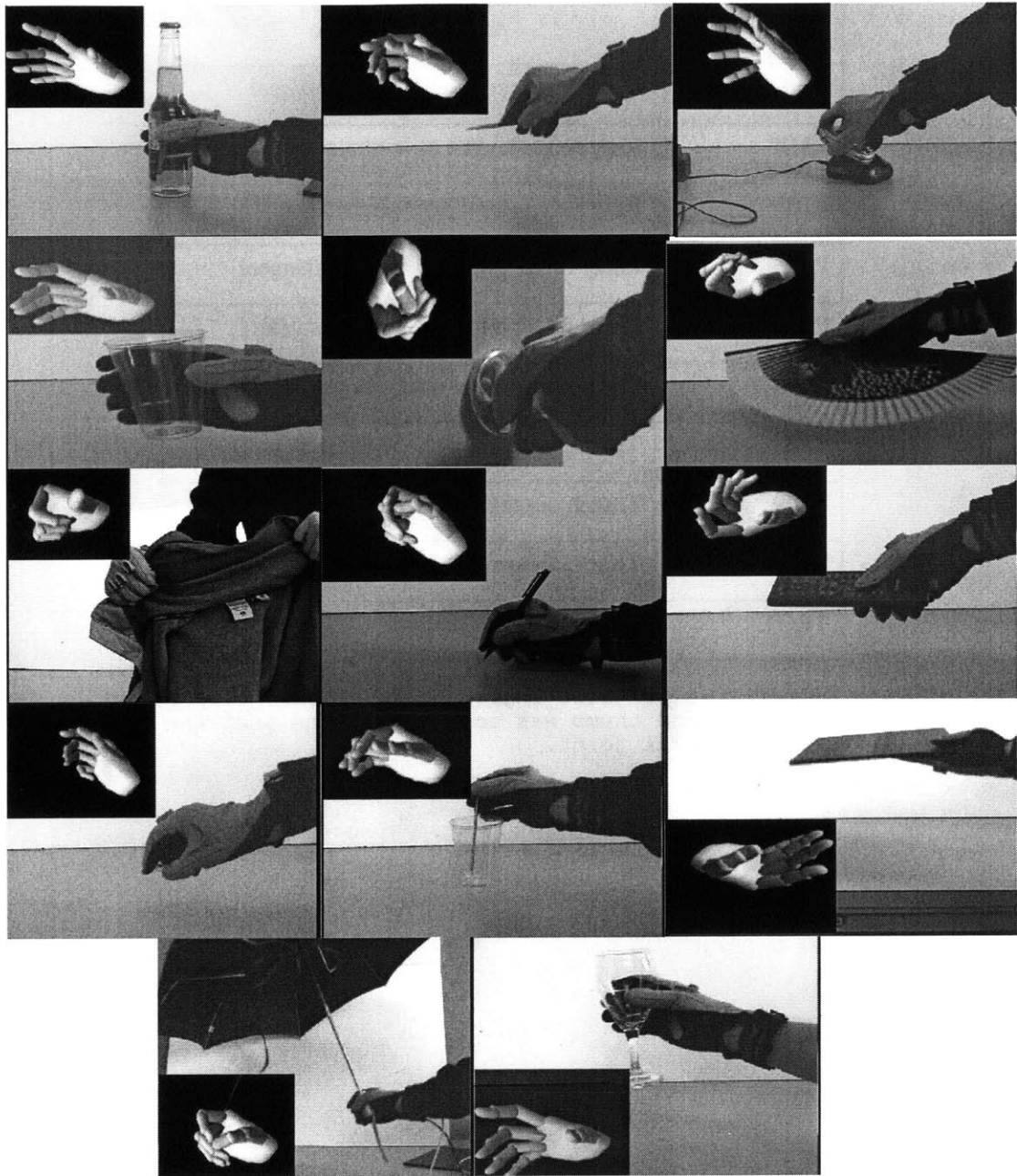


Figure 5.3: Cyberglove captures of the 14 posture subset. From top-left to lower-right: bottle, brush, cell phone, cup, doorknob, fan, jacket, pen, remote control, shogi (Japanese chess) piece, toothpick, tray, umbrella, wine glass. The bottle grip closely resembles the more generic "transverse volar grip" from the Sollerman hand function test [1], while the brush, doorknob, pen, shogi, remote control, toothpick, and tray grips resemble the lateral pinch, spherical volar grip, tripod pinch, five-fingered pinch, diagonal volar grip, pulp pinch, and extension grip, respectively. These similarities are reiterated in Table 5.2.

Posture Name	IDIP	IMIP	IPIP	MDIP	MMIP	MPIP	RDIP	RMIP	RPIP	PDIP	PMIP	PPIP	TDIP	TPIP	TAB	IAB	TROT
FLAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bottle (transverse volar)	30	45	40	30	45	45	30	45	45	30	45	45	15	30	0	10	90
Brush (lateral pinch)	45	70	80	45	70	80	45	70	80	45	70	80	10	40	30	0	60
Cell phone	15	30	17	15	20	60	20	25	65	20	25	65	30	20	0	10	60
Cup	21	35	40	20	28	60	20	25	68	20	25	65	10	20	0	0	90
Doorknob (spherical volar)	30	20	10	30	20	10	30	20	10	30	25	25	30	20	0	0	45
Fan	35	70	80	35	70	80	37	70	80	38	70	80	5	30	45	0	60
Jacket	50	82	85	50	82	85	50	82	85	50	82	85	35	40	0	0	75
Pen (tripod pinch)	30	22	50	40	50	65	50	75	85	50	75	85	15	35	20	10	60
Remote Control (diagonal volar)	0	0	0	15	20	60	20	25	68	35	40	80	0	0	28	10	55
Shogi Piece (five-fingered pinch)	30	28	55	40	50	65	50	75	85	50	75	85	15	30	20	0	65
Toothpick (pulp pinch)	30	27	55	20	35	35	15	30	30	10	25	22	15	30	20	0	65
Tray (extension grip)	25	40	35	25	40	40	25	40	40	25	40	40	15	30	0	10	90
Umbrella	25	27	55	30	22	62	30	27	55	30	27	55	15	30	5	0	75
Wine Glass	15	30	17	15	20	60	40	65	75	40	65	75	10	40	20	10	52

Table 5.2: Posture angle data. All measurements are given in degrees. The FLAT posture has been added to the set from Figure 5.3 for completeness. The Cyberglove is a difficult instrument to calibrate

perfectly, so the roundness of some of the numbers here indicates our post-data-collection efforts to make the digital postures match the real-world postures. This was done through the use of a 3D solid model of the hand, shown later in this chapter.

5.2. Principal Components Analysis

Table 5.2 is in fact the joint-space posture matrix, defined in chapter 3. Although a variety of grips are included, the data can be described with only a few eigenpostures, as desired. Of course, we eventually will need to perform PCA in tendon-space to calculate the tendon-space eigenpostures, but a check now confirms that our idea of using only two actuators should produce satisfactory results. The effect of the number of eigenpostures used is shown in Figure 5.4.

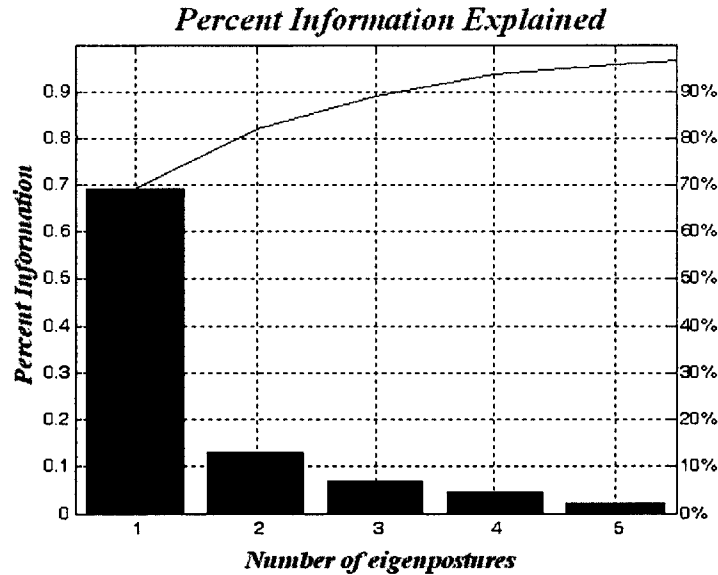


Figure 5.4: Principal components analysis of the posture matrix shows that over 80% of the total information can be explained using only two eigenpostures.

5.3. Converting to Tendon-Space

Once the joint-space posture matrix has been determined, the transformation to tendon-space begins. As mentioned in chapter 4, this is an iterative process involving continuously updating the robot finger geometry. We begin this procedure with a 3-D solid model of the hand. The initial geometry is purely aesthetic, with the primary goal of establishing a human-like appearance. Each joint was given a tendon connection radius r of approximately 7mm as an initial value, although this parameter is modified in

the next step. The 3-D model was modified only slightly from previous work in the MIT d'Arbeloff lab on anthropomorphic hands [2], and is shown in Figure 5.5.

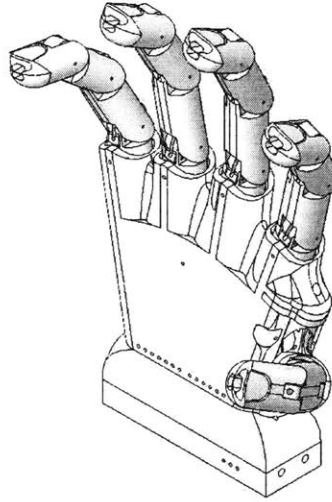


Figure 5.5: 3-D solid model of robot hand. Adapted from [2].

With a solid model available, we can convert the joint-space posture matrix to tendon-space. For this research, we chose to underactuate the hand, using only 10 tendons to drive the 17 joint angles. This decision was made primarily to ease actuator requirements, since each additional tendon adds friction and other resistive torques. However, underactuating the joints can help in gripping tasks as well. By underactuating, and incorporating some compliance, we allow the hand to conform to the shape of an object being gripped, rather than rigidly define the posture shape. The effect of compliance is explored further in chapter 8. Table 5.3 lists the 10 tendon names, along with the joint angles actuated by each.

Tendon Name	Attachment point	Associated Joint Angles
T1	Tip of thumb	TDIP, TPIP
T2	Base of thumb, controlling abduction	TAB
I1	Base of index finger, controlling abduction	IAB
T3	Base of thumb, controlling rotation	TROT
I2	Tip of index finger	IPIP, IMIP, IDIP

Tendon Name	Attachment point	Associated Joint Angles
I3	Base of index finger	IPIP
M1	Tip of middle finger	MPIP, MMIP, MDIP
M2	Base of middle finger	MPIP
R1	Tip of ring finger	RPIP, RMIP, RDIP
P1	Tip of pinky finger	PPIP, PMIP, PDIP

Table 5.3: Tendon descriptions. The 17-DOF hand is underactuated, driven by only 10 tendons. The table shows which angles are influenced by each tendon.

Using the solid model geometry, the tendon layout, and the kinematics described in chapter 4, we calculate the tendon-space posture matrix. Performing principal components analysis on the results, we can compute the maximum-to-minimum ratio of eigenposture elements, i.e. d_{max}/d_{min} , from chapter 4. There will be 2 such ratios, one for each eigenposture. Using the maximum of these 2 ratios as a cost function, we seek to minimize the ratio by updating the values $r_1 \dots r_{17}$. The bound on r is

$6.25mm \leq r_p \leq 12.7mm$, as determined by the geometric and actuator constraints described in chapter 4.

One final important detail must be considered. Although updating the radii r will allow some manipulation of the eigenpostures, it is often the case that the important ratio d_{max}/d_{min} is still excessively large, on the order of 50-100. We planned on a minimum shaft size of 4.75mm for sufficient torsional stiffness, so the maximum acceptable ratio is about 6-10 in order to avoid extremely large pulleys; the motivation, after all, is to design a *compact* robotic hand. However, such a ratio is easily acquired, by a slight modification to the algorithm. The very large ratios arise when the d_{min} element of the eigenpostures is extremely *small*. This has a significant physical meaning, since a small pulley will have very little effect on tendon displacement. Thus, we can simply ignore such an excessively small value, simply tying off the summing mechanism (Figure 4.3) to a static point on that side. For mathematical purposes, we change the small value $d_{k,j}$ to 0, and use the next smallest value as d_{min} . We must simply be careful not to change the

same j^{th} element from both eigenpostures to 0, or we will not be able to influence the j^{th} tendon. In practice, this change of an element to 0 occurs only once or twice, and quickly brings the ratio down to an acceptable level. The entire optimization algorithm is performed using a numerical gradient descent approach, and is detailed in Appendix B. The minimum ratio found was 5.73, and the resulting tendon-space posture matrix is given in Table 5.4.

Posture	T1	T2	I1	T3	I2	I3	M1	M2	R1	P1
FLAT	0	0	0	0	0	0	0	0	0	0
Bottle	5.70	0	2.21	12.13	14.07	5.75	18.13	5.62	13.29	13.29
Brush	6.46	6.63	0	8.09	24.26	11.51	29.27	9.99	21.61	21.61
Cell Phone	6.09	0	2.21	8.09	7.43	2.44	13.24	7.49	12.19	12.19
Cup	3.80	0	0	12.1	11.96	5.75	15.43	7.49	12.52	12.19
Doorknob	6.09	0	0	6.06	6.98	1.43	8.66	1.24	6.64	8.86
Fan	4.56	9.95	0	8.09	23.15	11.51	28.16	9.99	20.72	20.83
Jacket	9.32	0	0	10.11	26.86	12.23	32.90	10.61	24.04	24.04
Pen	6.37	4.42	2.21	8.09	12.96	7.19	22.76	8.12	23.27	23.27
Remote Control	0	6.19	2.21	7.41	0	0	13.24	7.49	12.52	17.17
Shogi Piece	5.70	4.42	0	8.76	14.27	7.91	22.76	8.12	23.27	23.27
Toothpick	5.70	4.42	0	8.76	14.2	7.91	13.73	4.37	8.31	6.31
Tray	5.70	0	2.21	12.1	12.24	5.03	15.93	4.99	11.63	11.63
Umbrella	5.70	1.10	0	10.11	13.68	7.91	15.57	7.74	12.41	12.41
Wineglass	6.46	4.42	2.21	7.01	7.43	2.44	13.24	7.49	19.94	19.94

Table 5.4: Tendon-space posture matrix. The figures given are the linear tendon displacements to best recreate the postures from Figure 5.3. All measurements given in mm.

5.4. References

- [1] C.Sollerman, A.Ejeskar. "Sollerman hand function test. A standardised method and its use in tetraplegic patients," *Scandinavian Journal of Plastic and Reconstructive Surgery and Hand Surgery*. Vol. 29: 167-76, 1995.
- [2] K.J.Cho, H.H.Asada, "Segmentation architecture of multi-axis SMA array actuators inspired by biological muscles," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and System*, pp.254-259.

6. EIGENPOSTURE ANALYSIS AND POSTURE RECONSTRUCTION

6.1. Eigenposture Interpretation

In chapter 5, we determined the tendon-space posture matrix. In turn, this yields the tendon-space eigenpostures, which are given in Table 6.1.

	T1	T2	I1	T3	I2	I3	M1	M2	R1	P1
First Eigenposture	6.79	6.35	0	6.53	30.05	15.34	36.34	11.61	28.57	27.27
Second Eigenposture	-6.36	10.88	6.35	-6.37	-36.35	-17.68	-6.35	6.35	23.52	31.83
Average posture	5.18	2.77	0.89	8.47	12.64	5.94	17.54	6.72	14.83	15.14

Table 6.1: Tendon-space eigenpostures. Although strictly speaking, eigenpostures represent only ratios of tendon displacement, and thus have no units, for our purposes they also represent pulley diameters, so here we have scaled the values for a minimum diameter of 6.35mm. All measurements are given in mm. Recall that a negative component value here signifies that the tendon must be wrapped in the opposite direction (see Figure 4.2 and discussion for full details).

A few details should be noted. All the components in the first eigenposture have positive sign. This means that the effect of the first eigenposture is a co-contraction of all joint angles. This is a common result in postural synergies [1]. The second eigenposture, on the other hand, contains mixed sign, indicating that some joints will be flexing while others are extending. Thus, the eigenpostures themselves are physically meaningful. The primary mode causes a co-contraction (or co-extension) of the joints to position the fingers roughly around the desired positions, while the secondary mode causes a more subtle *repositioning* of the fingers to refine the posture. Co-contraction, or power grip, is a well-known phenomenon, but without PCA it is unlikely that the second eigenposture would be obvious as a pattern of movement. Since we can transform from tendon-space back to joint-space using the inverse kinematics, a graphical interpretation can be provided here. Figure 6.1 shows the effect of the 2 eigenpostures working independently, as the scalar weight $q_{i,k}$ is varied from minimum to maximum. The average posture has been added to each figure already, and is shown separately in Figure 6.2.

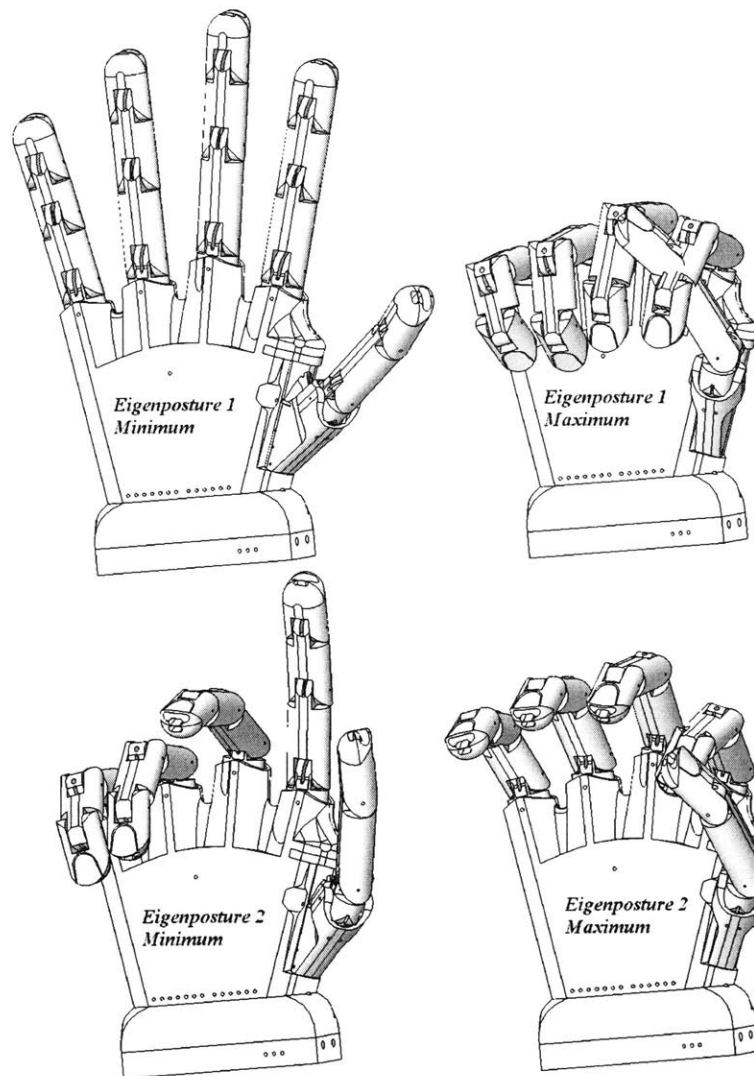


Figure 6.1. Eigenposture interpretation. This figure demonstrates the effect of activating each eigenposture independently. The first eigenposture acts as a co-contraction of all joint angles, while eigenposture 2 repositions the joints to clarify the position. All figures include the effect of the average posture.

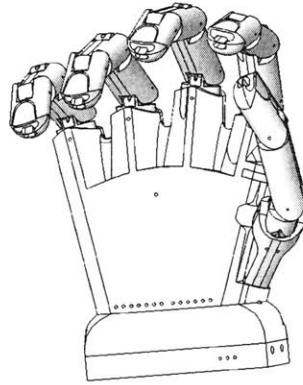


Figure 6.2: Average posture. Every reconstructed posture includes this position added to its joint angles.

6.2. Posture Reconstruction

With all the details in place, we can now simulate the results of the eigenposture actuation scheme. Figure 6.3 shows the residual error in reconstructing the original posture set using only 2 actuators.

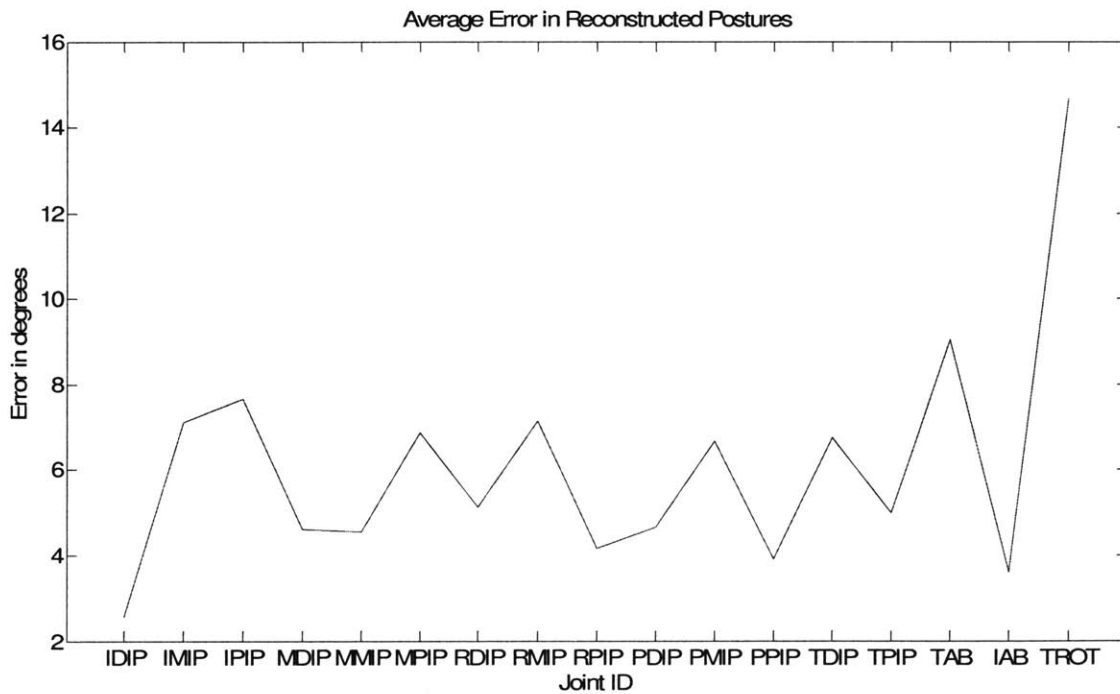


Figure 6.3: Average error (across all 15 postures) for each joint angle, using only 2 eigenpostures for actuation. The average overall error is 6.13 degrees.

For most joint angles, the results are very promising. The average error overall is only 6.13 degrees, which is an acceptable tradeoff for the huge decrease in actuator count. However, it is difficult to judge from Figure 6.3 the real effect of the eigenposture strategy. Since one of our goals is to create human-like postures, we must take a qualitative perspective as well. Figures 6.4 - 6.6 show two rows each. The top row consists of the original posture, captured from the data glove and corrected in the 3-D model. The bottom row shows the reconstructed posture obtained using only two actuators.

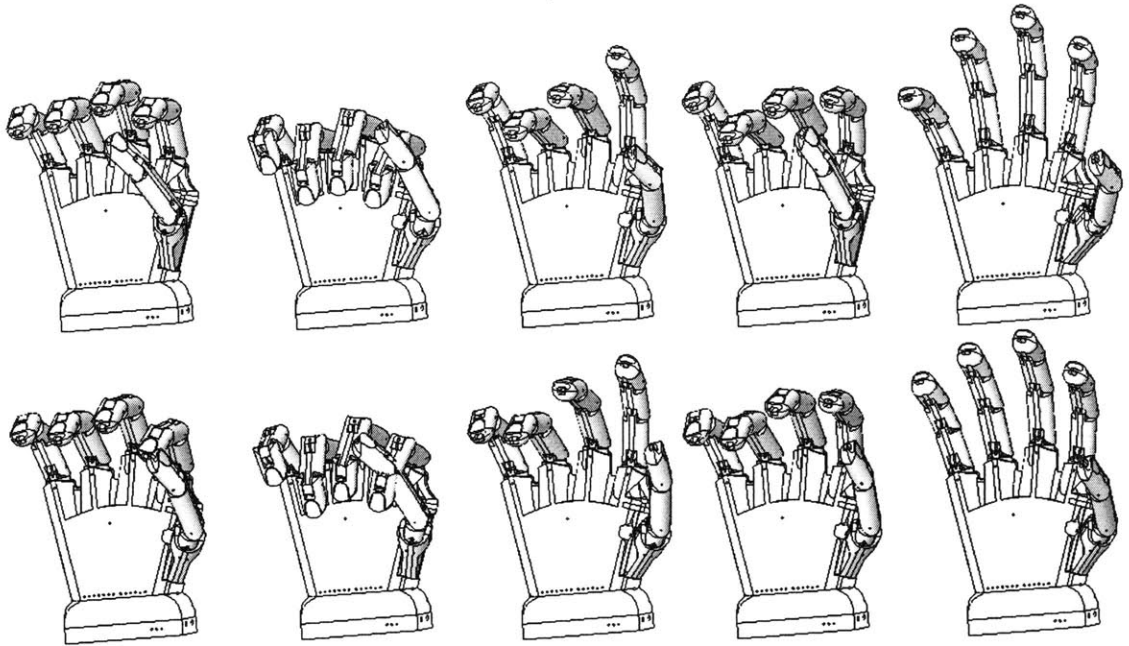


Figure 6.4: Postures reconstructed from 2 eigenpostures. The top row shows the original posture, while the bottom shows the approximation. From left to right: bottle, brush, cell phone, cup, doorknob.

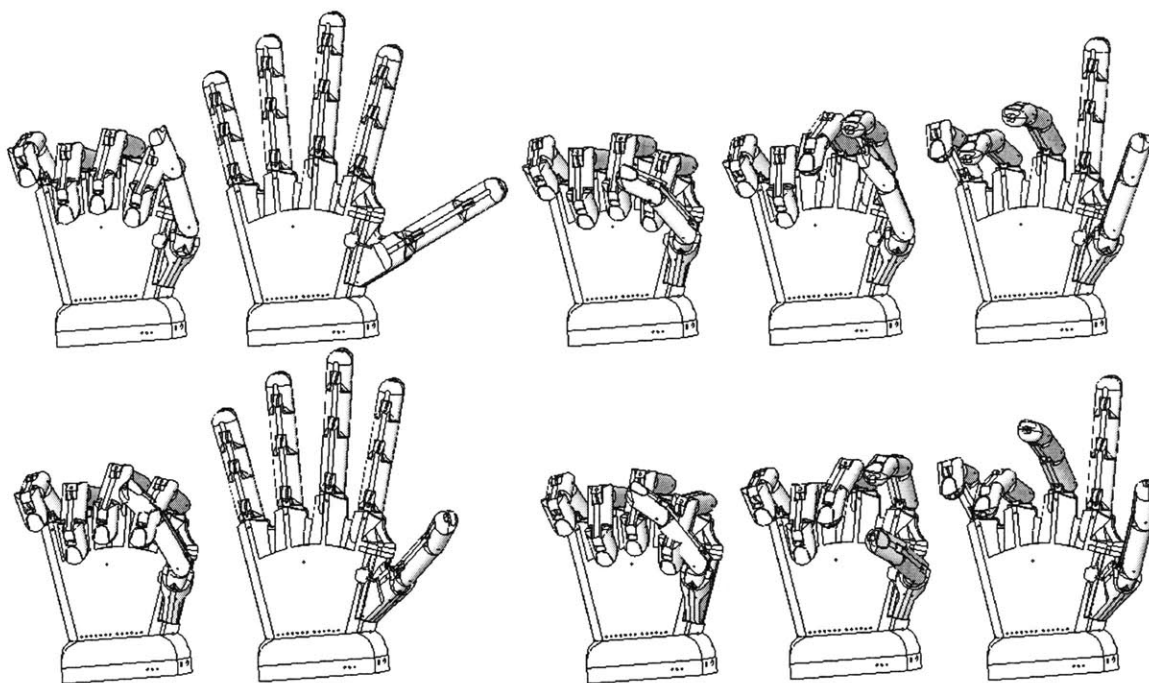


Figure 6.5: Postures reconstructed from 2 eigenpostures. The top row shows the original posture, while the bottom shows the approximation. From left to right: fan, flat, jacket, pen, remote control.

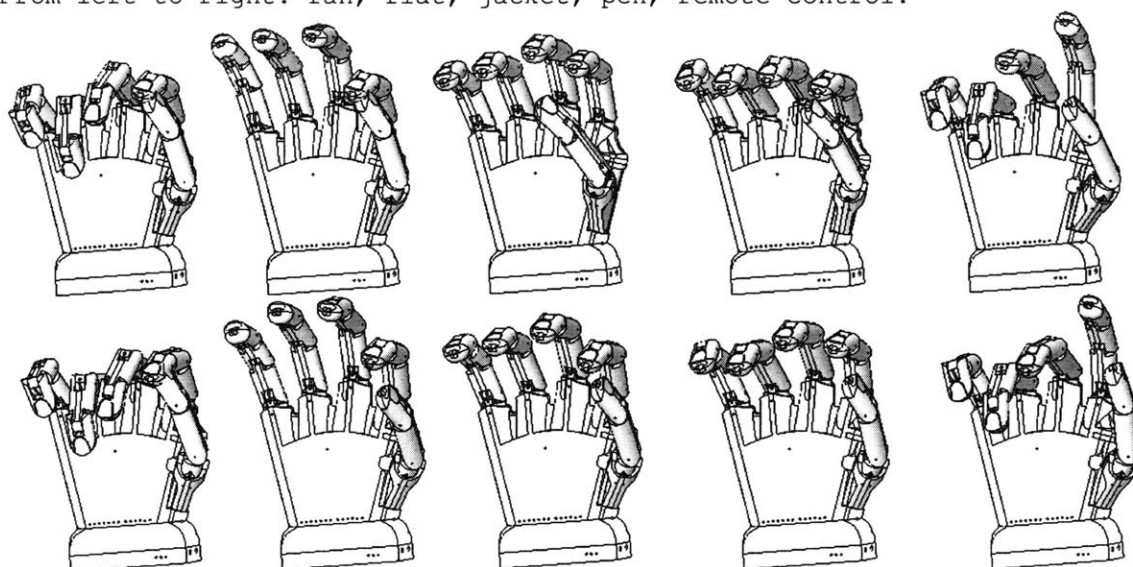


Figure 6.6: Postures reconstructed from 2 eigenpostures. The top row shows the original posture, while the bottom shows the approximation. From left to right: shogi piece, toothpick, tray, umbrella, wine glass.

From the figures, we see a somewhat different story. Although Figure 6.3 shows that some joints have a particularly large average error, in Figures 6.4 - 6.6 almost every posture remains functional. The dominating error in Figure 6.3 is in the rotation of the thumb. This result is expected, because the rotation allows the opposable nature of the

thumb which is so vital to complex hand tasks. The thumb is the most independent of the fingers, and since synergies rely on coordination between all joints, we expect a large error in our approximation of thumb position. Fortunately, in the actual reconstructed postures, the position of the thumb rarely seems to severely hinder functionality. In addition, note that in some of the reconstructed postures – the brush grip, for example – there is some interference between joints. Since two joints cannot occupy the same space in the real world, the error in these cases would be slightly lower. In fact, this is the case with several other postures that don't seem to show interference, because each posture is actually a grip. The joints cannot contract further once they contact a solid object, so any large positive errors in joint displacement would likely be smaller in an actual grip. We can even use this contact to our advantage, by introducing some compliance into the tendons. This design modification will be explored further in chapter 8.

6.3. Maximum Reconstruction Errors

For a better understanding of some of the maximum errors, see Table 6.2. This table lists the maximum error of each joint angle, along with the posture which includes that error. For example, we see that the maximum thumb rotation error of 38 degrees occurs in the “flat” posture, and looking back at Figure 6.5, we can see the effect of this error. It is necessary to adopt a qualitative perspective to these errors, since only our personal judgment may dictate whether such an error is acceptable. It is unlikely that the flat posture would be necessary for any precision tasks, so this large error may not be critical. In addition, many of the maximum errors occur in the same postures (doorknob and cup grips, for example). If such tasks do not require high precision, then the overall error must be evaluated with this in mind.

Angle	Max Error (degrees)	Posture with Max Error
IDIP	10.34	Doorknob
IMIP	16.86	Pen
IPIP	14.61	Umbrella
MDIP	12.92	Doorknob
MMIP	14.32	Umbrella

Angle	Max Error (degrees)	Posture with Max Error
MPIP	14.13	Doorknob
RDIP	15.14	Doorknob
RMIP	15.30	Cup
RPIP	13.96	Cup
PDIP	16.26	Doorknob
PMIP	16.61	Cup
PPIP	9.52	Cup
TDIP	18.97	Cell phone
TPIP	17.88	Wine glass
TAB	26.18	Fan
IAB	8.17	Bottle
TROT	38.15	Flat

Table 6.2: Maximum joint angle errors and associated postures.

Overall, the posture reconstruction is promising, showing that a variety of different grips and postures can be constructed using only two basis eigenpostures. However, such a result is only useful if it can be realized in mechanical form. We must combine the mechanism concepts from chapter 4 with the data collection and analysis from chapters 5 and 6 to further emphasize the power of the eigenposture-based design. The next chapters detail our efforts to develop a prototype hand using only two actuators to implement the eigenpostures, along with observations and recommendations on the overall design concept.

6.4. References

- [1] E.J.Weiss, M.Flanders. "Muscular and postural synergies of the human hand," Journal of Neurophysiology. Vol. 92: 523-535, 2004 Feb. 18

7. PROTOTYPE DESIGN

7.1. Initial Design Observations

Figure 7.1 shows the eigenposture mechanism from chapter 4, which forms the heart of our design. The figure shows a basic schematic form, but although several configurations are possible, this diagram reflects some careful consideration on the preferred embodiment of the mechanism. A few important points should be noted. The location of the two input shafts is highly flexible. The central guide rods ensure that the tendons always pull directly downward on the output pulleys, so the input shafts can be relocated freely. Without the guides, the tendons would contact the central pulleys at many different angles, so that the output would not be a pure sum of the 2 inputs.

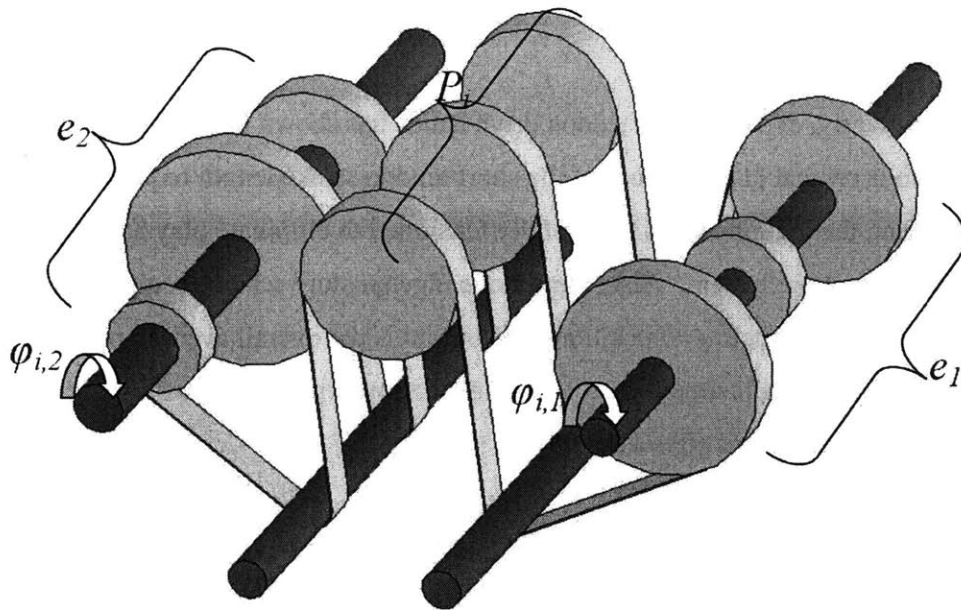


Figure 7.1: The eigenposture mechanism is shown again here for convenience.

The central pulleys themselves likely present the greatest difficulty. Since they must translate in the vertical direction without interference, careful attention must be paid in their design. Sliding, or *prismatic* joints, are some of the most difficult interfaces in robot design, and we will see several incarnations of these components in the coming pages. In addition, the diagram does not indicate how the primary finger tendons will connect to the output pulleys. Here, again, several configurations are possible, as will be explored.

Mechanical design is primarily an iterative process, and improvements are almost always possible. This chapter is an attempt to summarize the thinking and design process that went into the construction of one configuration of an eigenposture-based robot hand.

7.2. Design for Manufacturing, Design for Assembly

Even with our underactuated design requiring only 10 driving tendons, the mechanism will be considerably intricate. The diameter of the pulleys on the eigenposture shafts must be very precise in order to take advantage of the mathematical concepts at work, so smaller pulleys will require tighter manufacturing tolerances. On the other hand, a design goal is to keep the overall size of the device small, so the pulleys must not be too large either. In our earlier analysis, we chose a minimum pulley diameter of 6.35mm, (leading to a maximum size of about 36mm), to account for some of these difficulties.

More importantly, experience in tendon drive hands has shown that a final adjustment mechanism is critical [1]. Because of the short tendon stroke length required for full joint displacement, the tendons must be carefully tensioned to eliminate play and unwanted compliance from the system. Also, our unique eigenposture scheme requires that, when both actuators are at their $\varphi=0$ positions, the output is the overall average posture, \bar{z} . To account for this, the adjustment mechanism must allow a precise repositioning of the output pulleys during and after assembly.

The eigenposture shafts can be constructed 2 ways, both of which were considered in various prototypes. One method is to assemble them from multiple components, stacking multiple discrete pulleys on a central input shaft. Alternatively, the entire shaft can be constructed from a single solid rod – most likely by turning down the multiple pulley diameters in a CNC machining operation. Both designs present advantages and difficulties – the discrete method, for example, leads to an increased part count and introduces an additional issue of how to best fasten the pulleys to the shaft. However, it also permits individual pulleys to be replaced on demand in case of component failure or design changes, and can allow the pulleys to be rotated with respect to one another as an

adjustment technique during assembly. Constructing the shafts from a single piece requires a complex manufacturing method and makes changing eigenpostures difficult, but greatly reduces the total number of components and could increase the torsional stiffness of the mechanism.

Almost all of the above considerations relate to a central idea: we must design for assembly, and design for manufacturing. The hand itself is designed to be printed from ABS plastic on a fused deposition modeling (FDM) rapid prototyping machine, and is adapted from a design in [1]. Other components must be designed with the manufacturing and assembly constraints of a graduate student research lab in mind. We now examine the prototype evolution and associated observations.

7.3. First Prototype

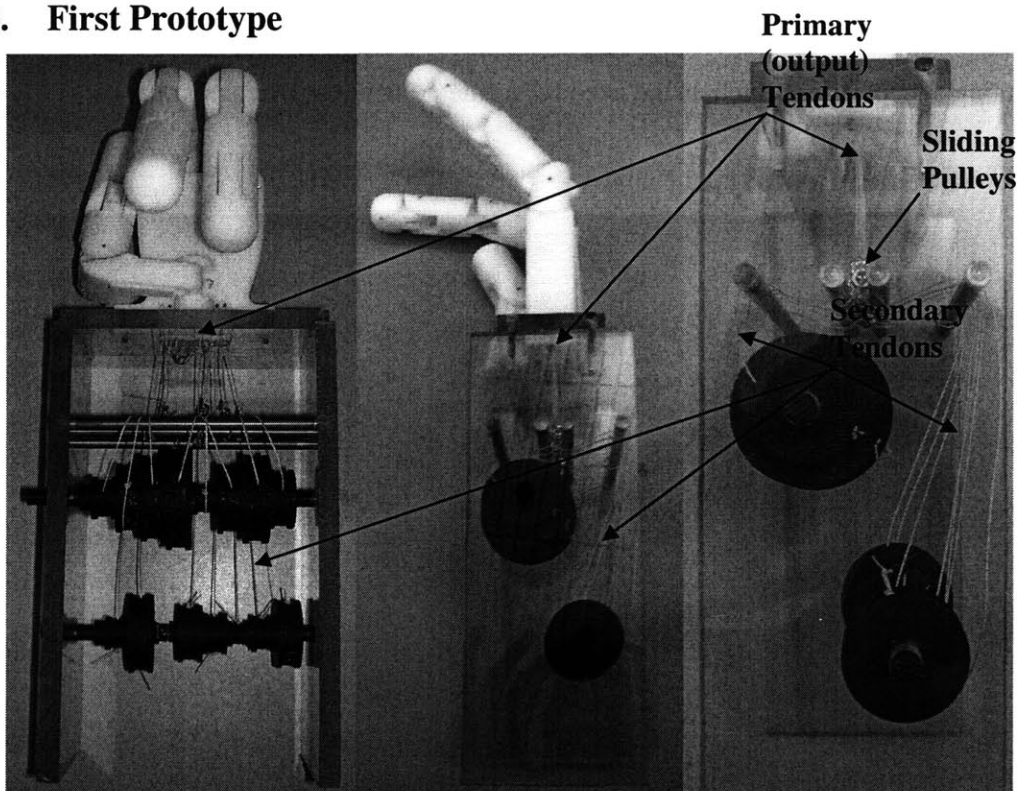


Figure 7.2: First prototype assembly.

Figure 7.2 shows the first prototype, a mockup created mainly for concept demonstration purposes, and to study some of the physical relationships involved. The design uses only a 3-fingered hand with the thumb pivoted about the palm base, as seen in the far left of

Figure 7.2. In this design, the eigenposture shafts, which were manufactured on the FDM machine, have been located below the hand, rather than farther off to the sides, as in Figure 7.1. However, this location requires two additional guide rods to keep the tendons from interfering with each other, which increases friction in the mechanism.

Difficult to see in Figure 7.2 are the sliding pulleys, located between the central guide rods. In this design, these output pulleys are made from simple metal rings through which the secondary tendons are threaded. The primary tendons are tied to the rings and then connect to the finger joints. Since there are no guides to constrain the motion of the sliding pulleys, their behavior is somewhat unpredictable.

This design clarified the geometric considerations, and allowed for a preliminary measurement of the frictional forces at work. In addition, the assembly involved made certain concepts clear, such as the need for a fine adjustment mechanism.

7.4. Second Prototype

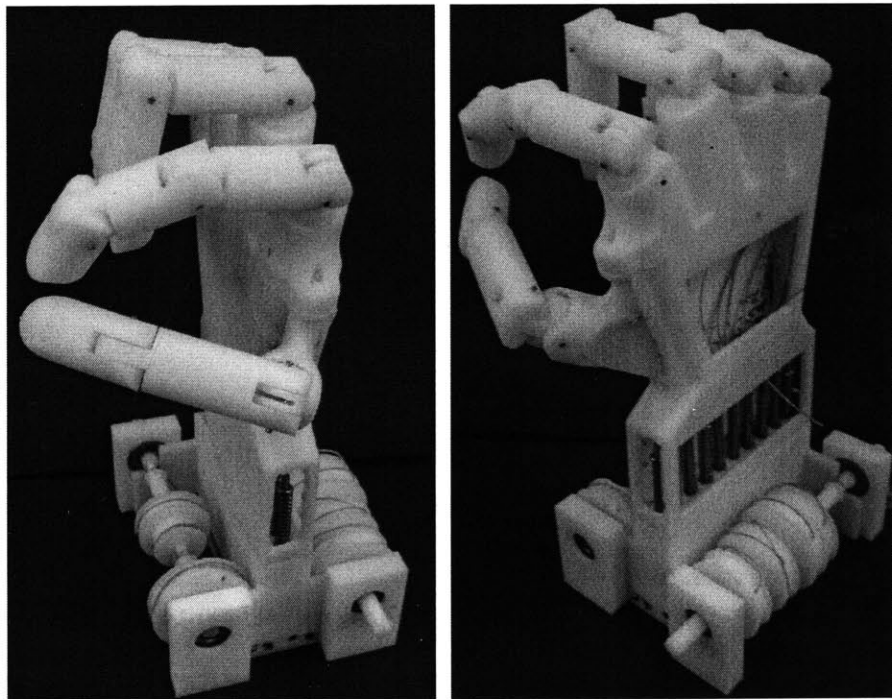


Figure 7.3: Second prototype assembly.

The second prototype, shown in Figure 7.3, is more true to the design concept in Figure 7.1, with the eigenposture shafts located to the sides. The idea behind this placement is that with more advanced manufacturing techniques, the shafts could be miniaturized and located inside the wrist of a complete robot. Since most of the mechanism is hidden inside the base, a cross-sectional view, showing tendon routing and other details, appears in Figure 7.4.

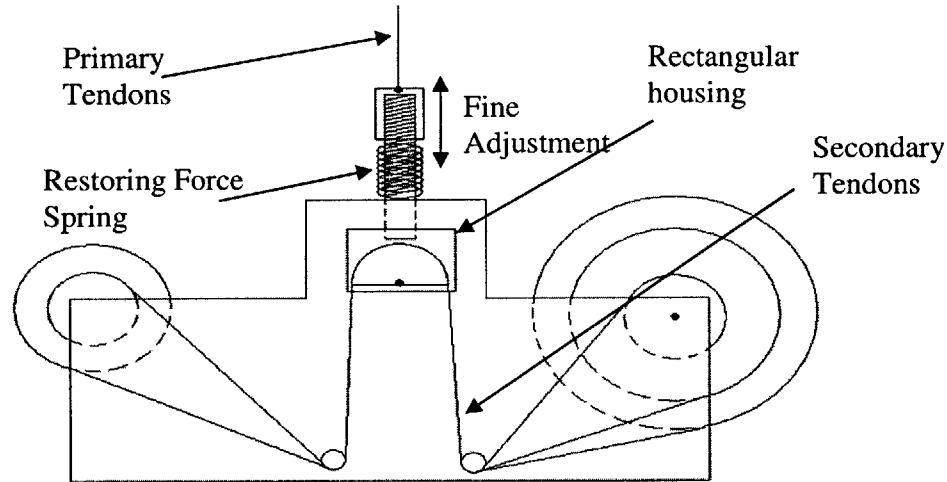


Figure 7.4: Cross-sectional view of prototype 2 configuration.

In this design, the sliding joint consists of half of a low-friction plastic pulley inside a rectangular housing. Note that the pulley does not turn, but instead is in sliding contact with the secondary tendons. A rigid rod is threaded into the rectangular housing, with a long nut on the opposite end. The primary tendons connect to this nut, so that by screwing the nut farther onto the rod, fine adjustment of the primary tendons can be accomplished. The movement of the housing, rod, and pulley is constrained to the vertical direction because the entire assembly sits in a channel in the base, as shown in Figure 7.5. In addition, the spring on the rod ensures that the secondary tendons remain in tension.

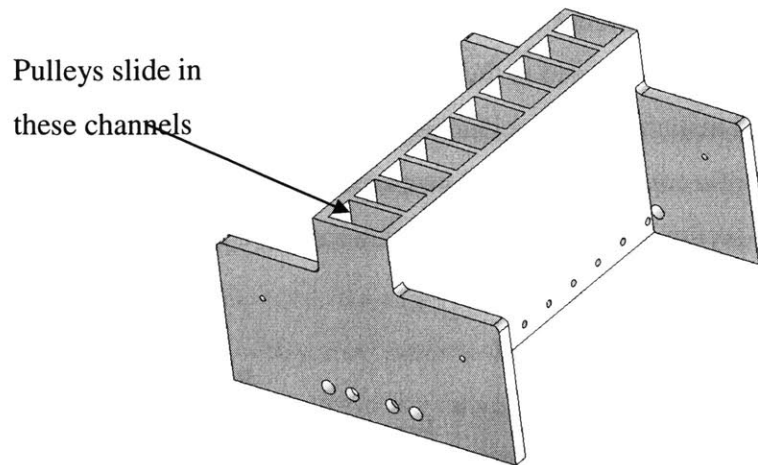


Figure 7.5: 3-D solid model of prototype 2 base. The sliding pulleys, with their integrated housing and guide rods, slide in the 10 channels shown.

This prototype is the start of a complete realization of the eigenposture drive, but still has its own problems. The sliding pulleys, as expected, presented a great difficulty. The rectangular housing tended to jam in the guide channels, and the tensioning springs and adjustment nuts interfered with each other. In addition, the adjustment mechanism did not work as well as expected. The rotation of the nut caused the primary tendons to become twisted, and a greater range of adjustment was also necessary. The smooth required motion of the sliding pulleys, and a more advanced adjustment mechanism, became the focus of the third and final prototype.

7.5. Final Prototype

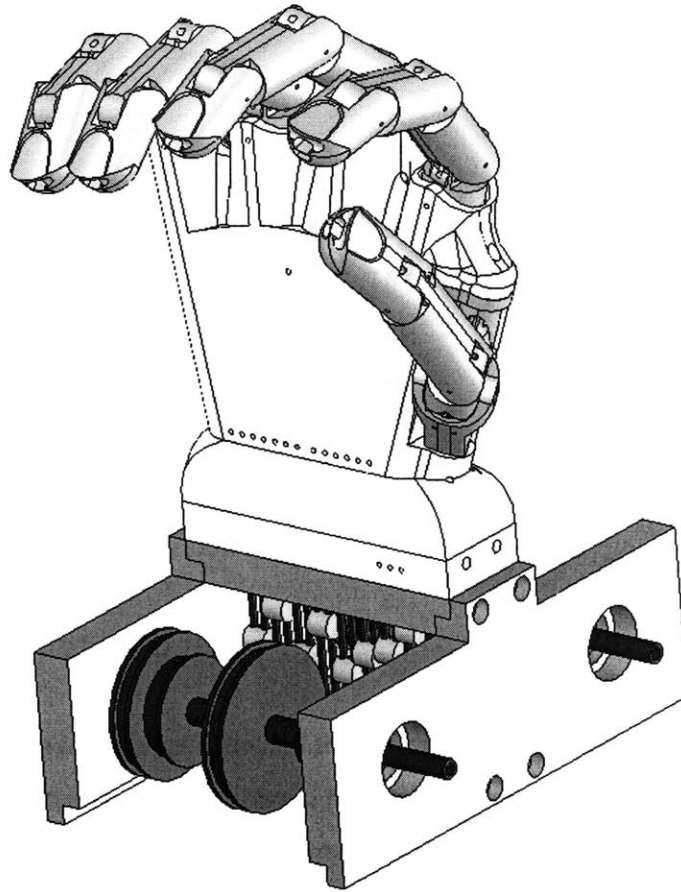


Figure 7.6: Final Prototype Design.

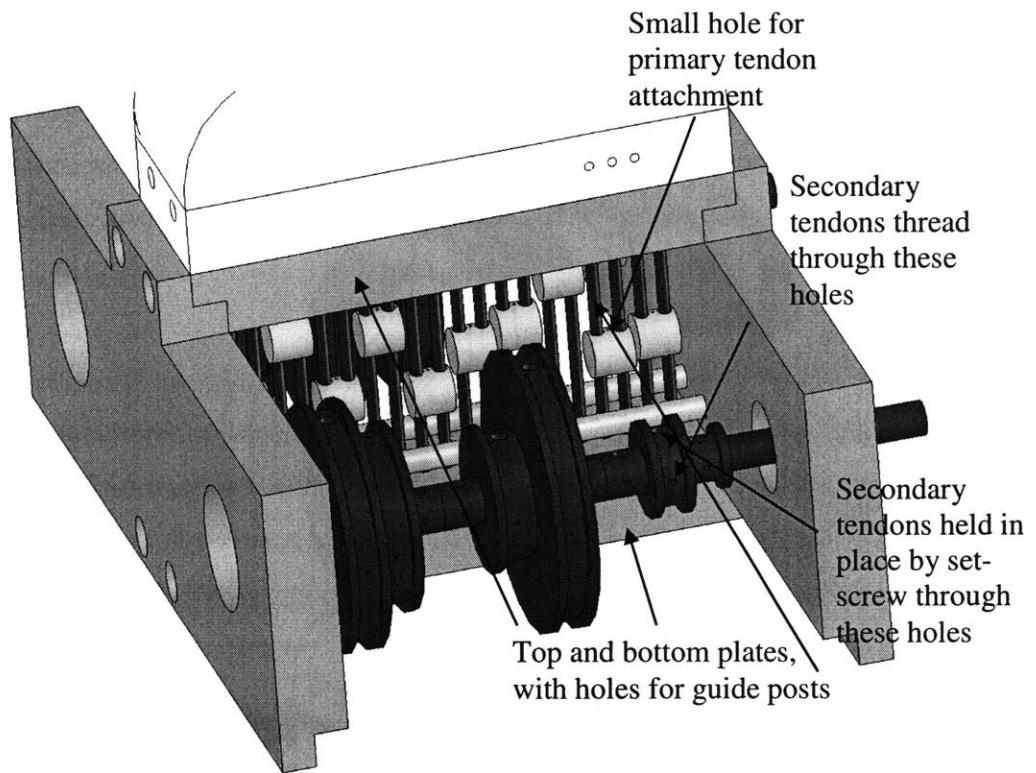


Figure 7.7: Close-up of final prototype base, showing sliding pulleys and adjustment mechanism.

Mechanical design, as mentioned at the start of this chapter, is an iterative process, and the third prototype reflects the lessons learned in earlier trials. The movement of the sliding pulleys in this configuration is much smoother, as each consists of a short length of aluminum rod sliding on two precision-ground steel posts. The posts are held in vertical alignment by holes in the top and bottom plates, and the assembly features (such as the notches shown in Figure 7.7) on the main structure ensure that all the components are aligned properly. Provided that the diameter of the sliding pulleys is sufficiently large compared to the diameter of the guide posts, jamming is kept to a minimum, and the friction can be reduced by lubrication applied to the guide posts (unlike in the second prototype, where the contact was between two plastic surfaces). The primary tendons are attached through a central hole in the sliding pulleys.

This design also incorporates a simple but effective adjustment mechanism. The secondary tendons are fed through small holes in the eigenposture shafts. Perpendicular to these holes is a larger hole, threaded for a #4-40 set screw. When this screw is loosened, the tendons can be tightened and pulled through the holes until the sliding pulleys reach the required positions. Tightening the screw locks the tendon in place.

Bench-level experiments with this scheme showed that extreme force was necessary to cause the tendon to slip.

In the manufactured version of this prototype, an important design change was included. The eigenposture shafts, unlike in previous designs, are made of discrete components. A central aluminum rod with a diameter of 6.35mm forms the core. Each pulley was created on the 3D printer, and attaches to the rod via a keyway. In some cases – the smallest values of $d_{k,j}$ – the rod itself serves as the pulley. This design allows for simple changing of individual pulleys as necessary, and eases manufacturing requirements. The completed prototype assembly is shown in Figures 7.8 - 7.10.

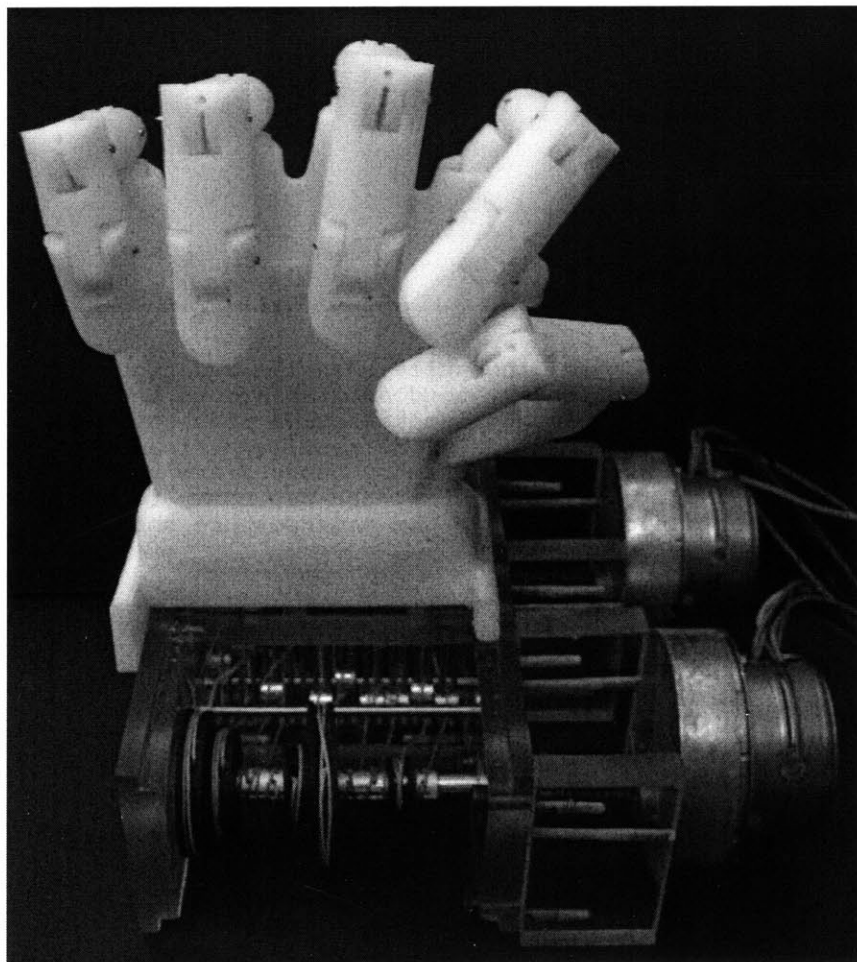


Figure 7.8: Front view of final prototype. 2 stepper motors, on the right, control the eigenposture shafts. The motors are attached to the shafts via a clamp-type flexible coupling.

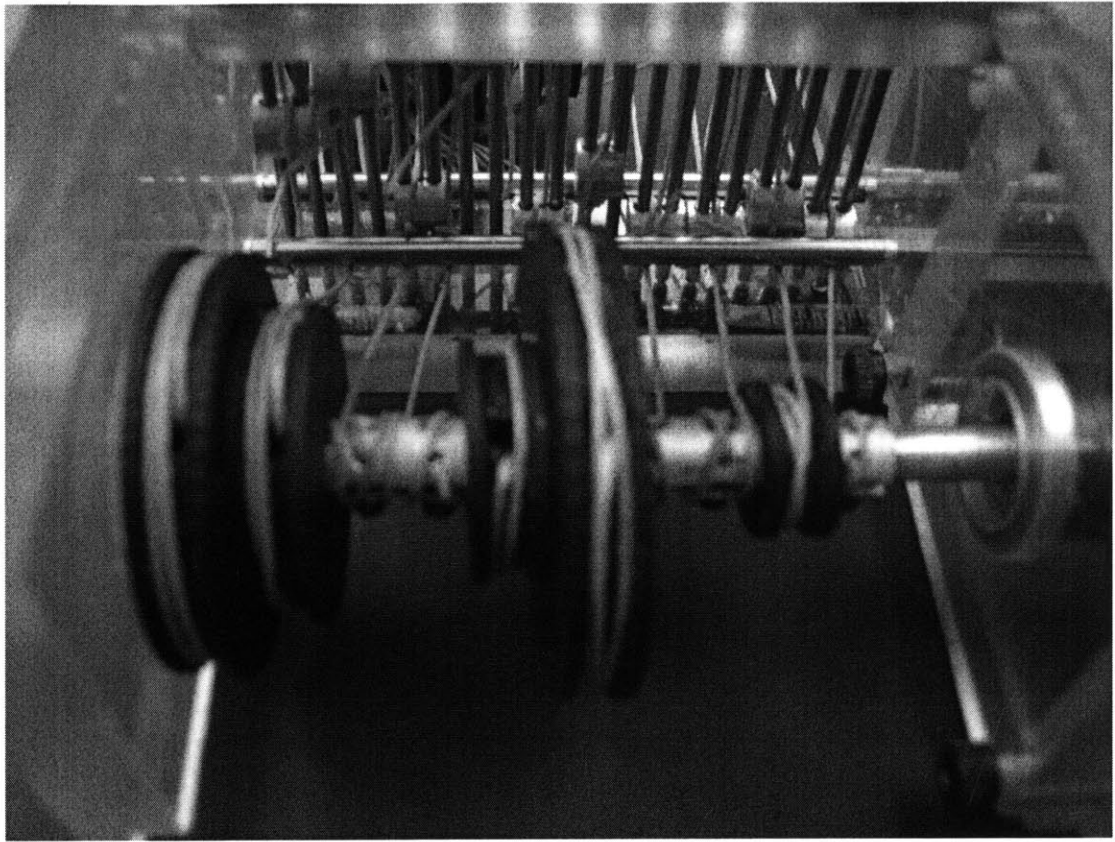


Figure 7.9: Close-up of sliding pulley details and tendon routing. In the foreground, the secondary tendons wrap around the eigenposture shafts, then travel under the horizontal guide rods, over the sliding pulleys, under another guide rod in the rear, and finally to the second eigenposture shaft. Some of the primary tendons are visible between the vertical guide posts. Note again that the secondary tendons can be wrapped in either direction around the eigenposture shafts.

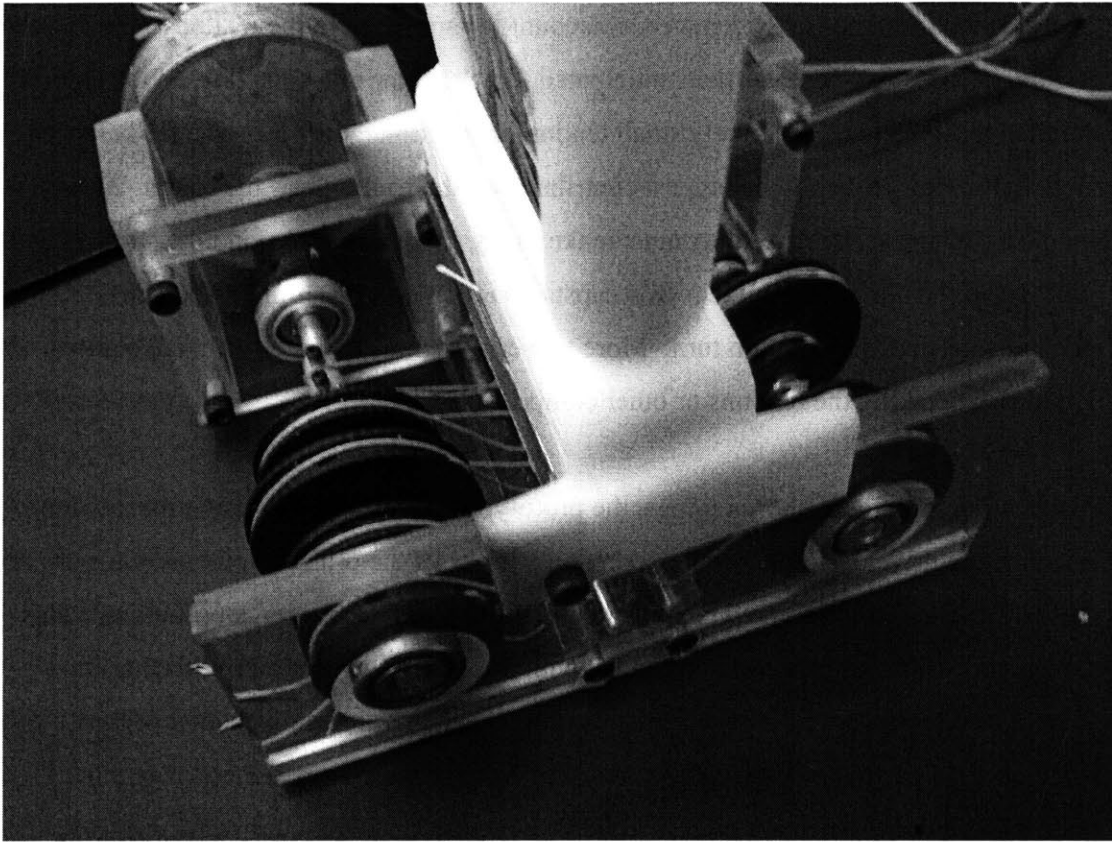


Figure 7.10: Another view of the prototype, showing both eigenposture shafts simultaneously. Also visible at the bottom are the bearings which support the shafts. In the top center, we catch a glimpse of the tendons through an access hole in the back of the hand.

7.6. Prototype Performance

Unfortunately, the final prototype did not meet performance expectations. Despite the improved adjustment mechanism, there was too much play in the assembly, so that the primary tendons could not be properly set to their zero-offset average posture. In other tendon-drive robot hands, such as in [1], the tendons are often connected with the fingers in the fully extended position. This allows maximum tension to be applied to the tendons, since the geometry of the fingers prevents further movement of the joints. In our design, however, the tension must be created with the fingers in the average posture position, which proves more difficult. In future versions, a jig should be created to clamp the fingers firmly into the average position, while assembly occurs.

In addition, the efforts to make the mechanism as small as possible were overenthusiastic. Since the adjustment procedure was more difficult than anticipated, the sliding pulleys

required more vertical travel distance to account for any slack in the tendons. Finally, the frictional effects were greater than anticipated. Each motor must displace all the tendons simultaneously, and the relatively small tendon connection radii r causes a fairly large frictional force. Probably the biggest contributor to the friction was the routing of the secondary tendons. Because they must make many sharp turns – around the sliding pulleys and the horizontal guide rods – capstan friction dominates and makes the eigenposture shafts difficult to turn. More powerful motors could be utilized, but this would risk breaking the tendons or other components.

However, despite these shortcomings, the prototype still confirmed many aspects of the eigenposture drive design. By connecting only a few of the primary tendons, we can show distinct differential movement, which is the key to inter-finger coordination. This movement is shown in the video provided with the electronic version of this thesis. Although we could not confirm the reconstructed postures shown in chapter 6, the kinematics and actuation concept are sound, and the setbacks experienced here have clear-cut solutions which are presented in the conclusions to this thesis.

7.7. References

- [1] K.J.Cho, J. Rosemarin, H.H.Asada. “Design of vast DOP artificial muscle actuators with a cellular array structure and its application to a five-fingered robotic hand,” in Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp.2214-19. May 15-19, 2006.

8. IMPROVEMENTS TO THE EIGENPOSTURE DRIVE

8.1. Introduction

As the prototype construction demonstrates, several issues need to be resolved to fully prove the concept of the eigenposture drive. However, the theory and simulations in chapter 6 establish the extremely promising mathematical ideas. In addition, several other theoretical concepts have been explored which greatly increase the scope of eigenposture-based inter-finger coordination. Although these ideas have not been implemented in hardware, they are presented here as possible concepts for future study.

8.2. Multiple Eigenpostures

The inspiration for much of the work here was the knowledge that up to 80% or more of the total information in everyday tasks was accounted for by only two synergies. With this in mind, we set out to develop an actuation scheme based on two eigenpostures. However, looking at Figure 5.4, for example, we see that using four eigenpostures leads to over 90% accuracy. With some modification, the mechanisms explained in this thesis can be expanded beyond 2 eigenpostures. One way to accomplish this is to simply stack the mechanisms on top of each other. In our analysis, we connected the summing pulleys (from Figure 4.3) directly to the primary hand tendons. However, one could imagine 2 sets of eigenposture mechanisms, with their outputs attached to a third set of summing pulleys. Continuing in this manner, we can construct a mechanism based on any number of 2^n eigenpostures. Since our goal was a compact mechanism that explored the power of the minimum number of synergies, we did not explore this option, but if 4 actuators could be arranged in a suitably small configuration, far more precise postures could be constructed. In 8.5 we explore a completely different way of using multiple eigenpostures.

8.3. Compliant Tendons

Some simple grippers, such as in [1], combine underactuated design with compliant tendons to produce effective grasps. The disadvantage of this approach is that the resulting postures are not always anthropomorphic; a 1- or 2- DOF compliant gripper would never grasp a pen in a tripod pinch, for example. By combining the eigenposture

scheme with compliant grasping, we can reduce error in grip postures while maintaining the human-like appearance. This idea was mentioned briefly in chapter 6. For example, consider the toothpick grip from that chapter. This is a grip where contact between the thumb, toothpick, and index finger are critical. In our initial posture definition, we place the fingers in this configuration. However, if the primary tendons are compliant, then a better posture definition could include some interference between the thumb and index finger. In other words, we make the thumb and index finger try to occupy the same space. In the actual mechanism, once the joints come into contact, the primary tendons stretch, which provides the important gripping force. A larger initial interference results in a larger gripping force. Since the initial postures are defined independently of one another, we can include varying amounts of this compliant effect as desired. Postures needing strong gripping force would contain maximal interference, while gestures and other non-grips would contain none. By iterating the process and adjusting parameters as necessary, there is potential to greatly reduce overall error while introducing a definable gripping force.

8.4. Nonlinear Eigenposture Pulleys

This is a possibility which was considered from the beginning of this research. We chose to use principal components analysis to determine the shape of the eigenposture shafts, because it is a common method already in use for calculating postural synergies.

Because of the linear nature of the equations, PCA leads naturally to circular pulleys on the eigenposture shafts. However, this is by no means a requirement, and experimenting with different pulley forms could significantly increase the power of the concept. In general, the vector output of shaft k , for posture i , is a function of the input angle, $\phi_{i,k}$:

$$[y_{i,k,1} \cdots y_{i,k,j} \cdots y_{i,k,n}] = f_k(\phi_{i,k}; C_{k,1}, \cdots, C_{k,s}, \cdots, C_{k,M}) \quad (8.1)$$

The total vector output, which attaches to the primary hand tendons, is given by:

$$[z_{i,1} \cdots z_{i,j} \cdots z_{i,n}] = \frac{1}{2} \{f_1(\phi_{i,1}; C_{1,1}, \cdots, C_{M,1}) + f_2(\phi_{i,2}; C_{2,1}, \cdots, C_{M,2})\} \quad (8.2)$$

In these equations, the function f_k is parameterized by the M constants $C_{k,1} \dots C_{k,M}$. In all our previous analysis, these constants are simply the diameters $d_{k,1} \dots d_{k,n}$, and the function is just $\phi_{i,k}$ multiplied by the vector of constants. Consider, however, a minor design

change. If we modified the pulleys to be *elliptical* instead of round, then each pulley would have *three* parameters: the major radius, minor radius, and phase angle, for a total of $3n$ parameters for each shaft. In fact, more parameters are possible for an elliptical shape if it is not centered on the shaft. More parameters for us to tweak should mean greater accuracy in the results, but it comes at the cost of a complicated nonlinear optimization equation:

$$\min_{\phi_{i,k}, C_{k,s}} \left\{ \sum_{i=1}^N \left\| \sum_{k=1}^2 f_k(\phi_{i,k}; C_{k,1} \dots C_{k,M}) \right\| - P_i \right\}^2 \quad (8.3)$$

That is, given the N postures $P_1 \dots P_N$, we wish to minimize the sum of the squared norms of the reconstruction errors. In this equation, we have to calculate $2i$ input angles $\phi_{i,k}$, as well as $2M$ parameters $C_{k,s}$. PCA gives us an optimal result for a simple linear function, but depending on the exact form of f_k , an optimal solution may be difficult here.

However, this is an intriguing approach that requires further study if eigenposture actuation is to be fully explored. Some practical design factors must be considered. Arbitrary pulley shapes, while possible mathematically, may be difficult to realize in mechanical form. A shape with a concave section, for example, may not necessarily produce the desired output, because the tendon would not properly wrap around the shaft.

8.5. Reconfigurable Posture Groups Using Clustering Analysis

This idea arose late in the research stages, but it holds the potential for powerful and exciting results, so it is the last improvement described here. By using everyday tasks as the initial posture set, we have tuned the eigenpostures to be as broad and general as possible, able to reconstruct a variety of common tasks. However, as noted in 2.2, smaller subsets of tasks can be described with their own synergies as well. Figure 8.1 shows this concept graphically. In our initial analysis, we used the entire set labeled “Desired Tasks” as our posture matrix. It is possible, though, that a smaller subset of these tasks, as indicated by the separating line, is fine-tuned for a different set of eigenpostures. One can imagine, for example, that the set of postures for a dishwashing robot would utilize different eigenpostures than a clothes-washing robot, although either set of tasks might be desirable in a home robotics application. If we had a method for

identifying these task subsets and then *selectively reconfiguring* the eigenposture drive shafts, a huge increase in accuracy could be obtained.

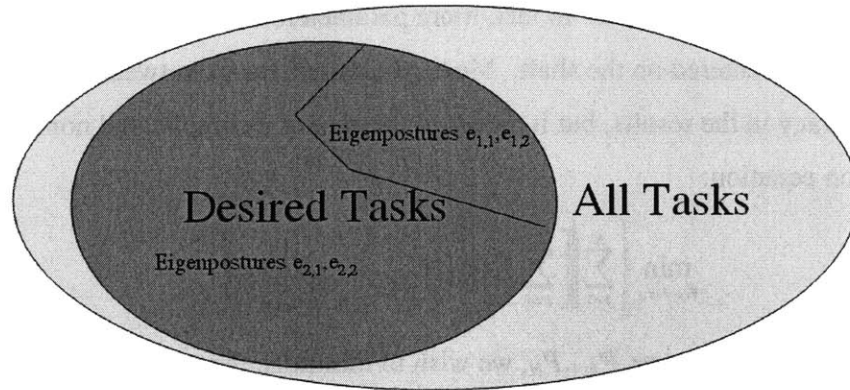


Figure 8.1: Clustering of eigenpostures for specific task subsets.

Identifying the task subsets is straightforward. We can use a clustering algorithm to identify unique groups. We began with our initial 15-posture set from chapter 5 and applied a k-means clustering algorithm, given in detail in Appendix B. A recent study [2] found that k-means clustering and PCA are closely related, as we will see ourselves shortly. In this algorithm, random initial points are picked in the n -DOF space as group centers. We began with only 2 distinct groups. Once the random centers are chosen, each posture is associated with a group by picking the closest center point. Next, the geometric center of each group of postures is calculated, all the postures are re-associated with the new centers, and the process is repeated. The algorithm quickly converges so that postures no longer switch between groups. Once the groups are determined, we apply the standard principal components analysis to each group to find its eigenpostures.

The results of this algorithm are very promising. The two groups are shown in Figure 8.2. This chart requires some explanation. Since the postures exist in 17 -space, they must be mapped to 2 -space or 3 -space for visualization. In this plot, we first calculated the eigenpostures of the entire posture matrix – that is, all 15 postures, with no grouping. Each posture is then mapped to this 2-dimensional space using the standard PCA analysis. We then conduct the grouping algorithm to see which postures are most closely related, and mark them in the chart with symbols to identify the groups. The circles around each group are simply the minimum size required to touch each posture in that

group. The center of each group is shown with a separate symbol, but it does not appear to be the geometric center in the figure. This is because each posture really exists in *17-space*, as noted previously. The centers really are the geometric centers in the full posture space, but appear somewhat skewed in the *2-space* mapping.

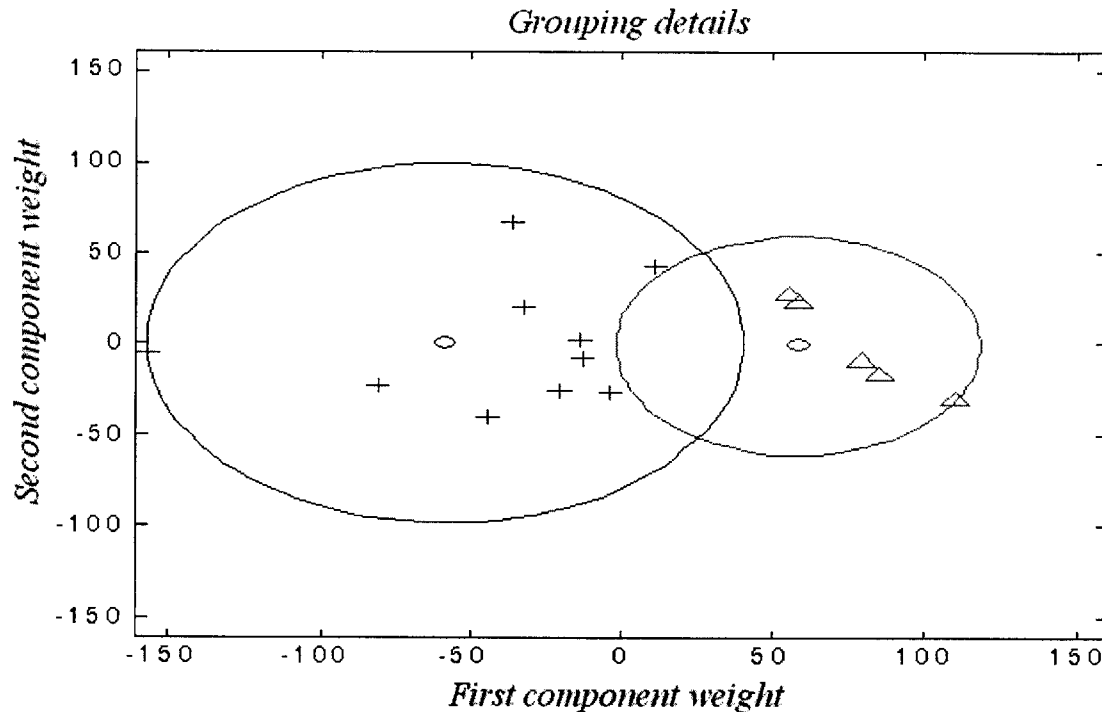


Figure 8.2: Grouping results from k-means clustering. The small circles mark the group centers, with each group identified by a unique symbol.

Notice that the clustering algorithm has identified groups of 2 distinct sizes, perhaps somewhat surprisingly. In fact, the smaller of these two groups can be almost *exactly* described with only 4 eigenpostures. Somehow, the clustering algorithm found an extremely useful result for our PCA procedure, even though PCA itself was not involved in the calculation. Figure 8.3 shows the real power of the procedure. In this figure, the reconstruction error for each posture is calculated, using the eigenpostures for its unique group. In other words, we use 2 sets of 2 eigenpostures each. This decreases the average error in every case, and significantly decreases our largest previous error, with excellent results. The original error from Figure 6.3 is copied here for convenience.

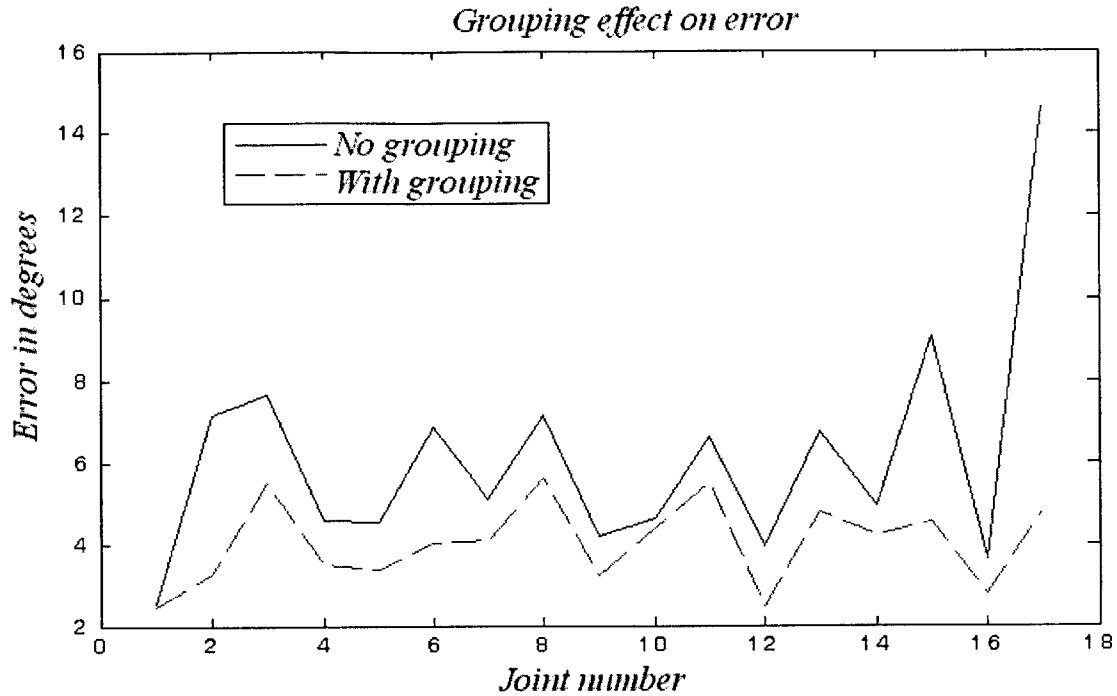


Figure 8.3: Effect of grouping on average posture reconstruction error.

Switching between eigenposture sets obviously leads to a significant increase in accuracy, but how do we implement such a scheme mechanically? One way would be to use the nested mechanisms already discussed, but this has no advantage over always using four eigenpostures. Instead, we can use the mechanism shown in Figure 8.4. The figure shows a side view of 2 shafts, along with several spur gears. The gears on the upper shaft are fixed rigidly to the shaft, so that they all rotate simultaneously. The lower shaft, on the other hand, consists of distinct units, each of which is *free to rotate* on the shaft. In the center of the unit is a pulley, which takes the place of the eigenposture pulley shafts – the secondary tendons wrap around these pulleys and then travel to the summing mechanism. The pulley is rigidly attached to 2 spur gears, one on each side. As the upper shaft turns, the pulleys turn at different rates on the lower shaft. The ratio of movement is controlled by the size of the pulleys as well as the gear ratio. When the upper shaft is shifted left or right, a new set of gears meshes with the pulleys, changing all the ratios simultaneously. In this manner, a simple left-right shift of the upper shaft allows us to reconfigure to an entirely new eigenposture. Only one additional actuator is required for this scheme, and in some applications is not even necessary. A robot with

two hands, for example, can use each hand to “change gears” on the other, allowing us to keep to a minimum actuator count if desired. This mechanism, although appearing simple in the diagram below, holds great promise for harnessing the full power of eigenposture-driven hands.

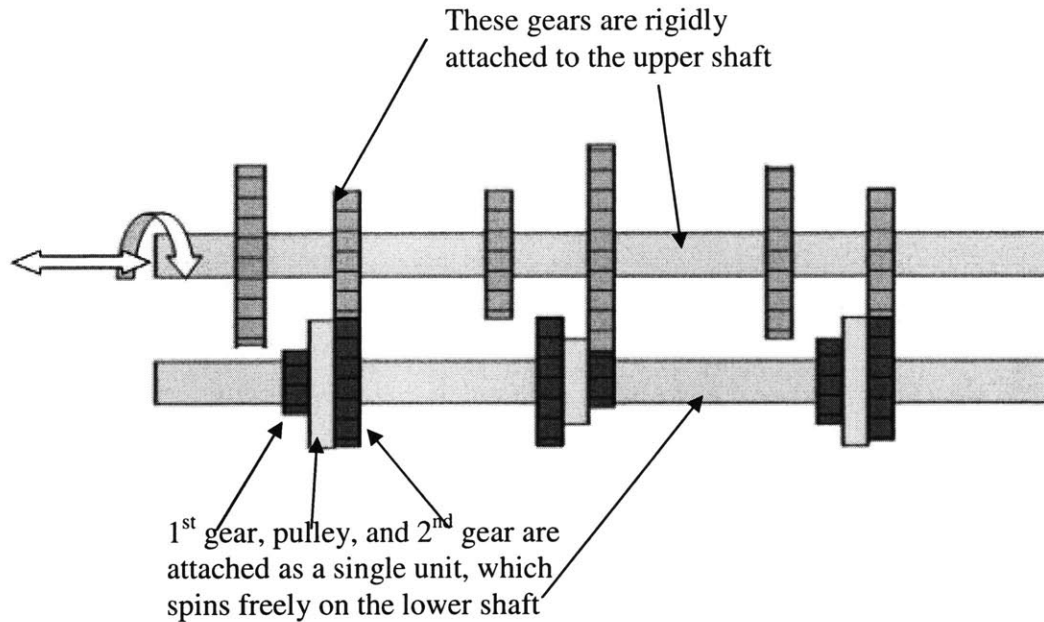


Figure 8.4: Mechanism for reconfigurable eigenpostures.

8.6. References

- [1] A.M.Dollar, R.D.Howe. “Towards Grasping in Unstructured Environments: Grasper Compliance and Configuration Optimization,” *Advanced Robotics*. Vol. 19(5): 523-543, 2005.
- [2] C.Ding, X.He. “K-means clustering via principal component analysis,” in *Proceedings of the 21st ACM international conference on Machine learning*. Vol 69: 29, 2004.

9. CONCLUSIONS

Muscular and postural synergies of the human hand are an important phenomenon in neuroscience and computer science. They provide insight into neurological control schemes, and can ease processing requirements in gesture recognition. However, no robot hand to date has implemented synergies at the lowest level of design. If coupling between joints appears at all in robot hands, it is normally within single fingers. Synergies allow the unique concept of *inter-finger* coordination, allowing complex finger movements to occur using only a small number of actuators.

The results of our research are threefold. First, we have independently confirmed the significance of synergies in everyday hand tasks. As detailed in chapter 6, the variety of tasks chosen as our posture set can be described with 80% information retention using only two synergies. We name these synergies eigenpostures because of their use as basis postures and because of the methods of calculating them. Chapter 6 also shows the effect of recreating postures using 2 basis eigenpostures. We evaluated several measures of error and discussed the implications of each in home robotics applications.

Secondly, we have proposed a novel design for implementing eigenpostures mechanically. Although the actual design configuration is important, it is the concept of mechanically actuating the eigenpostures which we consider to be the greatest contribution of this work. Previous synergy studies focused on recognizing synergies in use, rather than using them to create hand shapes. By interpreting the equations of principal components analysis as a linear vector differential, a vast opportunity is created in robot hand design. We attempted to physically demonstrate this new concept, with moderate results. Although manufacturing and time constraints prevented our prototype from replicating simulated results, we were able to successfully show how our mechanism design could produce useful differential outputs.

With more time and improved assembly and manufacturing, we are confident that a fully-functioning eigenposture-driven hand can be developed. Our research also suggests

certain considerations to be accounted for in a next-generation prototype. The fine adjustment scheme is critical to success, and should be accessible at any time after construction. In addition, frictional forces can become significant, and care should be taken to minimize them. Tendon routing plays a big role here, so the secondary tendons in particular should change direction as little as possible. Sliding joints must be precisely constrained, with lubrication or other linear bearings to reduce friction and prevent jamming. In addition, a jig or other scheme is necessary to properly tighten tendons when setting the zero-offset average posture.

Thirdly, we have suggested and partially explored many possible avenues for further research in eigenposture actuation. Several configurations using more than the 2 eigenpostures here are possible. Compliance can be selectively introduced into certain posture definitions, to define gripping forces and improve accuracy. We have also laid out the basic mathematical analysis for the more generalized nonlinear eigenposture actuation. A huge potential exists here for vast performance improvements, given an intelligent choice of eigenposture functions.

Finally, we have proposed a powerful design improvement using another closely-related mathematical technique, k-means clustering. Grouping of postures into distinct subsets has an important physical meaning, allowing us to distinguish between tasks with unique eigenpostures. Another novel mechanism proposal provides a new way to actuate the eigenpostures, using a combination of gear ratios and pulley ratios. With a simple linear movement, this mechanism allows us to reconfigure the eigenpostures at will, switching between the two groups as necessary. In simulations, the overall error was significantly reduced, with the most dramatic improvement occurring on the previously largest error.

Robot hand designers have recently started paying more attention to the concept of synergies, and the novel concepts presented here make a significant first step towards incorporating synergies into a functioning actuation scheme. The most important benefits of such a design are portability, compact size, lower power requirements, and decreased cost. Although an eigenposture drive cannot reproduce high-precision postures, it is a

logical choice for many applications, such as home robotics, where these advantages outweigh the moderate accuracy attained.

APPENDIX A – DERIVATION OF PRINCIPAL COMPONENTS ANALYSIS

One of the most concise methods for deriving principal components analysis uses the covariance method. PCA is essentially a coordinate transformation, so we begin with the problem of finding the $N \times N$ transformation matrix R such that:

$$Y = R^T X$$

where X is our initial data matrix, and our constraint that $\text{cov}(Y)$ is diagonal. The columns of R will be the principal components, and since these components must be orthogonal, $R^T = R^{-1}$. Now we have:

$$\begin{aligned}\text{cov}(Y) &= E[YY^T] \\ \text{cov}(Y) &= E[(R^T X)(R^T X)^T] \\ \text{cov}(Y) &= E[R^T XX^T R] \\ \text{cov}(Y) &= R^T E[XX^T] R \\ \text{cov}(Y) &= R^T \text{cov}(X) R\end{aligned}$$

Left-multiplying through by R :

$$\begin{aligned}R \text{cov}(Y) &= RR^T \text{cov}(X) R \\ R \text{cov}(Y) &= \text{cov}(X) R\end{aligned}$$

Now we re-write R in terms of the individual principal components:

$$[R_1 \cdots R_i \cdots R_N] \text{cov}(Y) = \text{cov}(X) [R_1 \cdots R_i \cdots R_N]$$

Since $\text{cov}(Y)$ is diagonal by definition, this is equivalent to:

$$[R_1 \lambda_1 \cdots R_i \lambda_i \cdots R_N \lambda_N] = \text{cov}(X) [R_1 \cdots R_i \cdots R_N]$$

where λ_i is the i^{th} diagonal entry of $\text{cov}(Y)$. This is equivalent to a system of uncoupled equations:

$$\text{cov}(X) R_i = \lambda_i R_i, i = 1 \cdots N$$

In other words, the principal components R_i are the eigenvectors of the covariance matrix of X . By finding these eigenvectors, we have the principal components.

APPENDIX B – SELECTED MATLAB FILES

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Eigenposture Calculation
%This program calculates eigenpostures from the initial posture matrix, stored in "data"
%It also calculates the Dmax/Dmin ratio and rescales the eigenpostures to obtain
% the desired minimum pulley size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
close;
clc;

%Load the posture matrix

data = [
0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0;
30     20     10     30     20     10     30     20     10     30     25     25
      30     20     0      0      0      0      0      0     45;
30     45     40     30     45     45     30     45     45     30     45     45
      15     30     0      10     0      0      0      0     90;
15     30     17     15     20     60     20     25     65     20     25     65
      30     20     0      10     0      0      0      0     60;
21     35     40     20     28     60     20     25     68     20     25     65
      10     20     0      0      0      0      0      0     90;
45     70     80     45     70     80     45     70     80     45     70     80
      10     40     30     0      0      0      0      0     60;
15     30     17     15     20     60     40     65     75     40     65     75
      10     40     20     10     0      0      0      0     52;
25     40     35     25     40     40     25     40     40     25     40     40
      15     30     0      10     0      0      0      0     90;
35     70     80     35     70     80     37     70     80     38     70     80
      5      30     45     0      0      0      0      0     60;
50     82     85     50     82     85     50     82     85     50     82     85
      35     40     0      0      0      0      0      0     75;
30     27     55     40     50     65     50     75     85     50     75     85
      15     30     20     0      0      0      0      0     65;
0      0      0      15     20     60     20     25     68     35     40     80
      0      0      28     10     0      0      0      0     55;
30     22     50     40     50     65     50     75     85     50     75     85
      15     35     20     10     0      0      0      0     60;
30     27     55     20     35     35     15     30     30     10     25     22
      15     30     20     0      0      0      0      0     65;

```

```

25    27    55    30    22    62    30    27    55    30    27    55
      15    30    5     0     0     0     0    75];

```

```

data = data*pi/180; %Convert to radians

```

```

%The data below contains geometric measurements from the kinematic analysis

```

```

X = [0.2504

```

```

0.2500

```

```

0.3246

```

```

0.2511

```

```

0.4600

```

```

0.2818

```

```

0.2500

```

```

0.2500

```

```

0.2500

```

```

0.2500

```

```

0.2500

```

```

0.2500

```

```

0.2583

```

```

0.3000

```

```

0.4988

```

```

0.5000

```

```

0.3042];

```

```

%Calculate the tendon-space posture matrix

```

```

for i = 1:15

```

```

    T(i,1) = X(13)*data(i,13) + X(14)*data(i,14);

```

```

    %Thumb Distal

```

```

    T(i,2) = X(15)*data(i,15);

```

```

    %Thumb Abbduction

```

```

    T(i,3) = X(16)*data(i,16);

```

```

    %Index Abbduction

```

```

    T(i,4) = X(17)*data(i,20);

```

```

    %Thumb Rotation

```

```

    T(i,5) = X(1)*data(i,1) + X(2)*data(i,2) + X(3)*data(i,3);

```

```

    %Index Distal

```

```

    T(i,6) = X(3)*data(i,3);

```

```

    %Index Proximal

```

```

    T(i,7) = X(4)*data(i,4) + X(5)*data(i,5) + X(6)*data(i,6);

```

```

    %Middle Distal

```

```

    T(i,8) = X(6)*data(i,6);

```

```

    %Middle Proximal

```

```

    T(i,9) = X(7)*data(i,7) + X(8)*data(i,8) + X(9)*data(i,9);

```

```

    %Ring All

```

```

    T(i,10) = X(10)*data(i,10) + X(11)*data(i,11) + X(12)*data(i,12);

```

```

    %Pinky All

```

```

end

```

```

%Run the PCA analysis and save the first to principal components

```

```

[c s l t] = princomp(T);

```

```

modes = c';

```

```

prinmodes = modes(1:2,:);

```

```

prinweights = s(:,1:2);

```

```

means = repmat(mean(T),15,1);

```

```

Eig1 = prinmodes(1,:);

```

```

Eig2 = prinmodes(2,:);
Eig1MaxRatio = max(abs(Eig1))/min(abs(Eig1));
Eig2MaxRatio = max(abs(Eig2))/min(abs(Eig2));

%Eliminate very small component values if the max/min ratio is too large
if(Eig1MaxRatio > 50)
    [ignore index] = min(abs(Eig1));
    Eig1(index) = 0;
    temp = Eig1;
    temp(index) = [];
    Eig1MaxRatio = max(abs(temp))/min(abs(temp));
end

%output the scaled eigenpostures
Eig1 = Eig1*.25/min(abs(temp))
Eig2 = Eig2*.25/min(abs(Eig2))

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Max Ratio Finder
%   This program is nearly identical to the last, with one exception. It is a function of X,
%   which consists of geometrical measurements that determine the tendon radii
%   connections, r. This function is called by fminsearch to find the optimum values of r
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
function maxRatio = findMaxRatio(X)
```

```

%Takes a vector input X, which defines the tendon radii connections
%Outputs the Dmax/Dmin ratio from the resulting eigenpostures
%Use this output as a cost function to optimize the choice of X
%
% Usage:
% Xopt=fmincon(@(x)findMaxRatio(x),Xinit,[],[],[],[],LowerBound,UpperBound)

```

```

%Load the posture matrix
data = [30    20    10    30    20    10    30    20    10    30    25
        25    30    20    0     0     0     0     0    45;
0       0     0     0     0     0     0     0     0     0     0
        0     0     0     0     0     0     0     0;
30      45    40    30    45    45    30    45    45    30    45    45
        15    30     0    10     0     0     0     0    90;
15      30    17    15    20    60    20    25    65    20    25    65
        30    20     0    10     0     0     0     0    60;
21      35    40    20    28    60    20    25    68    20    25    65
        10    20     0     0     0     0     0     0    90;
45      70    80    45    70    80    45    70    80    45    70    80
        10    40    30     0     0     0     0     0    60;
15      30    17    15    20    60    40    65    75    40    65    75
        10    40    20    10     0     0     0     0    52;
25      40    35    25    40    40    25    40    40    25    40    40
        15    30     0    10     0     0     0     0    90;
35      70    80    35    70    80    37    70    80    38    70    80
        5     30    45     0     0     0     0     0    60;
50      82    85    50    82    85    50    82    85    50    82    85
        35    40     0     0     0     0     0     0    75;
30      27    55    40    50    65    50    75    85    50    75    85
        15    30    20     0     0     0     0     0    65;
0       0     0    15    20    60    20    25    68    35    40    80
        0     0    28    10     0     0     0     0    55;
30      22    50    40    50    65    50    75    85    50    75    85
        15    35    20    10     0     0     0     0    60;
30      27    55    20    35    35    15    30    30    10    25    22
        15    30    20     0     0     0     0     0    65;

```

```

25      27      55      30      22      62      30      27      55      30      27      55
      15      30      5       0       0       0       0      75];

```

```

data = data*pi/180;

```

```

%Convert to tendon-space given the input geometry X

```

```

for i = 1:15

```

```

    T(i,1) = X(1)*data(i,1) + X(2)*data(i,2) + X(3)*data(i,3);    %Index Distal

```

```

    T(i,2) = X(3)*data(i,3);                                     %Index Proximal

```

```

    T(i,3) = X(4)*data(i,4) + X(5)*data(i,5) + X(6)*data(i,6);    %Middle Distal

```

```

    T(i,4) = X(6)*data(i,6);                                     %Middle Proximal

```

```

    T(i,5) = X(7)*data(i,7) + X(8)*data(i,8) + X(9)*data(i,9);    %Ring All

```

```

    T(i,6) = X(10)*data(i,10) + X(11)*data(i,11) + X(12)*data(i,12); %Pinky All

```

```

    T(i,7) = X(13)*data(i,13) + X(14)*data(i,14);                %Thumb Distal

```

```

    T(i,8) = X(15)*data(i,15);                                    %Thumb Abdduction

```

```

    T(i,9) = X(16)*data(i,16);                                    %Index Abdduction

```

```

    T(i,10) = X(17)*data(i,20);                                   %Thumb Rotation

```

```

end

```

```

%Calculate principal components

```

```

[c s l t] = princomp(T);

```

```

modes = c';

```

```

prinmodes = modes(1:2,:);

```

```

prinweights = s(:,1:2);

```

```

means = repmat(mean(T),15,1);

```

```

%Calculate eigenpostures and max/min ratios

```

```

Eig1 = prinmodes(1,:);

```

```

Eig2 = prinmodes(2,:);

```

```

Eig1MaxRatio = max(abs(Eig1))/min(abs(Eig1));

```

```

Eig2MaxRatio = max(abs(Eig2))/min(abs(Eig2));

```

```

%Modify eigenpostures if ratios excessively large

```

```

if(Eig1MaxRatio > 50)

```

```

    [ignore index] = min(abs(Eig1));

```

```

    Eig1(index) = 0;

```

```

        temp = Eig1;

```

```

    temp(index) = [];

```

```

    Eig1MaxRatio = max(abs(temp))/min(abs(temp));

```

```

end

```

```

%Output the max of the max ratios found

```

```

maxRatio = max([Eig1MaxRatio Eig2MaxRatio]);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Clustering Analysis
%   This program implements k-means clustering and produces the grouping and average
%   error plots for several different scenarios
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc

%Load posture matrix
data = [
    30  20  10  30  20  10  30  20  10  30  25  25  30  20  0  0  45;
    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
    30  45  40  30  45  45  30  45  45  30  45  45  15  30  0  10  90;
    15  30  17  15  20  60  20  25  65  20  25  65  30  20  0  10  60;
    21  35  40  20  28  60  20  25  68  20  25  65  10  20  0  0  90;
    45  70  80  45  70  80  45  70  80  45  70  80  10  40  30  0  60;
    15  30  17  15  20  60  40  65  75  40  65  75  10  40  20  10  52;
    25  40  35  25  40  40  25  40  40  25  40  40  15  30  0  10  90;
    35  70  80  35  70  80  37  70  80  38  70  80  5  30  45  0  60;
    50  82  85  50  82  85  50  82  85  50  82  85  35  40  0  0  75;
    30  27  55  40  50  65  50  75  85  50  75  85  15  30  20  0  65;
    0  0  0  15  20  60  20  25  68  35  40  80  0  0  28  10  55;
    30  22  50  40  50  65  50  75  85  50  75  85  15  35  20  10  60;
    30  27  55  20  35  35  15  30  30  10  25  22  15  30  20  0  65;
    25  27  55  30  22  62  30  27  55  30  27  55  15  30  5  0  75];

dataSize = size(data);
numPostures = dataSize(1);
numJoints = dataSize(2);

%Plot errors with no clustering

plot(1:numJoints,pcars(data,2))
title({'Joint Angle Errors With No Grouping','2 Eigenpostures'})
xlabel('Joint Number')
ylabel('Error in Degrees')
axis([1 numJoints -30 30])

figure
plot(1:numJoints,pcars(data,4))
title({'Joint Angle Errors With No Grouping','4 Eigenpostures'})
xlabel('Joint Number')
ylabel('Error in Degrees')
axis([1 numJoints -30 30])

```

```

%% ***** %%
%% Begin K-means Clustering Algorithm for Grouping Analysis %%
%% ***** %%
numGroups = 2;

bestOuterMSE = inf;
s = 1;
for outerloop = 1:100    %Outer loop restarts after each convergence to avoid local sol'n
    centers = zeros(numGroups,numJoints);
    for i = 1:numGroups
        for j = 1:(numJoints)
            maxAngle = max(data(:,j));
            minAngle = min(data(:,j));
            centers(i,j) = (maxAngle-minAngle)*rand    %Start with random centers
        end
    end

    oldMSE = inf;
    MSE = 0;
    l = 0;

    Q = zeros(numPostures,numGroups);
    %while MSE < oldMSE    %Uncomment both lines and re-comment next line to try to
    % use MSE to judge stopping time
    %    l = l+1

    for l = 1:100
        %Find nearest center for each posture
        for j = 1:numPostures
            minD = inf;
            for k = 1:numGroups
                D = norm(data(j,:)-centers(k,:));
                if D < minD
                    minD = D;
                    indexMin = k;
                end
            end
            for i = 1:numGroups
                if indexMin == i
                    Q(j,i) = 1;
                else
                    Q(j,i) = 0;
                end
            end
        end
    end
end

```

```

    for k = 1:numGroups
        for p = 1:numJoints
            centers(k,p) = data(:,p)'*Q(:,k)/sum(Q(:,k));
        end
    end

    if(l>1)
        oldMSE = MSE;
    end

    MSE = 0;
    for i = 1:numGroups
        for j = 1:numPostures
            MSE = MSE + (norm(data(j,:)-centers(i,:)))^2*Q(j,i);
        end
    end
    MSEs(l) = MSE;
end
outerMSE = MSE;
if(outerMSE < bestOuterMSE)
    bestOuterMSE = outerMSE;
    Qbest = Q;
    centerBest = centers;
    bestOuterMSEvector(s) = bestOuterMSE;
    s = s+1;
end
end

Q = Qbest;
figure
plot(bestOuterMSEvector)
title('Mean Squared Error')
xlabel('Iterations')
ylabel('MSE')

%Divide DATA into 2 groups (won't work for arbitrary numGroups, have to
%change this section by hand :(
group1rows = [];
group2rows = [];
for i = 1:numPostures
    if Q(i,1)
        group1rows = [group1rows i];
    end
    if Q(i,2)
        group2rows = [group2rows i];
    end
end

```

```

    end
end

dataSet1 = data(group1rows, :);
dataSet2 = data(group2rows, :);

res1 = pcares(dataSet1,2);
res2 = pcares(dataSet2,2);
restotal2 = [res1;res2];

figure
plot(1:numJoints,restotal2)
title({'Joint Angle Errors With Grouping','2 Eigenpostures'})
xlabel('Joint Number')
ylabel('Error in Degrees')
axis([1 numJoints -30 30])

res1 = pcares(dataSet1,4);
res2 = pcares(dataSet2,4);
restotal4 = [res1;res2];

figure
plot(1:numJoints,restotal4)
title({'Joint Angle Errors With Grouping','4 Eigenpostures'})
xlabel('Joint Number')
ylabel('Error in Degrees')
axis([1 numJoints -30 30])

figure
plot(1:numJoints,mean(abs(pcares(data,2))))
hold on
plot(1:numJoints,mean(abs(restotal2)), 'r--')
legend('No grouping','With Grouping')
title('Average error, 2 Eigenpostures')
xlabel('Joint Number')
ylabel('Error in Degrees')

figure
plot(1:numJoints,mean(abs(pcares(data,4))))
hold on
plot(1:numJoints,mean(abs(restotal4)), 'r--')
legend('No grouping','With Grouping')
title('Average error, 4 Eigenpostures')
xlabel('Joint Number')
ylabel('Error in Degrees')

```

```
clc
```

```
disp(['Average Error, 2 Eigenpostures, no grouping: '
num2str(mean(mean(abs(pcares(data,2)))))]
disp(['Average Error, 2 Eigenpostures, WITH grouping: '
num2str(mean(mean(abs(restotal2)))))]
disp(['Average Error, 4 Eigenpostures, no grouping: '
num2str(mean(mean(abs(pcares(data,4)))))]
disp(['Average Error, 4 Eigenpostures, WITH grouping: '
num2str(mean(mean(abs(restotal4)))))]
fprintf('\n')
disp(['Average Max Error, 2 Eigenpostures, no grouping: '
num2str(mean(max(abs(pcares(data,2)))))]
disp(['Average Max Error, 2 Eigenpostures, WITH grouping: '
num2str(mean(max(abs(restotal2)))))]
disp(['Average Max Error, 4 Eigenpostures, no grouping: '
num2str(mean(max(abs(pcares(data,4)))))]
disp(['Average Max Error, 4 Eigenpostures, WITH grouping: '
num2str(mean(max(abs(restotal4)))))]
fprintf('\n')
disp(['Max Error, 2 Eigenpostures, no grouping: '
num2str(max(max(abs(pcares(data,2)))))]
disp(['Max Error, 2 Eigenpostures, WITH grouping: '
num2str(max(max(abs(restotal2)))))]
disp(['Max Error, 4 Eigenpostures, no grouping: '
num2str(max(max(abs(pcares(data,4)))))]
disp(['Max Error, 4 Eigenpostures, WITH grouping: '
num2str(max(max(abs(restotal4)))))]

[c s] = princomp(data);
weights = s(:,1:2);
modes = c';
modes = modes(1:2,:);
figure
plot(weights(group1rows,1),weights(group1rows,2),'+')
hold on
plot(weights(group2rows,1),weights(group2rows,2),'r^')

centersAdj = centers - repmat(mean(centers),numGroups,1);
centerWeights = centersAdj/modes;
if(Qbest(2,1) & (centerWeights(1,1)>centerWeights(2,1)))
    centerWeights = [centerWeights(2,:);centerWeights(1,:)];
end
plot(centerWeights(1,1),centerWeights(1,2),'o');
```

```

plot(centerWeights(2,1),centerWeights(2,2),'or');

Dmax = zeros(1,2);
D = zeros(1,numPostures);
for i = 1:numPostures
    for j = 1:numGroups
        if Q(i,j)
            D(i) = norm(weights(i,:)-centerWeights(j,:));
            if (D(i) > Dmax(j))
                Dmax(j) = D(i);
            end
        end
    end
end
end

theta = linspace(0,2*pi);
plot(Dmax(1)*cos(theta)+centerWeights(1,1),Dmax(1)*sin(theta)+centerWeights(1,2))
plot(Dmax(2)*cos(theta)+centerWeights(2,1),Dmax(2)*sin(theta)+centerWeights(2,2),'r')
axis([-160 160 -160 160])
axis square
title('Grouping Details')
xlabel('1st Component Weight')
ylabel('2nd Component Weight')

```