From Robotic Hands to Human Hands: A Visualization and Simulation Engine for Grasping Research

A. Miller[†], P. Allen[†], V. Santos[‡], F. Valero-Cuevas[‡*]
[†]Dept. of Computer Science, Columbia University, NY, USA
[‡]Neuromuscular Biomechanics Laboratory, Cornell University, NY, USA
[*]The Hospital for Special Surgery, NY, USA
E-mail: {amiller, allen}@cs.columbia.edu,{vj14, fv24}@cornell.edu

Abstract

Robotic hands are still a long way from matching the grasping and manipulation capability of their human counterparts. One way to push the research further along is to use computer modeling and simulation to learn more about human and robotic grasping. We have created a publicly available simulator to serve this purpose. It can accommodate a wide variety of hand designs, and it can evaluate grasps formed by these hands, as well as perform full dynamic simulation of the grasping process. In this paper, we present the various components of the system, and we describe two projects which use it as an integral part of larger grasp planning systems. We also discuss the development of a human hand model that uses accurate geometry and kinematics derived from experimental measurements. This is part of our current project to create a biomechanically realistic human hand model to better understand what features are most important to mimic in the designs of robotic hands.

1 Introduction

The progress in building capable robotic hands has been slow. An important factor that has affected the progress in this field is the lack of easily obtainable, low cost experimental robotic hands that can be used as test beds. The high cost and difficulty of building a robotic hand, along with the associated electronics, control systems, and integrated sensing has led to a serious lack of experimental devices in the field. In fact, were one desirous of purchasing a robotic hand today, there appear to be few, if any, available. Custom designs exist, but usually are lacking in the higher levels of system integration that can turn a mechanical device into a full-fledged grasping system.

What can push this research further along? How can we design intelligent systems with the ability to grasp objects? A partial solution to this problem is to use computer modeling and simulation to effectively design and test robotic hands in typical task environments. As the computer models get better, and the hardware faster, realistic simulation can be used to learn more about robotic grasping.

At Columbia University, we have created a publicly available simulator to serve as a useful tool for grasping research. The system, called *GraspIt!*[1], can accommodate a wide variety of hand and robot designs. It includes a rapid collision detection and contact determination system that allows a user to interactively manipulate the joints of the hand and create new grasps of a target object. Each grasp is evaluated with numeric quality measures, and visualization methods allow the user to see the weak point of the grasp and create arbitrary 3D projections of the 6D grasp wrench space. The dynamics engine within *GraspIt!* computes the motions of a group of connected robot elements, such as an arm and a hand, under the influence of controlled motor forces, joint constraint forces, contact forces and external forces. This allows the dynamic simulation of an entire grasping task, as well as the ability to test custom robot control algorithms.

In this paper we present further details regarding the various components of the system, and we also describe two projects which use *GraspIt!* as an integral part of a larger grasp planning system: one generates candidate grasps using heuristics and predefined grasp taxonomies, and the other uses a support vector machine to learn high quality grasps of parametric objects. Finally, we discuss our development of a human hand model that uses accurate geometry and kinematics derived from experimental measurements. This is part of an ongoing project to create a biomechanically realistic human hand model to better understand what features are most important to mimic in the designs of robotic hands. Such a model will also enable studies of the functional abilities of the intact and impaired human hand using the rigorous mathematical framework of robotics.

2 *GraspIt!*

In building *GraspIt!*, we were aware of several commercial robotics simulators available, including Delmia's IGRIP, Flow Software Technologies' Workspace5, MCS Software's ADAMS, and the Easy-Rob system, as well as past and present research projects in robot simulation, including among others Corke's Robotics Toolbox for MATLAB [4], Speck and Klaeren's RoboSiM [29], and Ruspini and Khatib's Simpact [25]. There are a number of important elements that set *GraspIt!* apart from this body of previous work, the most important being that it has been developed *specifically* to analyze and visualize the robotic grasping task. Accordingly, none of the simulators above have the capability to compute dynamic frictional

[1]The source code for *GraspIt!* is available for download from http://www.cs.columbia.edu/~amiller/graspit.

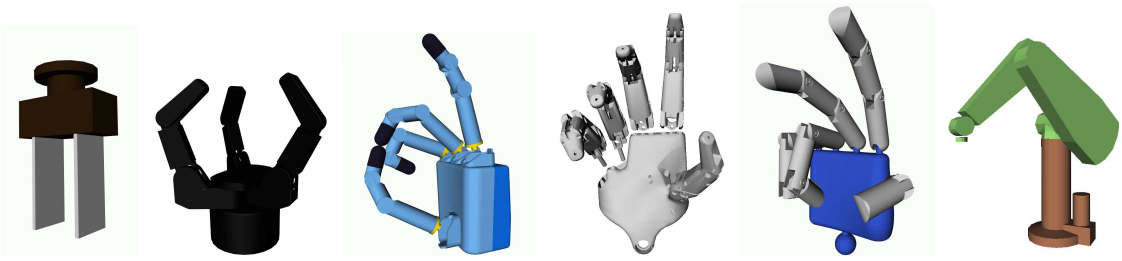

Figure 1: *GraspIt!* robot models of (from left to right): a parallel jaw gripper, Barrett hand, DLR hand, NASA Robonaut hand, Rutgers hand, and Puma 560 arm.

contact forces accurately, which is an essential component of grasping problems. Further, they also lack the grasp analysis and planning algorithms that are incorporated in *GraspIt!*. Finally, *GraspIt!* is a total package that includes not only models and a simulation engine, but a powerful and advanced user interface that facilitates its use. In the following sections we provide a brief overview of the components and features of the system, as well as describe some of its applications. Further information regarding the system and its applications can be found in the following papers[12, 14, 15, 16, 17, 23].

2.1 *GraspIt!* World Elements

Body Types: A basic body consists of a pointer to its geometry, a material specification, a list of contacts, and a transform that specifies the body's pose relative to the world coordinate system. The body geometry is read from an Inventor model file that has essentially the same format as VRML 1.0. The material is one of a set of predefined material types and is used when computing the coefficient of friction between two contacting bodies.

A dynamic body inherits all of the properties of a body and defines the mass of the body, the location of its center of mass, and its inertia tensor. If the mass distribution is unknown, the system can compute the inertia tensor by assuming a uniform mass distribution. The reason for distinguishing between bodies and dynamic bodies is that some bodies are simply considered obstacles, and while they are elements of the collision detection system and can provide contacts on other bodies, they are not part of the dynamics computations and remain static. This makes it possible to create a complex world full of obstacles without making the dynamics intractable to compute.

Robots: We have tried to make the definition of a robot as general as possible to allow a wide variety of robot designs to be imported. The system reads the kinematic parameters (specified in standard Denavit-Hartenberg notation) for each chain of links and joints from a text file, loads the geometry for each link, and constructs the robot. Our definition separates degrees of freedom (DOF) from joints and allows multiple joints to be driven by the same DOF because it is common in many hand designs to have coupled joints that are passively controlled by other joints.

A hand is a special type of robot that can form grasps of objects, and these grasps will be analyzed by the system. It also includes an auto-grasp method which closes the joints of the hand at preset velocities. Each joint stops when it has reached its limit or when a link that follows it in the kinematic chain contacts an object or another finger link.

The Robot Library: The ability to easily add new robot designs is a key benefit of our system. It is a relatively simple process of specifying the parameters required by the configuration file, creating the link geometry files, and in most cases takes less than an hour or two to set up. We have already created models of a parallel jaw gripper, a Puma 560 arm, and a simplified Nomadics XR4000 mobile robot. Additionally, through collaboration with other research sites we have obtained CAD models and kinematic descriptions of four different articulated hand designs (see figure 1).

Another feature of *GraspIt!* is the ability to attach multiple robots to create robotic platforms. The system allows the definition of a tree of robots where any number of robots can be attached to the last link of a kinematic chain of another robot. This allows us to construct more complex manipulation systems, and together with a world model specifying the robot's environment, we can plan and test our entire grasping task so that we can avoid planning grasps that will conflict with these obstacles.

2.2 User Interface

One of the design goals we believe we achieved was to make the user interface as intuitive and transparent as possible. When a user starts a new session, he is presented with an empty world into which he can import new obstacles, graspable bodies, or robots, and at any point the current state of the world can be saved to be loaded again later in another session. The primary element of the main window is a standard viewer, which displays a projection of a 3D world in a 2D window. The virtual camera can be rotated, panned, or zoomed, and a seek tool allows close up inspection of a particular detail in the scene.

When the dynamics are not being used, obstacles, graspable bodies, and robots may be translated and rotated in 3D using an appropriate manipulator that appears upon clicking on the object. Manipulating the individual degrees of freedom of a robot is equally intuitive. Clicking on a kinematic chain of the robot brings up an interactive manipulator for each actively controllable joint in the chain. Revolute joints are controlled by dragging a disc whose axis of rotation is coincident with the joint axis (see figure 2), and prismatic joints are controlled by dragging an arrow which is aligned with the joint axis. These manipulators obey the joint limits defined in the robot configuration

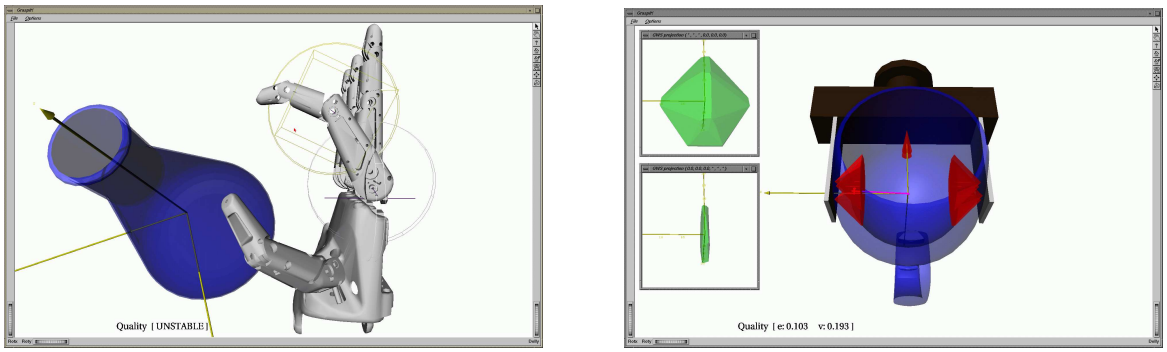

Figure 2: Left: The user interface makes manipulating joints simple. The angle of a revolute joint can be changed by dragging a disc manipulator located at the joint. The passive distal joint moves in a fixed relationship with the medial joint. The quality of a grasp is shown as "Unstable" until a force-closure grasp is formed. Right: A force-closure grasp of a mug using a parallel jaw gripper. The worst and average case quality values are displayed in the lower left portion as the values e and v. The pair of purple indicators show the force and torque components of the worst case disturbance wrench. In the upper left is a projection of the grasp wrench space that shows the space of forces that can be applied to the mug without creating a moment, and in the lower left is a projection that shows the space of torques that can be applied without a net force acting on the object.

file and prevent the user from moving beyond them.

Another important feature is the ability to interact with portions of *GraspIt!* through MATLAB. The system can be run as a server that accepts TCP socket connections, and then in MATLAB, compiled MEX functions are used to communicate with the server to set robot motor torques, step the dynamic simulation by one time step, and retrieve the state of the world including all of the contact forces and the current position and velocity of each body. This allows external MATLAB functions to control the simulation.

2.3 Contacts

To prevent bodies from passing through each other while they are being manipulated by the user, the system performs real-time collision detection using the Proximity Query Package [13]. If a collision is detected, the motion of the bodies must be reversed back to the point when the contact first occurs. To find this instant, *GraspIt!* performs a binary search that ends when the bodies are separated by a distance that is less than the contact threshold (currently set at 0.1mm). Then the system determines the convex regions of contact between the two bodies and draws a red friction cone at each contact point, which serves to visually mark the position of the contact and its normal. The width of this cone identifies how large any frictional forces can be with respect to a force applied along the contact normal.

2.4 Grasp Analysis

When *GraspIt!* is used in the static mode, the fingers of a hand may be closed around the object without causing it to move, and every time a contact is formed or broken the system evaluates the current grasp. This evaluation is done using one or more quality metrics. The current metrics evaluate the ability of the grasp to resist disturbance forces, but future metrics will evaluate object manipulability and the grasp's robustness with respect to errors in contact placement.

To evaluate a grasp's efficiency against disturbances, the system builds the 6D grasp wrench space using a convex hull operation on the set of possible contact wrenches given a unit strength grasp [14]. The volume of this space is used as an average case quality measure, since the larger the space is, the more efficient the grasp is. The point on the hull boundary that is closest to the origin represents the grasp's weakest point (i.e. the wrench that is most difficult for the grasp to apply). The distance to this point is used as a worst case quality measure.

These numeric results allow an objective comparison between grasps, but in some cases it is desirable to visualize more of the grasp's characteristics. To support this, the system can produce arbitrary 3D projections of the grasp wrench space which give a better sense of a particular grasp's strengths and weaknesses. It can also display the worst case disturbance wrench, which is the force-torque combination that is most difficult for the grasp to resist. Figure 2 shows an example of these results, and demonstrates why parallel jaw grippers are not good at grasping round objects. Because the flat plates only contact the mug surface in two places, the grasp cannot easily resist a torque about the axis between those two contact regions, as shown by the purple worst case disturbance wrench indicator.

2.5 Simulating Dynamics

The system allows the user to form grasps of an object and analyze these grasps without using dynamics. To study the evolution of a grasp and to test grasping control algorithms, however, we must consider how the hand and object move over time under the influence of controlled motor forces, gravity, inertial forces, and in response to collisions. To compute the motion of each dynamic body in the world, we use a numerical integration scheme that computes the change in velocity of each body over a small finite time step given a set of external forces acting on the body and any joint constraints, non-interpenetration constraints, and friction constraints that may also be present. These constraints are linearized and formulated as a linear complementarity problem (LCP) [1], that is solved using Lemke's algorithm [5], a pivoting method similar to the simplex algorithm for linear programming. The solution provides not only the new velocity of the bodies, but also the normal and

frictional impulses at the contact points. After each iteration of the dynamics is completed, the system can draw the contact forces at each contact point, and at any time the dynamics may be paused to examine the state of the system or to change the current simulation parameters.

With the dynamics in place, it is possible to study the temporal formation of grasps. In the example presented in figure 3, the Barrett hand is positioned above a wine glass which rests on its side on the table. The PD joint controllers of the Puma robot hold the wrist in place, and the desired final position of the hand joints is set to fully closed. The figure shows the initial setup and 5 different time slices during the simulation. The default time step is 2.5 ms, but smaller steps may occur due to contact events. Because there is very little friction between the plastic and glass surfaces, and because the glass is tapered, the hand squeezes the glass out of its grasp. As the simulation continues, the wine glass slides and rolls off the table, hitting the Puma robot on its way down to the floor.

3 Applications

GraspIt! has become a platform that supports research in a variety of areas related to grasping. With its integrated grasp analysis methods, we have applied it to the problem of planning high quality grasps of objects. Below we describe two different systems we have built with *GraspIt!* that approach this challenging problem in different ways. Finally, we present some of our most recent work in creating a biomechanically realistic human hand model that will not only help clinicians better understand the mechanics of the hand and plan reconstructive surgeries, but will also help robotic hand designers build hands that come closer to matching the capabilities of our own hands.

3.1 Grasp Planning

The grasp planning problem is extremely difficult because of the number of degrees of freedom of a robotic hand. For example, the relatively simple Barrett hand has 10 degrees of freedom, 6 for orientation relative to the object, and 4 for finger manipulation. This number of DOF's creates a large search space of hand configurations. Of course, large portions of this space are worthless because the fingers are not in contact with the object, but even if the problem were reparameterized, a brute force search would still be intractable.

A variety of other approaches have been used to tackle this problem. A number of papers present contact-level grasp synthesis algorithms [6, 18, 24]. These algorithms are concerned only with finding a fixed number of contact locations without regard to hand geometry. Other systems built for use with a particular hand restrict the problem to choosing precision fingertip grasps, where there is only one contact per finger [3, 11]. These types of grasps are good for manipulating an object, but are not necessarily the most stable grasps because they do not use inner finger surfaces or the palm.

One way of limiting the large number of possible hand configurations is to use grasp preshapes. Before grasping an object, humans unconsciously simplify the task to selecting one of only a few different prehensile postures appropriate for the object and for the task to be performed. These postures have been enumerated in various grasp taxonomies (e.g. [19]).

Our first planner takes advantage of this fact, in order to reduce the search space size to a small number of grasps that are more likely to have a high quality. The system consists of two parts, one to generate a set of starting grasp locations based on a simplified object model, and one to test the feasibility and evaluate the quality of these grasps. The simplified object model is constructed from a small set of shape primitives such as spheres, cylinders, cones and boxes, and heuristic grasping strategies for these shapes allow the system to generate a set of grasp possibilities. The grasp tester moves the hand from a grasp starting position toward the object, closes the fingers around the object, and evaluates the grasp. After testing all of the generated grasp possibilities, the user is presented with the best grasps of the object in descending order of quality (see figure 4). In order to prevent infeasible grasps from being planned, the user may import a world model containing obstacles, as well as a robot arm model so that reachability constraints may be considered (see figure 5).

The drawback of this approach is that since it only considers a subset of the possible grasps, it may miss a better possibility. Humans may initially choose a sub-optimal grasp of a novel object, but with experience they will adapt their grasp to the most appropriate one for the task. Our most recent planning system applies new machine learning techniques to this problem to effectively have robots learn how to grasp arbitrary objects.

We are using supervised training to learn what is a good grasp for a robotic hand. This requires a method that allows us to try a large number of grasps of an object and report a metric on the quality of each grasp in the training set, and *GraspIt!* is perfectly suited for this. Using this training set, we can then generate basis functions that can effectively both predict the quality of an arbitrary new set of grasping parameters and also use these basis functions to find an optimal set of grasping parameters for an object. These parameters correspond to the degrees of freedom of an actual hand, rather than the placement of point contacts on the object surface. It is also important to note that our method is not dependent on a single type of robotic hand or class of objects. It provides a robust system for testing different robotic hands and analyzing the quality space that they span.

For our tests we again used the Barrett hand, and the objects in our training set were 9 different superellipsoids, which can be described with two shape parameters, ϵ_1 and ϵ_2. For each superellipsoid we generated 1,600 grasps, which consisted of 100 random roll and spread angle combinations for each of 16 regularly sampled grasp starting positions, as shown in the left portion of figure 6. This gave us a total of 14,400 grasps, which were evaluated by GraspIt! over the course of approximately 4 hours on a Pentium IV 2.3 GHz Windows machine.

We then used an SVM regression to create a mapping between object shape, grasp parameters and grasp quality. Our learned regression mapping accepts a fixed length in-

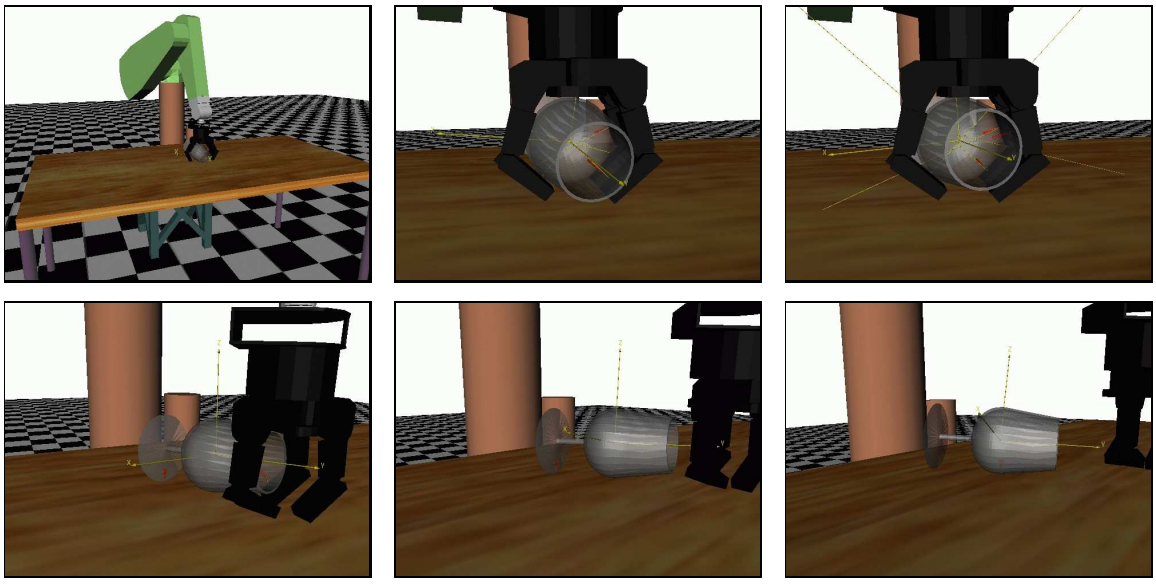

Figure 3: The Barrett hand attempts a grasp of the wine glass, but due the low degree of friction at the contacts and the taper of the glass, the glass is squeezed out of the grasp. The first frame shows the initial setup, and following from left to right, snapshots are taken at 0.5103, 0.5425, 0.6148, 0.6586, 0.6956 seconds of simulation time. The full movie is at http://www.cs.columbia.edu/~amiller/graspit/movies.html.

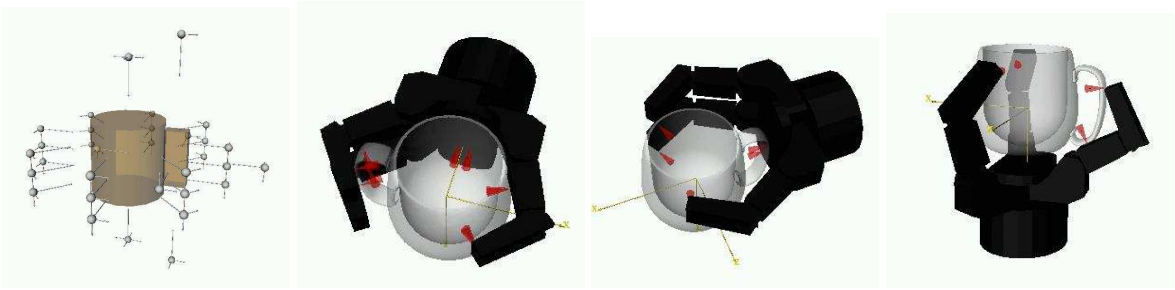

Figure 4: The first image shows the primitive model used for planning grasps for the coffee mug and shows the generated set of grasp positions to be tested. The other images show three of the best grasps of the mug in descending quality order.

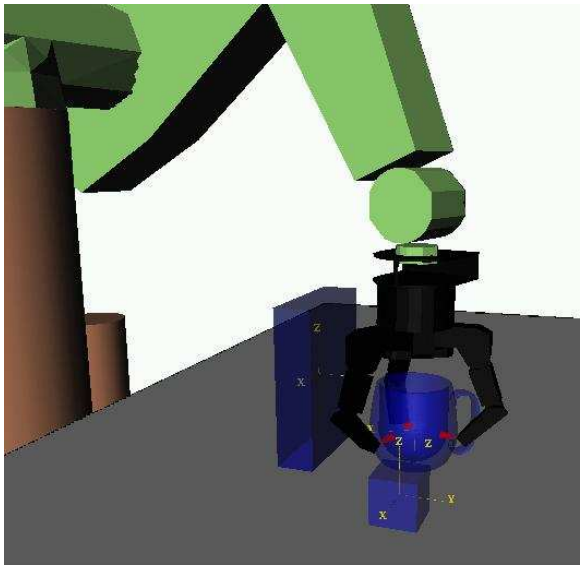

Figure 5: The best planned grasp of the mug in the presence of obstacles and using the reachability constraints of the Puma arm.

put vector that contains the shape and grasping parameters and returns a single scalar which estimates the grasp quality. If only provided with a subset of the input vector (i.e. the shape parameters), the regression algorithm will perform an efficient search for the optimal setting of the missing input vector values (i.e. the grasp parameters) that maximize grasp quality. The middle and right portions of figure 6 show the results of these searches for both a shape that was in the training set and one that wasn't.

Our initial superquadric models are clearly not able to model the full range of possible objects to be grasped, so our next task is to extend this technique to more complex shapes, which could be composite models composed of multiple superquadric shapes. Possible methods for solving this multi-superquadric grasping problem are discussed in [23].

3.2 Modeling the Human Hand

There is an inherent mismatch between the mechanical design and capabilities of robotic hands versus human hands. Why is this so? Robotics-based models of the human hand use idealized simple joints, torque motors and finger-pad elements. It has been shown that these simplifications do not suffice to create a realistic model

Figure 6: Left: For each superquadric in the training set we generate 1,600 grasp starting poses. These cover 16 positions over 1/8th of the total surface area, with 100 random combinations of different thumb orientations and finger spread angles. Here the long vector from each point denotes the grasp approach direction and the collection of short vectors shows the various thumb orientations. The spread angle is not shown. Middle: The optimal grasp of the SVM regression function for a superquadric shape previously seen in the training set ($\epsilon_1 = 1, \epsilon_2 = 0.3$). The quality of this grasp as tested in GraspIt! is 0.402. Right: The optimal grasp of the SVM regression function for novel a superquadric ($\epsilon_1 = 0.6, \epsilon_2 = 0.5$) The quality of this grasp as tested in GraspIt! is 0.315.

of the thumb because they do not replicate the complex functional interactions among bones, ligaments and muscles [30]. If we truly desire to create hand-like functions in a robot, we need to learn from a working system such as the human hand.

Our current efforts are focused on constructing a biomechanically realistic human hand model that would allow us to determine what features of the human hand are the most important to be mimicked when designing a robotic hand. These beneficial features will be identified by creating several versions of the human hand model, each with different sets of features, and analyzing the ability of each hand to perform a set of grasping and manipulation tasks. These iterative refinements begin with developing a hand model, with rigid bodies modeling the geometry of the digits, that has a simplified version of the actual human kinematics. This will then be compared to a version of the hand that has links that deform in a manner similar to the fleshy portions of the human hand to evaluate how compliant surfaces aid stable grasping. Another version of the hand will have realistic human joints to determine the benefits of a compliant kinematic structure, and a fourth version will incorporate the network of tendons to determine what are the advantages, if any, of indirect actuation of the joints. Our simulation system will provide an arena for these comparisons, because of its ability to simulate dynamic grasping tasks and evaluate the completed grasps with a variety of quality metrics.

As a first step, we have created a human hand model that uses accurate kinematic parameters for the joint axes of the thumb. A high quality polygonal model of the skin's surface was obtained from the program Poser, which is used for animating 3D characters. For each of the fingers we use three parallel flexion-extension (FE) joint axes and one abduction-adduction (AA) joint axis that intersects the first FE axis at a right angle. Initially, we used similar intersecting axes for the joints of the thumb (three FE axes and two orthogonal AA axes). However, while this simple orthogonal model has been considered in the past, such a model does not accurately predict maximal static thumbtip forces [30].

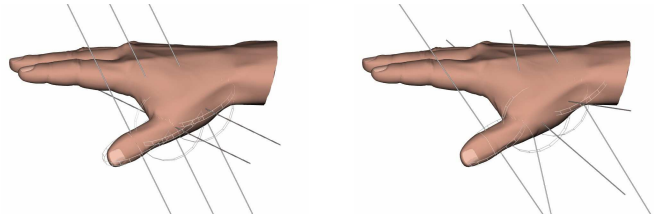

Figure 7: Left: The five joint axes of the thumb are arranged in a way commonly seen in mechanical designs with orthogonal and intersecting pairs of axes. Right: When using ideal revolute joints to model the kinematics of the hand, this set of skewed axes more accurately predicted experimental thumbtip forces and muscle coordination patterns.

Through the use of Markov Chain Monte Carlo simulations [8] and the virtual five-link thumb model [9], we have arrived at a set of kinematic parameters that can best fit, in a least squares sense, experimental thumbtip forces and electromyography data [27]. The difference in these two sets of joint axes can be seen in figure 7.

Since these models rely on ideal revolute joints, they will not be able to truly mimic all of the possible motions of the digits. This limitation can be seen in figure 8, where the hand grasps a mug. Initially the side of the thumb is in contact with the mug, but in a real grasp, the thumb would twist passively as pressure is increased at the contact. The current kinematic model prevents this from happening in simulation. What is needed is a model to predict the kinematics of a joint as a function of the shape of the bone articular surfaces in contact, the geometry and material properties of the ligaments and the applied load.

In addition, as the pressure increases at a contact, the finger surface deforms and begins to approach the local shape of the grasped object at the contact point. This results in an expanded contact area that reduces shear stresses for a given fingertip force and or torque magnitude, improves tactile information about the contact, and increases the safety margin to microslips, thus improving the stability of the grasp. Currently in *GraspIt!* all of the links are

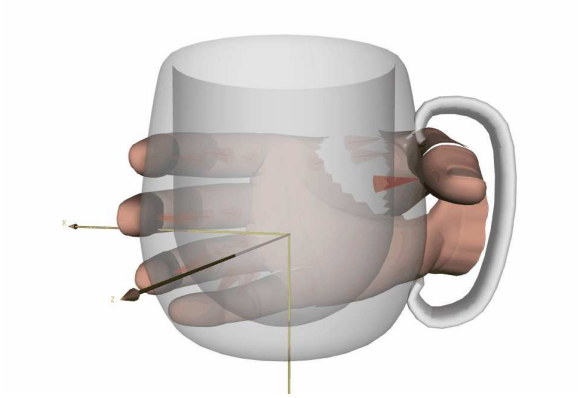

Figure 8: The thumb contacts the mug on its ulnar side, but because of the constraints of using idealized joints, it will not twist and make contact with the thumbtip pad as pressure is increased at the contact.

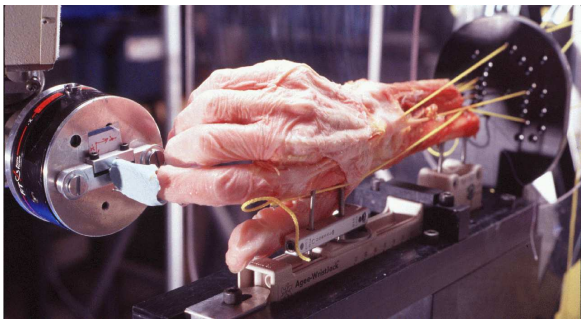

Figure 9: A computer-controlled system to deliver known tensions and displacements to cadaveric hands [22, 31].

considered individual rigid bodies; however, we are beginning to develop a physically-based deformable finger model to predict the geometric change in the contact regions and accurately compute contact forces between non-rigid bodies during grasping. We are investigating the use of adaptive refinement finite element methods [10] to compute the skin deformations in both contact and joint movement situations, and we plan to use an accurate model of the skeletal structure to constrain the positions of the innermost elements, which contact the bone. CT and MRI scans currently being collected will provide the necessary geometric detail to model the bones and soft tissue of the hand. We will also use published experimentally derived models of viscoelastic properties of the finger pads to accurately simulate these deformations [21, 28].

Finally, we are interested in modeling the indirect actuation of the tendon networks within each finger. Using an existing computer-controlled system for actuating multiple muscles in cadaveric hands (see figure 9), we can measure finger forces and torques or movement for given input tendon forces or excursions. Then with this data we will use model-based estimation techniques [2] to determine the statistically best suited representation of the finger extensor mechanism (see figure 10). This will involve estimating the number and location of the connections (i.e. points of bifurcation or confluence) among the tendons.

We have used an idealized extensor mechanism description dating from the 17th century (see figure 11a) and have simulated its behavior as a 3D floating net. The distribution of tension among the different branches depends on their geometric configuration [32] which is known to change with finger posture [7](see figure 11b). In contrast, previous descriptions of the extensor mechanism assumed a fixed distribution of tension among its elements. More recently, other studies have used this floating net approach to study finger movement [26] and thumb force production [30]. However, these models of the extensor mechanisms are not entirely validated for all finger postures (as would be desirable for a general-purpose model of the hand), nor is there a mechanics-based description of the extensor mechanism that can be efficiently encoded to be part of a computer simulator of the human hand.

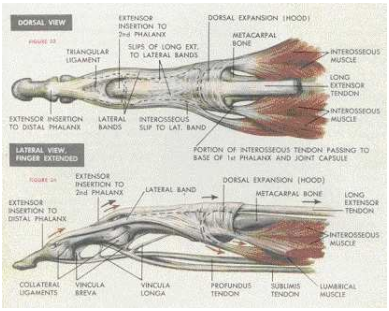

Figure 10: Stylized anatomy of the middle finger showing the tendinous network that makes up the finger extensor mechanism[20].

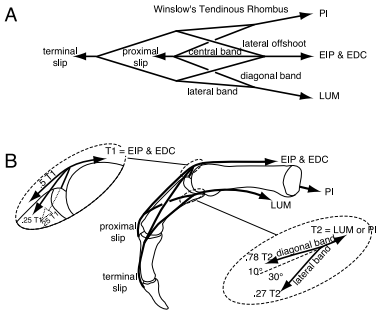

Figure 11: Dr. Valero-Cuevas' model of the extensor mechanism of the index finger [32].

4 Conclusion

We have presented a versatile simulation system that can aid grasping research by facilitating the testing of grasp planning and grasping control algorithms without the need for costly robotic hands. Of course, if one has a manipulation system, this software can be used effectively as a planning environment to devise a grasping strategy for that hand [12]. Synthesizing appropriate grasps for a given task is not a trivial problem, however, given the large number of possibilities, and we have demonstrated two different grasp planning systems that are inspired by the way humans approach the problem. The first attempts to limit this search space by using taxonomies of prehensile grasping postures and rules on how to choose appropriate postures for given primitive shapes within the object model, and the second

attempts to learn the relationship between grasp quality and the combination of object shape and grasping parameters.

GraspIt! can also be used to evaluate new robotic hand designs. Simulation is often used in the design of many other mechanical devices because it allows a designer to investigate how changes to various parameters affect the performance of the device without the need to build time-consuming and costly prototypes. Many current robotic hand designs attempt to emulate the human hand because of its proven manipulation capabilities [33]. Still however, we do not fully understand all the intricacies of this complex biological mechanism, nor do we have a clear sense of which elements are most instrumental in making the hand such an effective manipulator. Thus we have embarked on a project to create a simulated biomechanical model of the human hand that we hope will give us insight into how to build a better robotic hand.

Acknowledgment: This material is based upon work supported by the National Science Foundation under Grant No. 0312271 (ITR project) from NSF's Robotics and Computer Vision Program, NSF Grant No. 0237258 (CAREER award) from NSF's Biomedical Engineering/Research to Aid Persons with Disabilities Program (to F. J. Valero-Cuevas), a Biomedical Engineering Research Grant from the Whitaker Foundation (to F. J. Valero-Cuevas), and an NSF Graduate Research Fellowship (to V. J. Santos).

References

[1] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.

[2] Y. Bar-Shalom, X. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. John Wiley and Sons, Inc., New York, 2001.

[3] C. Borst, M. Fischer, and G. Hirzinger. A fast and robust grasp planner for arbitrary 3D objects. In *Proc. of the 1999 IEEE Intl. Conf. on Robotics and Automation*, pages 1890–1896, 1999.

[4] P. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, Mar. 1996.

[5] R. W. Cottle, J. S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[6] D. Ding, Y.-H. Liu, and S. Wang. Computing 3-D optimal form-closure grasps. In *Proc. of the 2000 IEEE Intl. Conf. on Robotics and Automation*, pages 3573–3578, 2000.

[7] M. Garcia-Elias, K. N. An, L. Berglund, R. L. Linscheid, W. P. Cooney, and E. Y. Chao. Extensor mechanism of the fingers: I. a quantitative geometric study. *J Hand Surgery (American)*, 16:1130–1140, 1991a.

[8] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markove Chain Monte Carlo in Practice*. Chapman & Hall/CRC, Boca Raton, FL, 1996.

[9] D. J. Giurintano, A. M. Hollister, W. L. Buford, D. E. Thompson, and L. M. Myers. A virtual five-link model of the thumb. *Medical Engineering and Physics*, 17(4):297–303, June 1995.

[10] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: a simple framework for adaptive simulation. In *Proc. of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290. ACM Press, 2002.

[11] R. D. Hester, M. Cetin, C. Kapoor, and D. Tesar. A criteria-based approach to grasp synthesis. In *Proc. of the 1999 IEEE Intl. Conf. on Robotics and Automation*, pages 1255–1260, 1999.

[12] D. Kragić, A. Miller, and P. Allen. Real-time tracking meets online grasp planning. In *Proc. of the 2001 IEEE Intl. Conf. on Robotics and Automation*, pages 2460–2465, 2001.

[13] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Dept. of Computer Science, University of North Carolina, Chapel Hill, 1999.

[14] A. Miller and P. Allen. Examples of 3D grasp quality computations. In *Proc. of the 1999 IEEE Intl. Conf. on Robotics and Automation*, pages 1240–1246, 1999.

[15] A. Miller and P. Allen. GraspIt!: A versatile simulator for grasping analysis. In *Proc. of the ASME Dynamic Systems and Control Division*, volume 2, pages 1251–1258, 2000.

[16] A. Miller and H. Christensen. Implementation of multi-rigid-body dynamics within a robotic grasping simulator. In *Proc. of the 2003 IEEE Intl. Conf. on Robotics and Automation*, pages 2262–2268, 2003.

[17] A. Miller, S. Knoop, H. Christensen, and P. Allen. Automatic grasp planning using shape primitives. In *Proc. of the 2003 IEEE Intl. Conf. on Robotics and Automation*, pages 1824–1829, 2003.

[18] B. Mirtich and J. Canny. Easily computable optimum grasps in 2-D and 3-D. In *Proc. of the 1994 IEEE Intl. Conf. on Robotics and Automation*, pages 739–747, 1994.

[19] J. Napier. The prehensile movements of the human hand. *J Bone and Joint Surgery*, 38B(4):902–913, Nov 1956.

[20] F. H. Netter. *Atlas of human anatomy*. Icon Learning Systems, 2 edition, 1997.

[21] D. Pawluk and R. Howe. Dynamic lumped element response of the human fingerpad. *J Biomech Engineering*, 121(2):178–183, Apr 1999.

[22] J. L. Pearlman, S. S. Roach, and F. J. Valero-Cuevas. The fundamental thumb-tip force vectors produced by the muscles of the thumb. *J Orthopaedic Research*, 22(2):306–312, 2004.

[23] R. Pelossof, A. Miller, P. Allen, and T. Jebara. An SVM learning approach to robotic grasping. In *Proc. of the 2004 IEEE Intl. Conf on Robotics and Automation*, pages 3215–3218, 2004.

[24] J. Ponce, S. Sullivan, J.-D. Boissonnat, and J.-P. Merlet. On characterizing and computing three- and four-finger force-closure grasps of polyhedral objects. In *Proc. of the 1993 IEEE Intl. Conf. on Robotics and Automation*, pages 821–827, 1993.

[25] D. Ruspini and O. Khatib. Collision/contact models for the dynamic simulation and haptic interaction. In *Proc. of the Ninth Intl. Symposium of Robotics Research, ISRR'99*, pages 185–194, 1999.

[26] J. L. Sancho-Bru, A. Perez-Gonzalez, M. Vergara-Monedero, and D. Giurintano. A 3-d dynamic model of human finger for studying free movements. *J Biomech*, 34:1491–1500, 2001.

[27] V. Santos and F. Valero-Cuevas. Investigating the interaction between variability in both musculoskeletal structure and muscle coordination for maximal voluntary static thumb forces. In *Proc. of the Neural Control of Movement Annual Meeting*, pages 27–28, 2004.

[28] E. R. Serina, E. Mockensturm, C. D. Mote, and D. Rempel. A structural model of the forced compression of the fingertip pulp. *J Biomech*, Jul;31(7):639–46, 1998.

[29] A. Speck and H. Klaeren. RoboSiM: Java 3D robot visualization. In *IECON '99 Proceedings*, pages 821–826, 1999.

[30] F. Valero-Cuevas, M. E. Johanson, and J. D. Towles. Towards a realistic biomechanical model of the thumb: The choice of kinematic description is more critical than the solution method or the variability/uncertainty of musculoskeletal parameters. *J Biomech*, 36(7):1019–1030, 2003.

[31] F. J. Valero-Cuevas, J. D. Towles, and V. R. Hentz. Quantification of fingertip force reduction in the forefinger following simulated paralysis of extensor and intrinsic muscles. *J Biomech*, 33:1601–1609, 2000.

[32] F. J. Valero-Cuevas, F. E. Zajac, and C. G. Burgar. Large index-fingertip forces are produced by subject-independent patterns of muscle excitation. *J Biomech*, 31:693–703, 1998.

[33] D. D. Wilkinson, M. V. Weghe, and Y. Matsuoka. An extensor mechanism for an anatomical robotic hand. In *Proc. of the 2003 IEEE Intl. Conf. on Robotics and Automation*, pages 238–243, 2003.