Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Versatile Toolbox for Studying Anthropomorphic Robot Hands

Jorge Nicho
Mechanical Engineering Department
The University of Texas at San Antonio
San Antonio, TX, USA
ivd394@my.utsa.edu

Brent Nowak, Ph.D.
Mechanical Engineering Department
The University of Texas at San Antonio
San Antonio, TX, USA
brent.nowak@utsa.edu

Abstract— **Recent efforts to develop virtual environments for the study of robot hands and grasping have been limited. The general approach to modeling the hand has overlooked many fundamental aspects that relate to the mechanical design of robot hand prototypes. In particular, resulting motions of hands due to tendon action have not been simulated in any existing virtual environment, to our knowledge. Furthermore, existing simulation environments do not incorporate generalized low-level control architectures for such complex mechanisms. In response to these needs, we present the ongoing development of our toolbox with the aim of providing a sufficient number of tools for modeling and simulating tendon-driven anthropomorphic hand mechanisms.**

Keywords—**analytic modeling, robotics, grasp theory, end-effectors, animation, robot hands**

I. Background and Introduction

Many industrial manufacturing processes require machines that provide a high degree of precision, dexterity and versatility. It is common to produce end-of-tooling devices specific to very specialized applications. However, these end-of-tooling mechanisms cannot be extended to other tasks that demand different levels of precision, complexity, dexterity, and/or versatility. This inherent limitation of current end-effectors has significant cost and time impacts in many aspects of manufacturing processes. For several years, there has been an increasing interest in studying advanced mechanisms with superior grasping and manipulative capabilities. In particular, special attention has been paid to dexterous mechanisms whose characteristics resemble in many aspects those of the human hand.

Justification for the study of this biological instrument is found in the fact that no other mechanism in nature has been shown to neither equal nor approach the versatility and precision that are evident in the human hand. Consequently, a device with such characteristics should contribute to the improvement of many key aspects of most manufacturing processes such as cycle time, cost and task complexity. Also, successful reproduction of the capabilities of the human hand can extend its use to other important areas such as prosthetics, exoskeleton, haptics and telepresence [16]. These reasons have been driving factors that have guided most of the research in this direction.

The human hand is a tendon driven mechanism that possesses four 4-Degree-Of-Freedom (4-DOF) fingers connected in parallel and a 5-DOF opposable thumb. Each finger is an under-actuated serial mechanism that can extend or retract (flexion) and can rotate about a vector that passes throughout the plane of flexion motion (abduction) [8]. With the addition of the motions produced at the wrist and the palm, the hand has about 27 DOF. This level of redundancy is what enables the hand to cope with an innumerable number of tasks of various levels of difficulty and requirements leading to its enviable dexterity.

There have been various attempts to produce physical models that closely resemble the human hand. Among those are the MIT hand, the Belgrade Hand, the NASA Robonaut Hand and the commercially available Shadow hand, see [14], [15], and [16]. It is found that the majority of these hands incorporate cable, actuators and supporting electronic components located at the wrist. As a result, this type of design not only resembles the human hand, but it has been able to resolve many mechanical issues such as weight, size, modularity and mechanism assembly. The most sophisticated hands incorporate a wide range of tactile sensors, force sensors and encoders placed in many different locations throughout the hand.

For a long time, using test-beds has been the preferred manner in which to conduct practical experiments to validate new or existing grasping methods and algorithms as found in [10], [11], [12], [13], and [14]. Nevertheless, prototyping robotic hands tend to be extremely costly and difficult and a number of issues regarding hardware interface, data acquisition and acceptable performance always arise. Furthermore, grasping strategies cannot be evaluated in different hands since only one unique experimental test-bed is normally available.

In recent times, a limited number of investigators have developed or have used existing software applications to provide a rich and sound single grasping simulation

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

environment that avoids the shortcomings of experimenting with mechanical prototypes. The most important work in this category was the development of GraspIt!, see [1], [2] and [7]. This notable work is an extensive simulation environment that allows carrying out grasping operations with imported CAD models. In addition, GraspIt! incorporates grasp analytical methods that provide quantitative and qualitative metrics that measure the stability of a particular grasp. Other studies such as [3], [5] and [6] have been conducted with commercially available software called ADAMS™ (MSC Software™). In [4], a 3D graphic model of a hand was generated. These works will be described more in detail in Section II.

Nevertheless, plenty of progress remains to be made in the area of robotic hand simulation. Issues such as accurate dynamic representation for simultaneous operation of multiple nonlinear serial chain mechanisms and autonomous grasp planning have yet to be explored in detail. Moreover, motions that result from tendon action have not been implemented. Additionally, integrating hardware-in-the-loop (HIL) of a simulated robot hand within a single environment is a topic that has yet to be addressed.

The aim of this work is to introduce, and to discuss the progress and future work of our toolbox for accurate simulation of the motions of tendon-driven anthropomorphic hands. Of great interest to our work is to accurately model the overall effect of tendons on the kinematics and dynamics of different hands with anthropomorphic design. It is planned to write a number of routines that can automatically build the mathematical kinematic and dynamic models. Moreover, it is intended to give the toolbox the capability to readily input and evaluate control algorithms for any particular hand and finger model. In its current state, this toolbox enables quick generation of finger and hand prototypes of a multitude of sizes and kinematic configurations. Additionally, it is capable of forward kinematics simulation and animations of fingers and hand mechanisms. The toolbox will be described more in detail in Section III.

This paper is structured as follows: Section II reviews current and past simulation environment of virtual robot hand models. Section III introduces the library of functions and tools of the Robot Hand Toolbox. Finally, Section IV describes the short and long term goals of this work.

II. Review of Robot Hand Simulation

In [4], a graphical robot hand with a simplistic bone structure was developed with the use of the graphics tools 3D Studio Max and Maya 3D. Some material properties and lightning effects were added to this model as well. Results produced from kinematic and dynamic analysis were presented, however they do not account or include the effects of tendon driven fingers.

In [7], Miller developed a virtual platform that could perform animations of specified hand posture sequences inside a virtual and interactive world. It incorporated a real-time collision detection system that prevented objects from passing through others during the animation. Another important feature was the capability of determining contact location. These two features made it possible to use existing methods that gave quality measures of the stability of a grasp. Moreover, object dynamics could be performed in combination with collision detection in order to produce more realistic grasping operations. However, this combined operation increased the computational load and hindered performance. Clearly, this is a very rich tool in which very meticulous studies of the mechanics of grasping could be carried out. However, finger joints were driven by directly commanding joint angles and resulting motions of tendon-driven fingers were not considered.

Additionally, other studies of grasping such as [3], [5] and [6] were carried out with the use of the commercial software package called ADAMS. This software is a modeling and simulation tool that allows studying the performance of designs of mechanical systems from the kinematic, static and dynamic perspective. However, the majority of its capabilities are tailored to automotive and aeronautics applications. Specialized tools for the analysis of grasping and tendon-driven hands are not available with this software.

III. Robot Hand Toolbox

As previously described, the toolbox can generate virtual finger and hand models with relative ease. Up until this point, the existing function library has been written in MatLab™ (MathWorks™). Therefore, the majority of functions tend to adhere to standard MatLab syntax that most users are familiar with. Furthermore, the toolbox makes extensive use of the object-oriented programming capabilities such as inheritance and member attributes and events.

The current version of the toolbox includes three base classes, “Solid”, “Finger” and “Hand”, one derived class, “Phalanx” and two supporting classes, “FingerTemplate” and “PhalanxTemplate”. This type of architecture allows separating routines that define construction and geometry details from those that implement object instantiation.

Furthermore, this setup enables extending the existing number of graphical finger and hand models, as long as their corresponding CAD files are provided. Only a few lines of code need to be written in order to make a newly added CAD model functional. We find this to be advantageous since the number of finger and hand models that can be added is virtually limitless.

A brief description of each class and their relevance to the overall functionality of the toolbox is provided below:

a) Solid: Reads geometry data from CAD files saved as "STL" files and uses it to rebuild the graphical model in the MatLab environment. Additionally, it provides utility subroutines that enable conducting a variety of tasks such as resizing, animating, and changing frame of reference. Moreover, the overloaded "plus" method adds "Solid" objects together in order to build a new composite "Solid" object. All of the classes in the toolbox that operate on graphics at some level, either subclass or contain instances of the "Solid" super class.

b) Phalanx: Inherits its methods and properties from the "Solid" class and builds a finger link (phalanx) according to the specifications defined by an instance of the "PhalanxTemplate" class. Each instance contains a unique anonymous frame transform function, which is used by the "Finger" class during animation.

c) PhalanxTemplate: It specifies construction details that are used by the "Phalanx" constructor during object instantiation. In this class, the "Type" property indicates the type of finger component that is to be constructed such as proximal, middle, distal, base, etc. The Style property refers to the common geometric and shape attributes that are shared by a specific group of finger components. Hence, it's possible to incorporate new component styles as long as the corresponding CAD files exist in the toolbox directory.

d) FingerTemplate: Its role is analogous to the phalanx template class in that it lays out the construction details of a "Finger" object. An instance of this class contains the kinematic definition of a particular finger type.

e) Finger: It instantiates an object of the "Finger" class in accordance to the parameters provided by a "FingerTemplate" object. The "Phalanges" property contains all of the "Phalanx" objects that form a single finger. This class also implements member functions for animation and set property subroutines.

f) Hand: It constructs a "Hand" object and defines its own animation methods. References to "Finger" objects are stored in one of the Hand object properties. It is possible to animate several fingers simultaneously as well.

Lighting and surface reflectance algorithms have been added in order to improve visualization of all graphical entities. The code and its corresponding graphical output are shown in Fig.1.

The Standard DH (Denavit-Hartenberg) algorithm [9] has been implemented in order to automatically generate the frame assignment that defines the kinematics of a finger object. The frame assignment, corresponding to each degree of freedom, is determined by translating and rotating the movable frame with respect to a fixed frame of reference.

```
figure(1);
dimensions=[4 1.4 1.2];
s=Solid('trial_middle1',dimensions)';

figure(2);
style=1;
dimensions=[8 3 4];
p=Phalanx('middle',style,dimensions)';
```

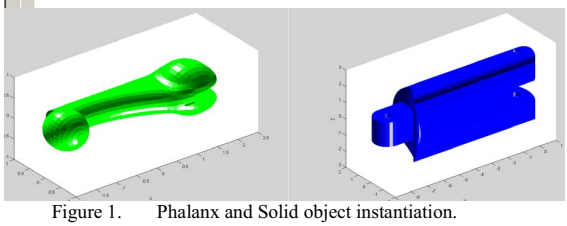

Figure 1. Phalanx and Solid object instantiation.

The actions of translation and rotation are represented as matrix multiplications of basic transformation matrices. Consequently, a resulting homogeneous transformation is defined as follows:

$$T(i) = T(i-1) \times Rot_z(\theta_i) \times Transl_z(d_i) \times Transl_x(a_i) \times Rot_x(\alpha_i) \quad (1)$$

However, the user is not required to input or define the kinematics at this level since known kinematic models of anthropomorphic fingers have been already predefined. A unique callback function containing the corresponding homogeneous transformation is attached to each mobile finger element. This function enables obtaining the corresponding transformation matrix for any value of the joint variable. Regardless of the fact that the kinematic configuration is partially predefined, the width and height, as well as, the number of finger elements (phalanx) can still be set by the user. We chose to predefine the kinematic configuration in order to simplify the model generation while maintaining our focus in the study of finger and hand mechanisms. The current kinematic models available in the toolbox are show in Table 1.

TABLE I. Default Kinematic Models

Finger Type	**Default DH Parameters**				
	Frame	*A*	*Alpha*	*D*	*Theta (variable)*
Index 1 4 DOF	1	0	pi/2	0	flexion/extension
	2	L1	-pi/2	0	abduction/adduction
	3	L2	0	0	flexion/extension
	4	L3	0	0	flexion/extension
Index 2 3 DOF	*Frame*	*A*	*Alpha*	*D*	*Theta (variable)*
	1	L1	0	0	flexion/extension
	2	L2	0	0	flexion/extension
	3	L3	0	0	flexion/extension
Index 3 4 DOF	*Frame*	*A*	*Alpha*	*D*	*Theta (variable)*
	1	0	-pi/2	0	abduction/adduction
	2	L1	0	0	flexion/extension
	3	L2	0	0	flexion/extension
	4	L3	0	0	flexion/extension
Thumb 1 4 DOF	*Frame*	*A*	*Alpha*	*D*	*Theta (variable)*
	1	0	pi/2	0	abduction/adduction
	2	L1	0	0	flexion/extension
	3	L2	0	0	flexion/extension
	4	L3	0	0	flexion/extension

The syntax and kinematic definition remains consistent with those found in the literature on human fingers [8]. Though, it does not strictly adhere to these models so that a reasonable amount of flexibility is still maintained. Fig. 2 demonstrates the relative ease with which a finger object can be generated. In this case, the "lengths" argument specifies the length of each phalanx in the finger. Since a "type" argument was not provided, the constructor assigns it a default value of "index 1" and the corresponding kinematic configuration is used. The "ctranspose" operator, denoted by an apostrophe, is added at the end of the finger constructor in order to produce a figure with the corresponding graphical output as shown in Fig 3.

A finger object is also generated in the MatLab workspace and its properties can be queried through standard "get" methods. For example, typing the following code as in Fig. 4 will display the DH parameters as well as the average size of the finger in its fully extended position.

```
lengths=[3 3 2];
f=Finger(lengths)';
```

Figure 2. MatLab finger creation code.

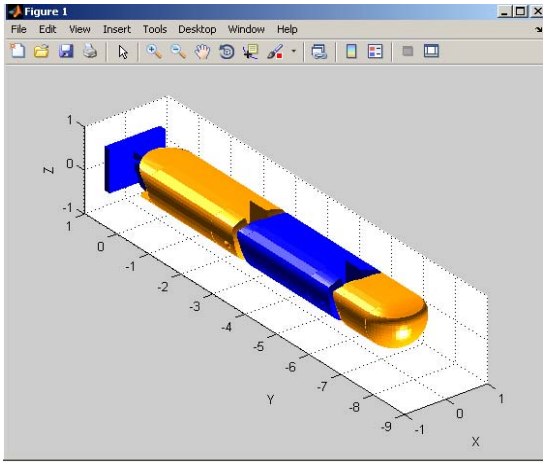

Figure 3. Resulting graphical finger object.

```
>> f.DH

ans =

         0    1.5708         0         0
    3.0000   -1.5708         0         0
    3.0000         0         0         0
    2.0000         0         0         0

>> f.Size

ans =

    8.0000    1.0000    1.3333
```

Figure 4. Using "get" methods to query object properties.

A variety of operations can be performed on finger objects after instantiation through "set" methods as shown in Fig. 5. In this example, three finger objects with different lengths are instantiated. Furthermore, the base frames of the first and second finger are translated along the x-axis. Then, the joint variables of the first finger are set to the designated joint values. Lastly, the transpose operator is called in order to display the fingers with their current settings. The results of these operations are displayed in Fig 6.

Even though only one argument is necessary to generate a finger object, additional optional arguments can be specified in order to produce fingers of various characteristics. These additional arguments include: Type, Base Frame, Joint offset, and style. They default to preset values whenever they are not specified as arguments in the finger constructor function. All of their values are saved as properties in the finger object and can be accessed with the "get" methods.

In a very similar manner, Hand objects can be created by entering existing finger objects as arguments to the hand constructor function. An algorithm automatically generates a graphical structure (palm) whose geometry adapts to the number of fingers and the dimensions of each. The properties of a hand object can also be queried. A copy of each finger object is stored as part of the hand object to whom it belongs.

```
figure(1);

f(1)=Finger([3 3 2],'index1');
f(2)=Finger([5 4 2],'index2');
f(3)=Finger([3 3 4 2],'index3');

f(1).Base=translation('x',-2)*f(1).Base;
f(2).Base=translation('x',2)*f(2).Base;
f(1).CurrentJoints=[pi/6 0 pi/6 pi/6];

f';
```

Figure 5. Various Operations with Finger Objects.

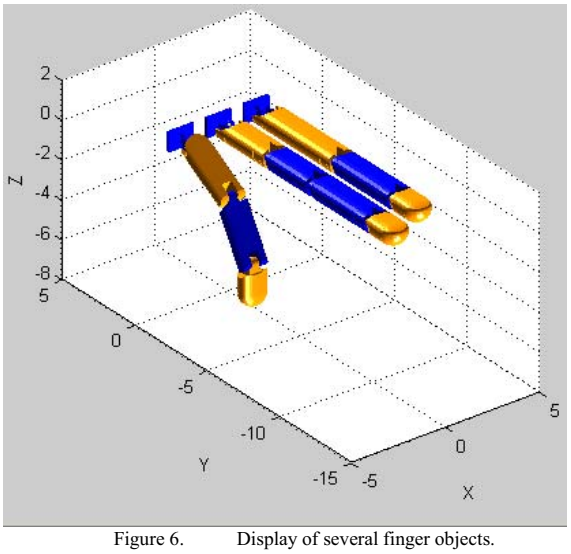

Figure 6. Display of several finger objects.

A greater level of detail can also be specified for use in the construction of the palm. For instance, position of each finger with respect to the Hand frame of reference can be set fairly easy. Other optional arguments allow setting the hand frame of reference, the size of the palm as well as the orientation. We consider that these additional options provide the necessary means to model accurate representations of human-like hands. Various hand objects are shown in Fig. 7. Moreover, animations can be easily accomplished by entering the desired joint commands as arguments to the "drive" public member function. Therefore, innumerable postures are achievable for either a finger or hand object. Several Postures are shown in Fig. 8.

IV. Future Work and Conclusion

Here, a toolbox for the study of robot hands has been introduced. Easy to use functions allowed creating finger and hand models with a generous amount of detail. Also, Forward Kinematics routines generate data and produce animations.

One of the immediate goals is to study and include the effect of tendons on the overall kinematics of each finger model. Improvements to the graphical representation of each finger will be made and more predefined finger models will be added. Additionally, existing functions will be expanded to allow users to specify greater level of detail in the construction of fingers.

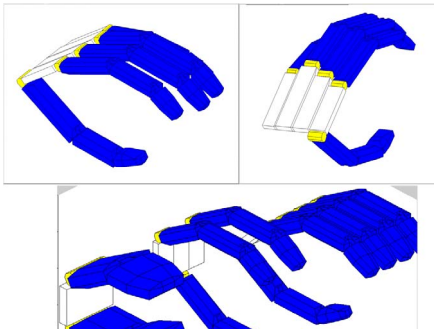

Figure 7. Various Hand Objects.

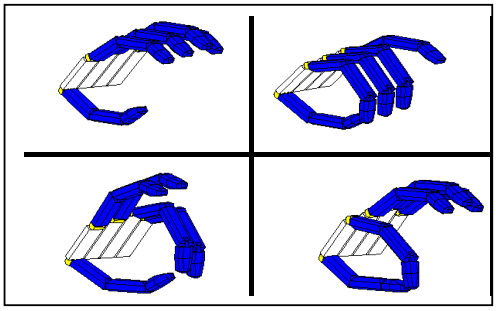

Figure 8. Multiple Postures.

Furthermore, it is intended to include routines capable of automatically generating accurate dynamic models. Perhaps, the Newton-Euler recursive algorithm will be the best choice for simulating the motions of tendon-driven hand mechanism. Routines with high computational burden will be written in C and called from within the MatLab environment.

In the long run, the kinematic and dynamic models will be used in conjunction with the power of Simulink® (MathWorks™) in order to obtain adequate control loops for robot hands. Hence, hardware in the loop operations can be carried out by making the minimum number of changes to the control loop of a virtual hand model.

References

[1] D. Kragic, A. T. Miller and P. K. Allen, "RealTime Tracking Meets Online Grasp Planning" Proceedings of the 2001 IEEE, International Conference on Robotics & Automation, May 21-26, 2001.

[2] Z. Xue, J. M. Zoellner and R. Dillman, "Planning Regrasp Operations For A Multifingered Robotic Hand" 4th IEEE Conference on Automation Science and Engineering, August 23-26,2008.

[3] L. Zollo, S. Roccella, E. Guglielmelli, M. C. Carrozza and P. Dario, "Biomechatronic Design and Control of an Anthropomorphic Artificial Hand for Prosthetic and Robotic Applications" IEEE/ASME Transactions on Mechatronics, Vol. 12, No. 4, August 2007.

[4] S. Ramasamy and M.R. Arshad, "Robotic Hand Simulation With Kinematic and Dynamic Analysis" IEEE 2000.

[5] Yuru Zhang, Jiting Li, Jianfeng Li. "Robotic Dexterous Hand: Modeling, Planning, and Simulation" Mechanical Press, Beijing. 2007

[6] V. Nguyen, S. Oh, J. Lim,C. Kang and S. Han, " A Study on Design of Three - Finger Hand System" International Conference on Control, Automation and Systems 2008, Oct. 14-17, 2088 in COEX, Seoul, Korea.

[7] A. T. Miller, "GraspIt!: A Versatile Simulator for Robotic Grasping" Thesis, Columbia University 2001.

[8] A. El-Sawah, N. D. Georganas, E. M. Petriu, "Finger Inverse Kinematics Using Error Model Analysis for Gesture Enabled Navigation in Virtual Environments" HAVE 2006 – IEEE International Workshop on Haptic Audio Visual Environments and their Application, Ottawa, Canada 4-5 November 2006.

[9] *www.cs.duke.edu/brd/Teaching/Bio/asmb/current/Papers/chap3-**forward-kinematics**.pdf*

[10] A. K. Bejczy and R. Tomovic, "Pattern Recognition and Control in Manipulators"

[11] G. Bekey and R. Tomovic, "Biologically Based Robot Control" Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 12, No. 5, 1990.

[12] D. Gunji, Y. Mizoguchi, S. Teshigawara, A. Ming, A. Namiki, M. Ishikawa and M. Shimojo, “Grasping Force Control of Multi-fingered Robot Hand based on Slip Detection Using Tactile Sensor” SICE Annual Conference 2008, August 20-22, 2008, The University Elector-Communications, Japan.

[13] A. M. Dollar, “Simple, Reliable Robotic Grasping for Human Environments” IEEE 2008.

[14] R. Platt, “Learning Grasp Strategies Composed of Contact Relative Motions” Nasa.gov

[15] S.C. Jacobsen, E.K. Iversen, D.F. Knutti, R.T. Johnson, K.B. Biggers, “Design Of The UTAH/M.I.T. Dexterous Hand” IEEE, 1986.

[16] Shadow Robot Company, Shadow Dexterous Hand C5 Technical Specification, November 8, 2006.

[17] R. Tomovic and G. Boni, “An Adaptive Artificial Hand” IRE Transactions on Automatic Control, January 15, 1962.