Research in robotic grasping has flourished in the last 25 years. A recent survey by Bicchi [1] covered over 140 papers, and many more than that have been published. Stemming from our desire to implement some of the work in grasp analysis for particular hand designs, we created an interactive grasping simulator that can import a wide variety of hand and object models and can evaluate the grasps formed by these hands. This system, dubbed "GraspIt!," has since expanded in scope to the point where we feel it could serve as a useful tool for other researchers in the field. To that end, we are making the system publicly available (GraspIt! is available for download for a variety of platforms from http://www.cs.columbia.edu/ ~amiller/graspit).

Some of the features of this system include:

- a robot library that includes several hand models, a Puma arm, and a simplified mobile base
- a flexible robot definition that makes it possible to import new robot designs

BY ANDREW T. MILLER AND PETER K. ALLEN

1070-9932/04/$20.00©2004 IEEE

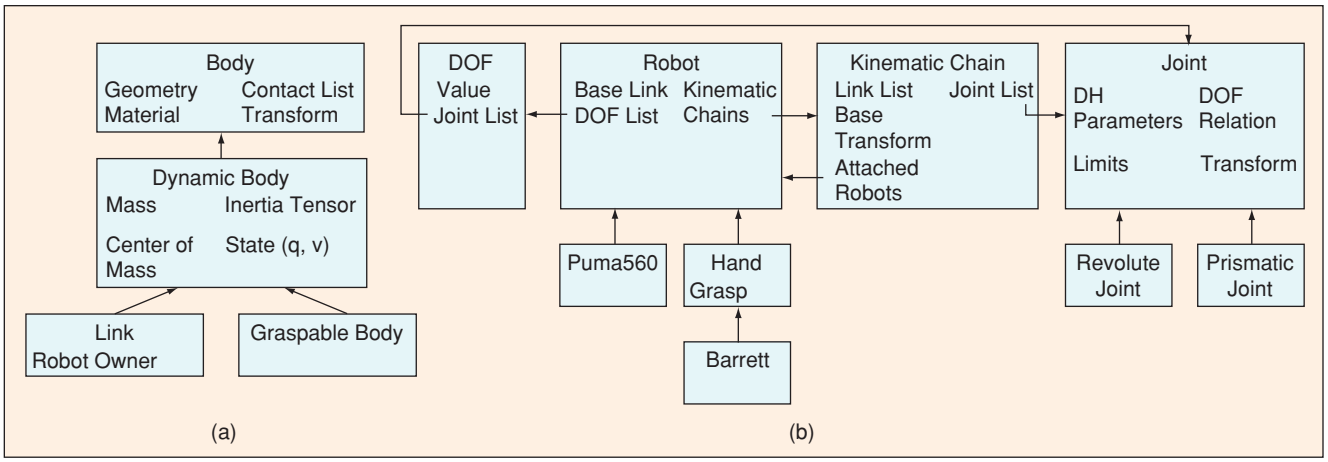

Figure 1. *(a) The body types defined within GraspIt! and their associated data. Sub-classes inherit the properties of the type above them. (b) The robot class definition and its associated data types. This definition can handle a wide variety of robots, but if a particular robot has special features or its own methods, it can be defined as a subclass of a robot or a hand.*

- the ability to connect robots to build a manipulation platform
- the ability to import obstacle models to build a complete working environment for the robots
- an intuitive interactive interface, as well as an external interface to MATLAB
- a fast collision detection and contact determination system
- grasp analysis routines that evaluate the quality of a grasp on the fly
- visualization methods that can show the weak point of a grasp and create projections of the grasp wrench space
- a dynamics engine that computes robot and object motions under the influence of external forces and contacts
- a simple trajectory generator and control algorithms that compute the joint forces necessary to follow the trajectory.

GraspIt! is an ideal environment for grasp analysis and planning, and it can serve as a test bed for new grasp evaluation, grasp synthesis, and manipulation planning algorithms. It is possible to test these algorithms much more quickly and for more hand designs than would be possible in the lab using an actual robot. Ultimately, the planning for an actual grasping task can be performed in simulation and then carried out on a physical system. The early Handey [2] worked in a similar way by combining approach, grasp, and regrasp planning for a Puma arm with a parallel jaw gripper within a simple simulated environment. The plan generated could then be executed on the real robot. In our own research, we are using GraspIt! to plan the grasps for a service robot equipped with a Barrett hand, and initial results [3] showed that by using a real-time vision system to reconcile the virtual world with the physical world, it is possible to plan and execute grasps with the real robot.

Designers of robotic hands could also benefit from such a simulation and analysis system. Currently, designing a robotic hand is a difficult task with many considerations, such as task requirements, mechanism complexity, and physical size and weight. Often, a physical prototype is necessary to truly test a hand's ability to perform tasks, but this can be quite costly and design changes are not easy to make. By using a simulation system, the designer can quickly see how different choices of kinematic parameters and link geometries affect the hand's ability to grasp different objects.

Of course, there are already 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 GRASP (an early robot arm simulator) [4], IRODESS [5], the Robotics Toolbox for MATLAB [6], RoboSiM [7], and Simpact [8], but none of these focus on the grasping problem. Chan and Liu presented a dynamic simulation system for a five-fingered robot hand [9], but they focus on dynamic manipulation and only allow fingertip contacts.

The following presents the various components of GraspIt! We begin by discussing the different types of world elements and our general robot definition, and we also present the robot library. Next we briefly describe the user interface of GraspIt! and present the collision detection and contact determination system. Subsequently, we present the grasp analysis and visualization methods that allow a user to evaluate a grasp and compute optimal grasping forces. We provide a brief overview of our dynamic simulation system, and finally we present our plans for future research.

World Elements

The simulation world in GraspIt! is made up of world elements that are either rigid bodies or robots, which impose constraints on how some of these bodies can move with respect to others. The following describes the different types of bodies defined within the system, and describes our flexible definition for robots. This definition has allowed us to model a number of complex articulated robots, which are a part of our robot library, and allows the construction of robotic platforms made up of multiple robots.

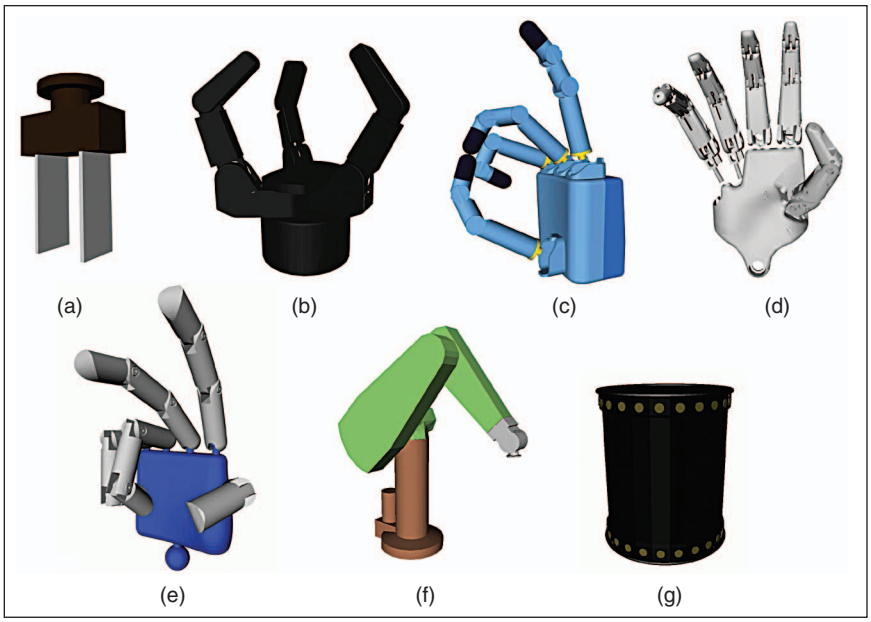

Figure 2. *Robot models: (a) a parallel jaw gripper, (b) the Barrett hand, (c) a DLR hand, (d) thke Robonaut hand, (e) the Rutgers hand, (f) the Puma 560 arm, and (g) the Nomadics XR4000 mobile robot.*

Body Types

Figure 1(a) summarizes the different types of bodies defined within the system. 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 defined as an Inventor scene graph, and it 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. Currently, a body can only have one material, but in the future it will be possible to define the surface material for different parts of a body.

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 relative to the body frame, and its inertia tensor. It also includes the body's dynamic state parameters, q and v, which specify the pose and velocity of a body frame located at the center of mass relative to the world coordinate system. The two types of dynamic bodies are links, which are elements of a robot, and graspable bodies, which are not. Graspable bodies are partially transparent and show the position of any contacts as well as any dynamic contact forces. 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. Figure 1(b) summarizes the definition of a robot. A robot consists of a base link, any number of kinematic chains, and a list of its degrees of freedom (DOF). Each kinematic chain contains a list of its links, a list of its joints, and a transform locating its base frame with respect to the robot's base frame. Each joint must be either prismatic or revolute and contains limits on its possible position as well as a Denavit-Hartenberg (DH) transform locating its frame relative to the previous joint in the list. This transform is computed from the four DH parameters associated with the joint. We define DOF separately from joints because it is common in many hand designs to have coupled joints that are passively controlled by other joints. Therefore, a single DOF has a current value and list of joints that it is connected to, and the free parameter of each of these joints can be related to the DOF value with a linear function. When a user imports a robot into the world, the system reads a configuration file specifying all of these parameters and imports each of the link bodies setting their positions based on the initial DOF values.

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. Individual types of robots can be defined as subclasses of the general robot or hand classes. These are only necessary if the robot has special features such as the breakaway feature of the Barrett hand (described in the following) or if the user wishes to specify an analytic solution for the inverse kinematics of the robot, which can be solved much faster than relying on an iterative routine.

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 computer aided design (CAD) models and kinematic descriptions of four different articulated hand designs (Figure 2). Unfortunately, we do not yet have accurate mass parameters for any of these hands, and we are currently using an approximate mass and computing the inertia tensor of each link assuming a uniform mass distribution.

A Parallel Jaw Gripper

Our gripper model has the approximate dimensions of an actual Lord gripper. For ease of use, we changed the kinematics of our gripper so that each plate can be independently controlled. When only one coupled DOF is used to grip an object in a static setting, the gripper must be centered over the object exactly for both plates to come in contact. Besides the kinematics, we also specified the surface material of the palm as metal and the plates as rubber. In the grasp example presented later, we have used a larger version of this gripper. To make this change is a simple matter of changing the joint limits in the kinematic description and modifying the palm geometry file.

The Barrett Hand

The Barrett hand, produced by Barrett Technology, is an eight-axis, three-fingered, mechanical hand with each finger having two joints. One finger is stationary and the other two can spread synchronously up to 180 degrees about the palm (finger 3 is stationary and fingers 1 and 2 rotate about the palm). Although there are eight axes, the hand is controlled by four motors. Each of the three fingers has one actuated proximal link, and a coupled distal link that moves at a fixed rate with the proximal link. A novel clutch mechanism allows the distal link to continue to move if the proximal link's motion is obstructed (referred to as breakaway). An additional motor controls the synchronous spread of the two fingers about the palm. This gives the Barrett hand only four internal degrees of freedom: one for the spread angle of the fingers, and three for the angles of the proximal links. The links are constructed of high density plastic.

The DLR Hand

The DLR hand [10], developed at the German Aerospace Center, is a four-fingered, articulated, robotic hand, and, unlike the Barrett hand, the placement of the fingers resembles the human hand structure. However, because the joint motors are all internally contained, the hand is about one and one-half times the size of an average human hand. The fingers are identical, and each consists of three links with two joints at the base, one joint between the proximal and medial links, and one joint between the medial and distal links. This last joint is coupled in a fixed ratio to the previous joint in the chain. The DLR hand has a total of 12 internal degrees of freedom, since there are three independently controllable joints in each of the four fingers. The materials of the palm and links are metal except for the fingertips, which are rubber.

The Robonaut Hand

Next, we examine the Robonaut hand [11], which was developed at NASA's Johnson Space Center. This hand has five fingers and a total of 14 internal degrees of freedom, and it is equivalent in size to a 95th percentile human male hand. The index and middle fingers along with the thumb are considered the primary manipulation fingers and are capable of abduction and adduction. The ring and pinkie are mounted on a squeezing palm joint, which makes them good grasping fingers capable of wrapping around a tool or other object. Although the link geometry only specifies the metal structure of the hand, we assumed rubber contact surfaces to facilitate stronger grasps.

The Rutgers Hand

The Rutgers Hand [12] is being developed at the Mechanical and Aerospace Engineering Department at Rutgers University. While this hand is also a five-fingered anthropomorphic design, its most novel aspect is the use of shape memory alloys to actuate the joints. Shape memory alloys are very compact and lightweight and provide a method of actuation similar to human muscle fibers, which contract when excited. Although one prototype finger has been built, the hand is still in the design phase, and the research team is using GraspIt! to examine how changes in the kinematic structure of the hand affect the hand's ability to apply the contact forces necessary for different tasks. The four fingers each have three links and the thumb has two; the base of each finger is connected to the palm with a ball and socket joint that has two independently controllable degrees of freedom, but is modeled as two revolute joints. The simulation model allows each joint to be individually controlled, even though the final model will likely have fewer degrees of freedom. This makes for a total of 19 degrees of freedom: four for each of the four fingers and three for the thumb.

The Puma560 Arm

The Unimate Puma560 arm is an industrial six-degree of freedom, spherical-wrist robot used at many research sites. We have modeled the geometry of the links using approximate dimensions, and we have used the mass parameters found in the Robotics Toolbox for MATLAB [6].

The Nomadics XR4000 Mobile Robot

The Nomadics XR4000 mobile robot has four-wheels, a holonomic drive system, and enough load capacity to support a Puma arm mounted on top of it. Again, we modeled the geometry of this robot using rough measurements. Although the wheels could be modeled as individual kinematic chains within the robot class definition, we have chosen to model the robot as having a fixed base point with two prismatic joints controlling its x and y position and a revolute joint aligned with its center z axis. In this way, we avoid having to simulate the drive system and can simply consider it as a three-degree of freedom robot.

Robot Platforms

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, each with a particular offset transform. Then, if the base link of a robot in the tree is moved, the system checks that the

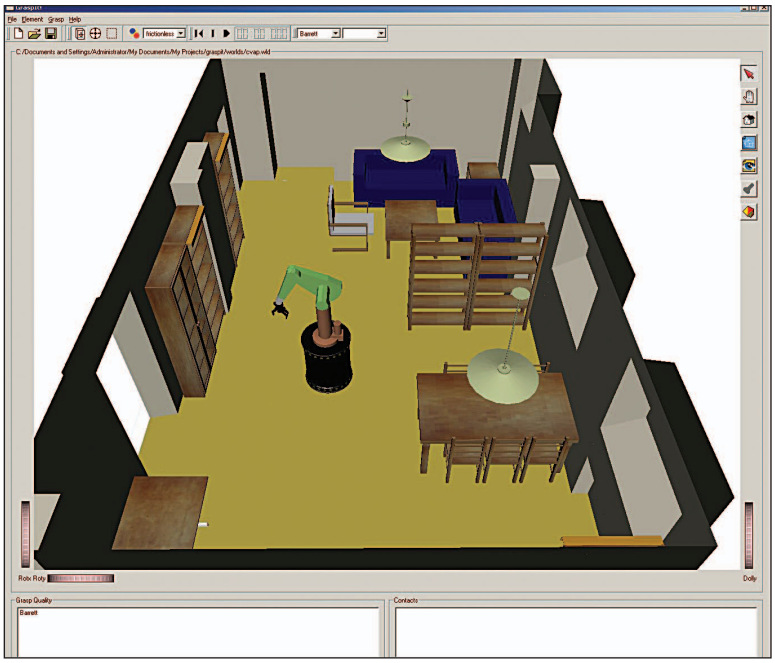

Figure 3. *A robot or robotic platform can operate within a user defined world. In this case, it is the manipulation platform and living room environment at the Center for Autonomous Systems.*

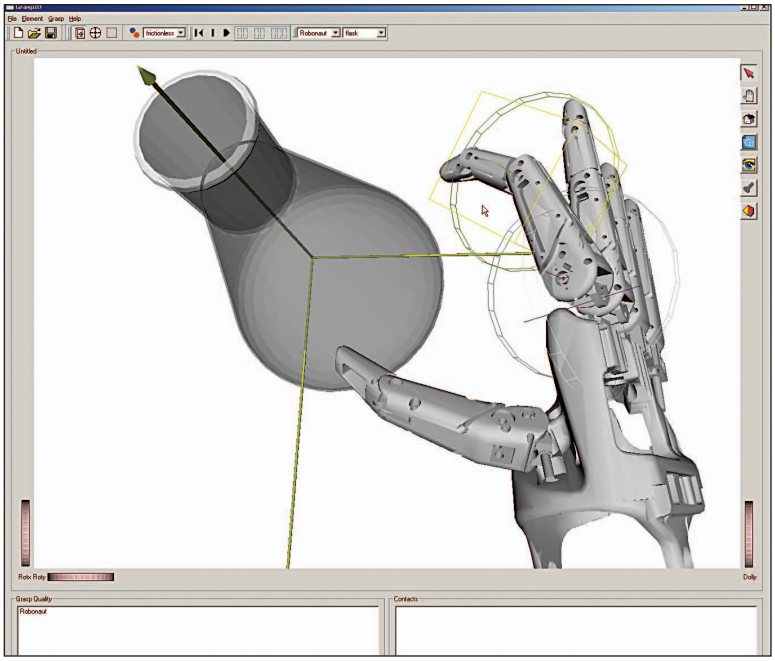

Figure 4. *The angle of a revolute joint can be changed by dragging a disc manipulator located at the joint. The passive distal joint moves in a fixed relationship with the medial joint.*

move is allowed by the inverse kinematics of the parent chain. If the move is possible, the robot is moved along with all robots below it in the tree, and the parent chain is moved to maintain the fixed attachment. This feature has allowed us to model Obelix, the mobile manipulation platform at the Center for Autonomous Systems at the Royal Institute of Technology in Stockholm, Sweden. It consists of an XR4000 mobile base, a Puma 560 arm, and a Barrett Hand. Having these additional robots allows us to test reach ability constraints for the platform. In addition, we have modeled the real obstacles within a livingroom at CAS, which serves as a test setting for this service robot (Figure 3). Together, the full platform and world model provide us with an environment where we can plan and test our entire grasping task so that we can avoid planning grasps that will conflict with these obstacles.

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 Inventor viewer, which displays, at the user's choice, a perspective or orthographic projection of a three-dimensional (3-D) world in a two-dimensional (2-D) 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.

Obstacles, graspable bodies, and robots may be translated and rotated in 3-D 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 (Figure 4), 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 file and prevent the user from moving beyond them. Passive joints also move in response to changes in the joints they are coupled with.

Various menus and dialog boxes allow the user to change a variety of simulation parameters, and these are implemented with the Qt library, which allows the code to be easily ported to different operating systems. Currently, both Linux and Windows versions of the system are available.

In addition to direct user interaction, other programs may communicate with GraspIt! using a TCP connection and a simple text protocol. We have created a preliminary Matlab interface, using mex-files that send queries and commands to GraspIt! This provides an easy way to write dynamic control algorithms because contact forces and body accelerations can be read into Matlab arrays and computed joint torques can be sent back to GraspIt! to control the motion of the robot.

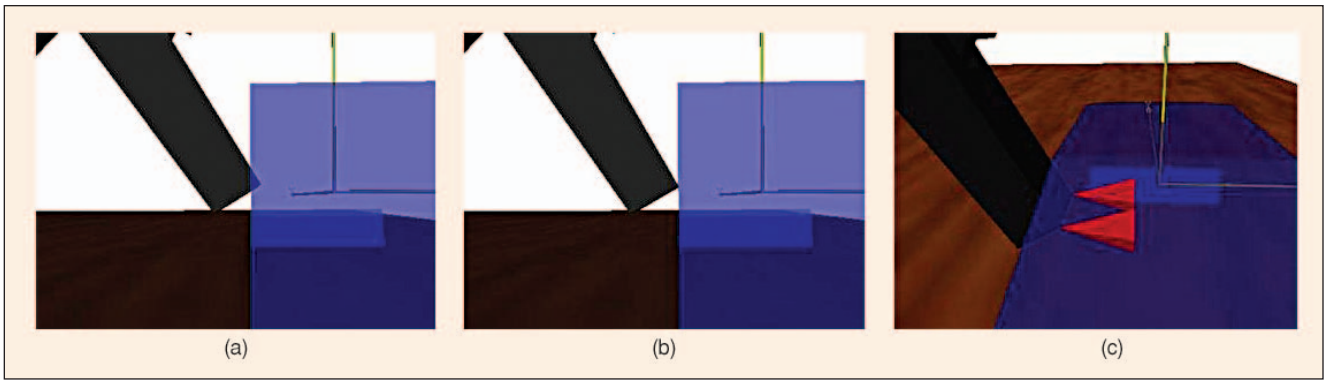

Figure 5. *The collision detection and contact location process: (a) The collision of a link of the Barrett hand with the side of the phone is detected. (b) A search is conducted to find the joint angle that will cause the link to be within 0.1 mm of the surface. (c) The geometry of the contact is determined and friction cones are placed at the vertices bounding the contact region (in this case a line).*

Contacts

Collision Detection

To prevent bodies from passing through each other while they are being manipulated by the user, the system performs real-time collision detection using a system based on the Proximity Query Package [13]. When a body is loaded into the simulator, it is faceted by the Inventor renderer, and its collection of triangles is passed to the collision detection system, where they become the leaves of a hierarchical tree of bounding volumes. At the top of the tree, the set of triangles that make up the body is bounded with an axis aligned bounding box, and it is split with a plane perpendicular to the set's longest axis. The two subgroups become the children of the root tree node, and they are each recursively bounded and subdivided until the individual triangles are reached and cannot be divided further. Fast recursive algorithms can determine if any of the triangles of one body intersect any of the triangles of another body, or can provide a minimum distance between the two bodies.

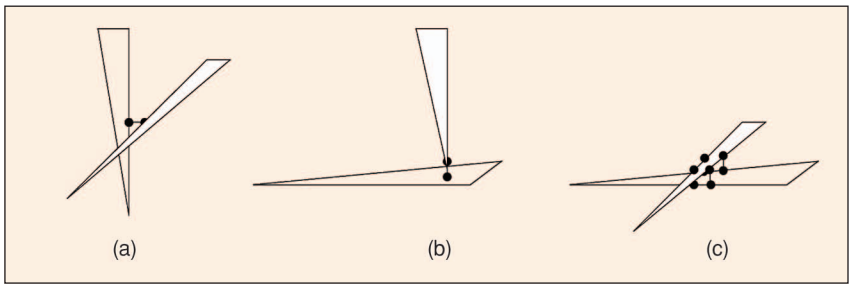

Figure 6. *Some of the possible contact regions between a pair of triangles that do not actually touch but are within the contact threshold distance of each other. (a) An edge-edge contact is located using the closest points on the two triangles. (b) A vertex-face contact is defined by the closest vertex on one triangle and its projected point on the face of the other triangle. (c) A face-face contact will have between three and six contact points (in this case four). They are found by projecting one triangle onto the other and finding the intersection points.*

If a collision is detected [Figure 5(a)], the motion of the bodies must be reversed back to the point when the contact first occurs. To find this instant, GraspIt! begins by moving the objects to their previous locations before the collision and determines the minimum distance between them. Since body transforms must be represented with fixed precision floating point numbers, it is impossible to find the instant of exact contact between two bodies. Thus, we define a thin contact region around the surface of each body, and if the distance between two bodies is less than this threshold (currently set at 0.1mm) then they are considered to be in contact. We use a binary search technique to move two bodies to within this distance after they have collided [Figure 5(b)]. When the two bodies interpenetrate, the minimum distance query returns zero rather than an interpenetration distance, which is difficult to define for nonconvex objects. It is possible to use a more efficient search algorithm, since we are provided with more than just binary information when the objects are not interpenetrating, but in practice, the binary search only needs a few iterations to reach the contact threshold distance.

Contact Determination

After the bodies have been moved to within the contact distance, the system must determine the regions of contact between the two bodies. The process begins by searching for pairs of triangles that are separated by a distance less than the contact threshold. For each pair found, the system must determine the region of overlap between them. To do this, it examines which pair of topological features produced the minimum distance between the triangles (Figure 6). The contact region for vertex-vertex, vertex-edge, and edge-edge contacts is a single point, and on each triangle, it is defined by the closest point to the other triangle. All other topological pairs include a face, and in these cases, the vertices of the other triangle are orthographically projected onto the plane containing the face. The projected triangle is then used to intersect the first triangle, and vertices of the resulting region are projected back up to the other triangle. Of those points that are projected back, only the ones closest to the face plane are kept. A vertex-face contact will only have one closest point, but a face-face contact could have a contact region

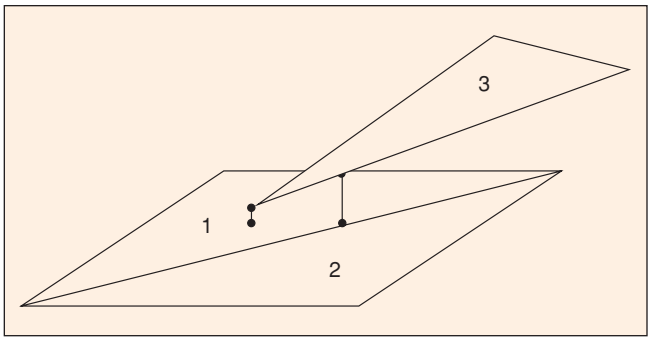

Figure 7. *Here triangles 1 and 2 lie in a plane, and triangle 3 is in a perpendicular plane. The edge-edge contact between triangles 2 and 3 cannot be matched with an edge-edge contact between triangles 1 and 3, which have a face-vertex contact, so the edge-edge contact is discarded.*

bounded by as many as six vertices, all equidistant from the face plane. Each of these vertices is considered a separate point contact and is added along with its corresponding point on the other surface to the set of contact points between the two bodies. This set grows as more triangle pairs are examined, but it keeps only those contacts that are unique (the position on the first body or the contact normal differ), since adjacent triangles on one surface will often have identical contact points with a triangle on the other surface. Unfortunately, this is not always the case, as shown in Figure 7, so we must verify that any contact involving an edge or vertex is matched by contacts on the triangles sharing the edge or vertex. If not, the contact is discarded.

After the traversal is complete, the system divides the set of contacts up into subsets that have the same contact normal. If there are more than two contact points in a subset, the system checks to see if the points are all colinear (as in an edge-face contact), and if so, it removes all contacts but the endpoints of the line. If they are not colinear (as in a face-face contact), it finds the planar convex hull of the contact set and removes any interior contact points. These interior contact points do not affect the mechanics of the grasp, because for any contact with a distribution of forces along a line or an area, the wrench applied at the contact can be represented as a single resultant wrench, and this can be specified as a convex sum of forces acting at points on the boundary of the contact region. Once the final set of contacts between the two bodies has been determined, the system draws a red friction cone at each contact point, which serves to visually mark the position of the contact and its normal [Figure 5(c)]. The width of this cone identifies how large any frictional forces can be with respect to a force applied along the contact normal. As each new contact is formed or broken, the system performs the grasp analysis routines described in the following and displays the results.

Grasp Analysis

Grasp analysis is used to assess the quality of a grasp by examining its properties. Much of the research in this field has concentrated on the placement of contacts on the grasped object, but other types of analysis are possible, including grasp force optimization, which is discussed later. Unfortunately, we cannot provide a complete introduction to grasp analysis theory, so we refer the reader to Murry et al.'s textbook [14] for relevant background.

Contacts and Friction

When two objects touch, it is possible for each of them to transmit forces through the regions of contact. Besides the forces transmitted along the contact normal, any contact can also support some amount of friction. We use the Coulomb model to determine the magnitude of forces acting in the tangent plane of the contact that can be resisted by friction. This law states: $\| \boldsymbol{f}_t \| \leq \mu \boldsymbol{f}_\perp$, where $\boldsymbol{f}_t$ is the tangential component of the contact force, μ is the coefficient of friction between the two contacting materials, and $\boldsymbol{f}_\perp$ is the normal component of the contact force. This means that the forces that may be applied at the contact lay within a cone aligned with the contact normal, commonly known as a friction cone. The half angle of this cone is $\tan^{-1}\mu$. To determine μ, we index a table of the coefficient values using the two contacting material types. Currently, we have five possible materials: glass, metal, plastic, wood, and rubber, and the user can change the μ values for any pair of materials from the preset values. In the future, we will support more complex friction models that add a relationship between torsional and tangential friction limits.

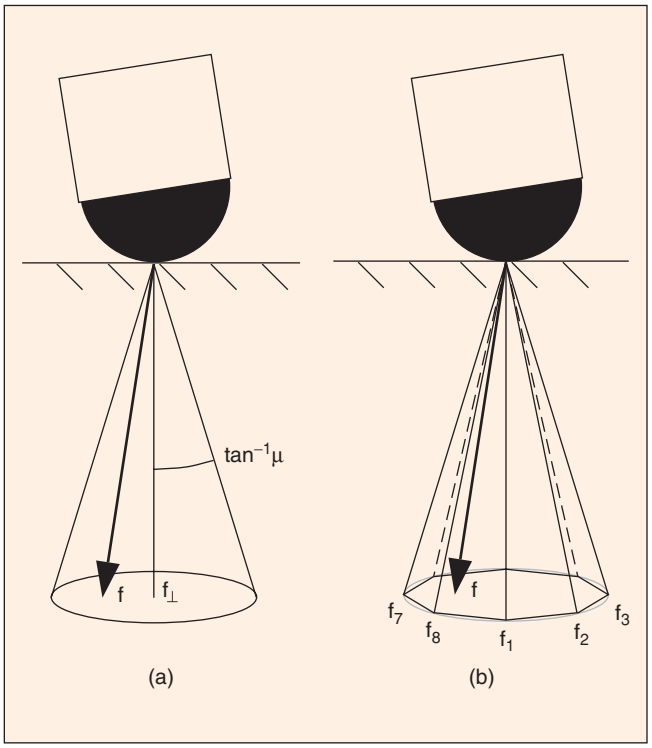

Figure 8. *(a) Frictional forces in the tangent plane have a maximum magnitude of $\mu \boldsymbol{f}_\perp$. To prevent slippage, the total contact force, f, must lie within a cone of possible directions. (b) During grasp analysis, the friction cone is approximated with an m sided pyramid. f is represented as a convex combination of m force vectors around the boundary of the cone.*

Building the Grasp Wrench Space

The space of wrenches that can be applied to an object by a grasp given limits on the contact normal forces is called the grasp wrench space (GWS). In order to construct this space, it is necessary to approximate a friction cone with a finite number m of force vectors equally spaced around the boundary of the cone (Figure 8). Initially we use $m = 8$, but the user can change this value.

Ferrari and Canny [15] describe two methods of constructing a GWS. In both cases, a convex hull bounds the space, but in one, the sum magnitude of the contact normal forces is bounded, and in the other, the maximum magnitude of the normal forces is bounded. They denote the first space as W_{L_1} and the second as W_{L_∞}. For the purposes of the quality measures described later, GraspIt! creates unit grasp wrench spaces.

Given the space of forces that can be applied at contact point i, assuming a unit contact normal force, we must determine the corresponding space of wrenches that can be exerted on the object by that contact. To accomplish this, we require a torque multiplier λ that relates units of torque to units of force. In this work, we have chosen to enforce $\|\tau\| \leq \|\boldsymbol{f}\|$ by choosing $\lambda = \frac{1}{r}$, where r is the maximum radius of the object from the torque origin, often chosen as the center of gravity of the object. This will ensure that the quality of a grasp is independent of object scale. Each of the m force vectors representing the boundary of the friction cone is translated to the wrench space origin by computing the corresponding object torque such that

$$\boldsymbol{w}_{i,j} = \begin{bmatrix} \boldsymbol{f}_{i,j} \\ \lambda(\boldsymbol{d}_i \times \boldsymbol{f}_{i,j}) \end{bmatrix} \tag{1}$$

where $\boldsymbol{f}_{i,j}$ is one the m force vectors on the boundary of the friction cone at contact i, and $\boldsymbol{d}_i$ is the vector from the torque origin to the ith point of contact. These wrenches form the boundary of the wrenches that can be applied at that contact point, given a unit normal force. To compute the W_{L_1} space, we simply take the convex hull over the union of each set of contact boundary wrenches:

$$W_{L_1} = \text{ConvexHull}\left(\bigcup_{i=1}^{n}(\boldsymbol{w}_{i,1}, \ldots, \boldsymbol{w}_{i,m})\right) \tag{2}$$

where n is the number of contacts. In the case of W_{L_∞}, we must compute the convex hull over the Minkowski sum of each set of contact boundary wrenches.

$$W_{L_\infty} = \text{ConvexHull}\left(\bigoplus_{i=1}^{n}(\boldsymbol{w}_{i,1}, \ldots, \boldsymbol{w}_{i,m})\right). \tag{3}$$

These six-dimensional (6-D), convex hull operations are performed quickly using the qhull library [20]. However, the first space is computed using only mn points, whereas the second space is computed with m^n points, so the W_{L_∞} space is only feasible for small numbers of contacts or crude approximations of the friction cone. In either case, if the wrench space origin is contained with the GWS, the grasp is considered to have force-closure, and the grasp can resist any disturbance wrench, assuming sufficiently large contact normal forces.

Quality Measures

In order to rate a grasp, we must first consider the properties of a good grasp. In general, a grasp exists to perform some task, which can be described by the space of wrenches that must be applied to an object to carry out that task. This space could be simply the wrench with a force component in the upward direction and no torque component, which would be required to suspend an object against the force of gravity, or it could be a complicated space of wrenches required in some manipulation task. This space is called the task wrench space. If the task wrench space is a subspace of the grasp wrench space, then the grasp is valid for that task. However, it is not necessarily the most efficient grasp that can accomplish the task, meaning that the internal forces on the object may be much larger than necessary. Thus, one method of defining grasp quality is the ratio of the magnitude of the task wrenches to the magnitude of the contact forces.

The most general quality measures assume nothing is known about the task wrench space. If we assume that disturbance wrenches could come from any direction, the task wrench space will be a 6-D ball. Thus, the maximum magnitude of any task wrench within this ball is equal to its radius. Since grasp quality is defined as a ratio of magnitudes, which scales linearly, we can rephrase our definition of grasp quality so that it is the radius of the largest wrench space ball, centered at the origin, that can just fit within the unit grasp wrench space. This radius, ε, will be the same as the magnitude of the shortest vector, $\boldsymbol{w}_{\min}$, from the wrench space origin to the outer boundary of the hull. We define the minimum radius of the L_1 unit grasp wrench space as ε_1.

Examining the definition of ε_1 more closely, we find a few of its intuitive characteristics. If an adversary, who knows the shape of the grasp wrench space, W_{L_1}, applies a worst-case disturbance wrench to the object, the sum magnitude of the normal contact forces would have to be $1/\varepsilon_1$ times the magnitude of the disturbance wrench. This means that the closer ε_1 is to 1, the more efficient the grasp is. The ε_1 quality measures can be computed by simply finding the smallest offset value among the hyper planes making up the facets of the convex hull.

Unfortunately, these measures are not invariant to the choice of torque origin, so the volume of the unit grasp wrench space can be used as an invariant average case quality measure for the grasp. We denote the next quality measure, v_1, as the volume of W_{L_1}. The value of this measure is reported directly by qhull when it computes the convex hull. Example grasps that were evaluated using the ε_1 and v_1, measures are presented later, and further examples can be found in [16].

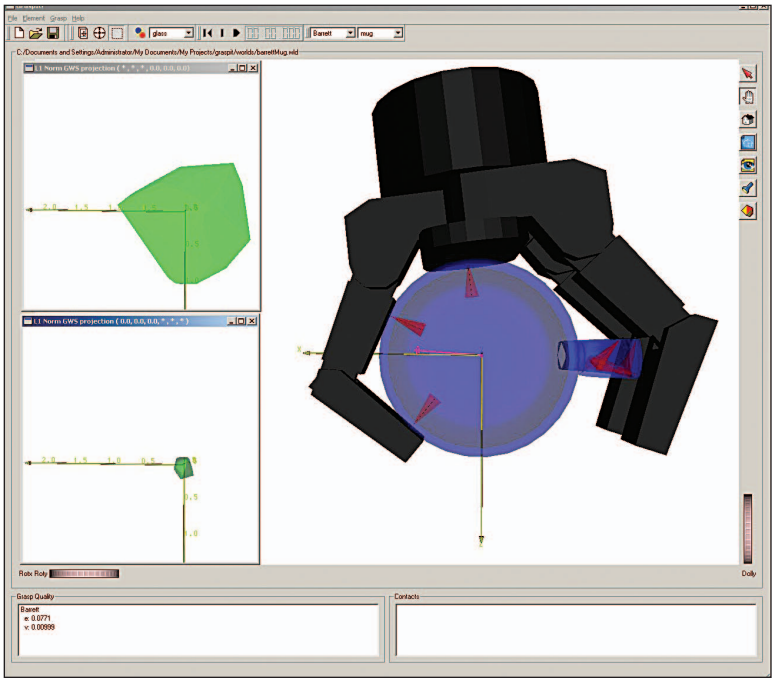

Figure 9. *A completed force-closure grasp of the mug (top view). The ε_1 and v_1 values that were computed for this grasp are displayed in the lower left portion as the values e and v. The pair of purple indicators shows the force and torque components of the worst-case disturbance wrench. In the upper left is a projection of the GWS 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.*

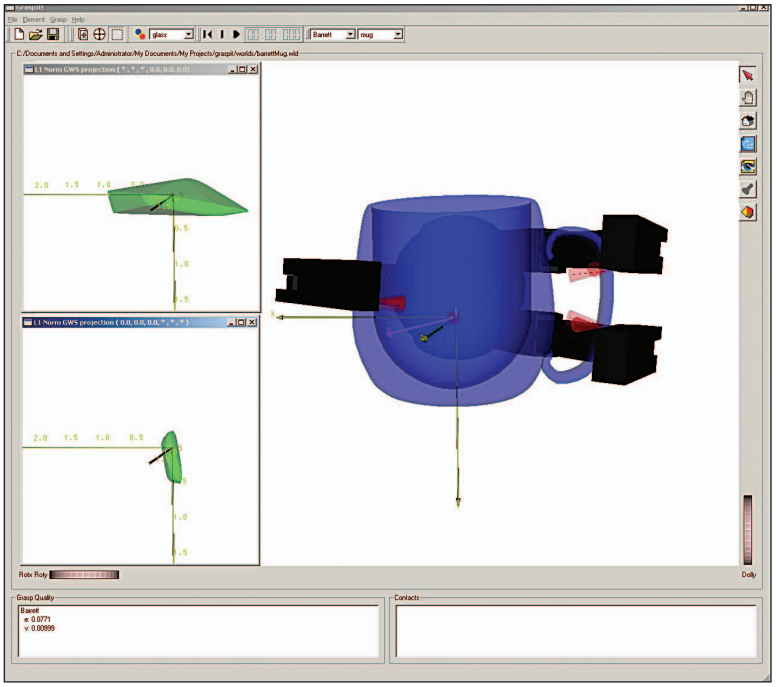

Figure 10. *A force-closure grasp of the mug (side view).*

Wrench Space Visualization

While the numerical quality measures provide some information about the efficiency of the grasp, graphical feedback can provide even more information. One reason 3-D examples are often avoided throughout the grasp-quality literature is because of the difficulty of visualizing the results. A 2-D planar object will have a 3-D wrench space, but to display the 6-D convex hull resulting from a 3-D object, we must project it into 3-D space by fixing three of the wrench coordinates. The GraspIt! system allows a user to chose which coordinates should be fixed and at what value. Although some choices for the fixed coordinates may be useful for particular types of grasps, there are two projections that are of general use for most grasps. If we fix the torque coordinates of the wrench space to 0, the resulting projection shows the space of forces that can be applied by the grasp without imparting a net torque to the object, and if we fix the force coordinates to 0, we can visualize the space of torques that can be applied to the object without a net force acting on it.

In order to construct a 3-D hull, we first project each of the 6-D hyper planes into 3-D. Then we use qhull to determine the vertices of the intersections of these half spaces, and construct the hull bodies, which are presented in the following examples. In addition to the projections, GraspIt! also shows the components of the worst-case disturbance wrench, $-\boldsymbol{w}_{\min}$, as a pair of purple indicators emanating from the object's frame of reference. These help to show where the grasp is the weakest.

In the example shown in Figures 9 and 10, the Barrett hand is closed around the outside of a mug. The purple indicators show that the grasp is weakest when a disturbance force is applied in the upward direction (with relation to the upright position of the mug), and a torque is applied along an axis between the thumb and fingers. From the wrench space projection into 3-D force space (upper projection window), we see that while the grasp can apply a variety of forces in a horizontal plane, it cannot apply large forces in the vertical direction. This is because the contacts are arranged around the outer perimeter of the mug and there are no contacts on the top or bottom of the mug. Thus, the grasp must rely on frictional forces only to resist a disturbance force in the vertical direction. We also see from the torque space projection (lower projection window) that this grasp cannot apply large moments to the mug without a net force. This is because the contact normals all point in

the general direction of the center of mass. Figure 11 shows the same grasp as before, but now the finger surface material has been changed from plastic to rubber. Since the coefficient of friction at the contacts is much higher, larger frictional forces can be supported by this grasp, and the quality of the grasp is dramatically increased. Note the size of the force and torque spaces has also increased greatly.

Comparing the Grasps of an Object Using Three Different Hands

Since GraspIt! provides objective measures of the quality of a grasp, it allows a user to compare different grasps of an object using the same or different robotic hands. It also allows the possibility of automatic grasp selection algorithms, which automatically evaluate a number of test grasps looking for the best one.

In these examples, the object to be grasped is the coffee mug, seen previously. It is fairly large, having a maximum diameter of 104 mm, and it is grasped from the side in each case. The Barrett hand is able to wrap around the mug, as shown in Figure 11, and contact occurs on two of the inner links for additional stability. The fingers of the DLR hand are longer and able to wrap further around the mug (Figure 12), and the additional finger is able to provide some vertical support underneath the mug handle. Even using the palm, the parallel jaw gripper (Figure 13) can only contact the rounded mug in three places, resulting in a lower quality grasp. As is typical with parallel jaw grips of a rounded object, the weak point of the grasp is a torque about the axis connecting the two small, opposing contact regions.

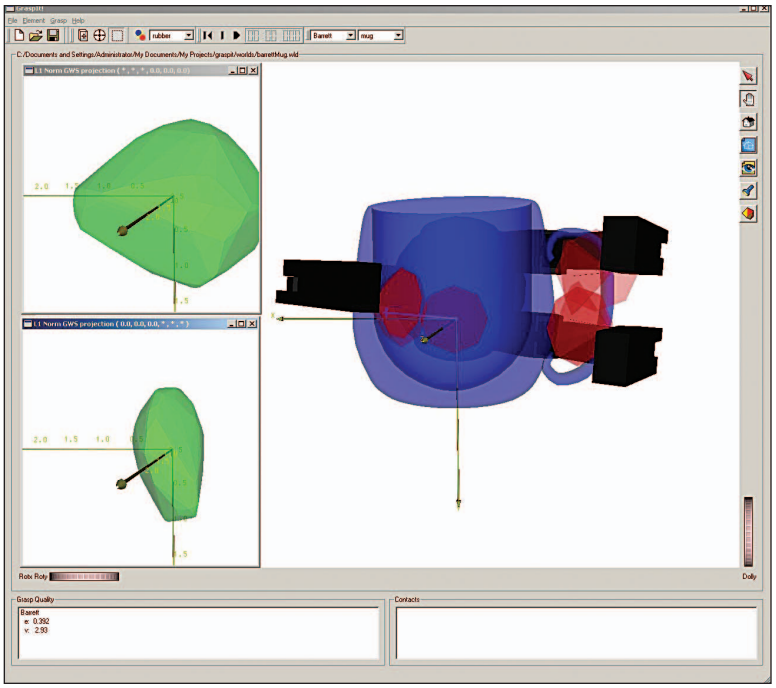

Figure 11. *A force-closure grasp of the mug using rubber finger surfaces instead of plastic.*

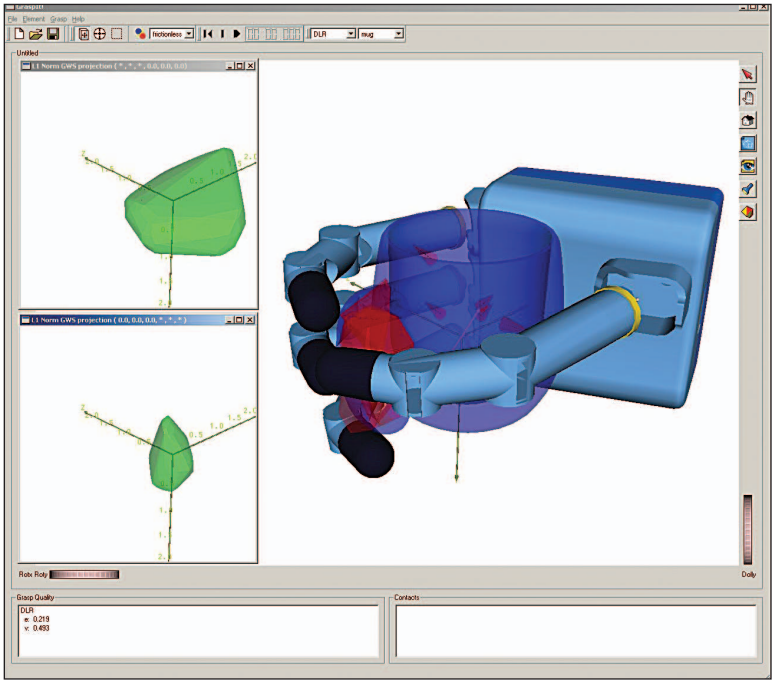

Figure 12. *A force-closure grasp of the mug using the DLR hand, which has a metal palm, inner link surfaces, and rubber fingertips.*

Dynamics

The analyses discussed previously are all performed on a static grasp. They take into account the types and positions of the contacts, but they do not tell us how to form the grasp or even whether such an acquisition is possible. In order to answer these questions, we must consider how the hand and object move over time under the influence of 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 forces acting on the body. Beyond the external forces acting on the bodies, such as gravity and motor forces, these forces must be solved for. This is done using two types of constraints on body motions: equality constraints are used to prevent bodies connected by a joint from separating, and inequality constraints are used to prevent other bodies from interpenetrating. These constraints can be formulated as part of a linear complementarity problem and solved with Lemke's algorithm, a pivoting method similar to the simplex method. The solution is the set of body velocities and impulses that satisfy the constraints (the details of our implementation and further references can be found in [17]). After each iteration of the dynamics is

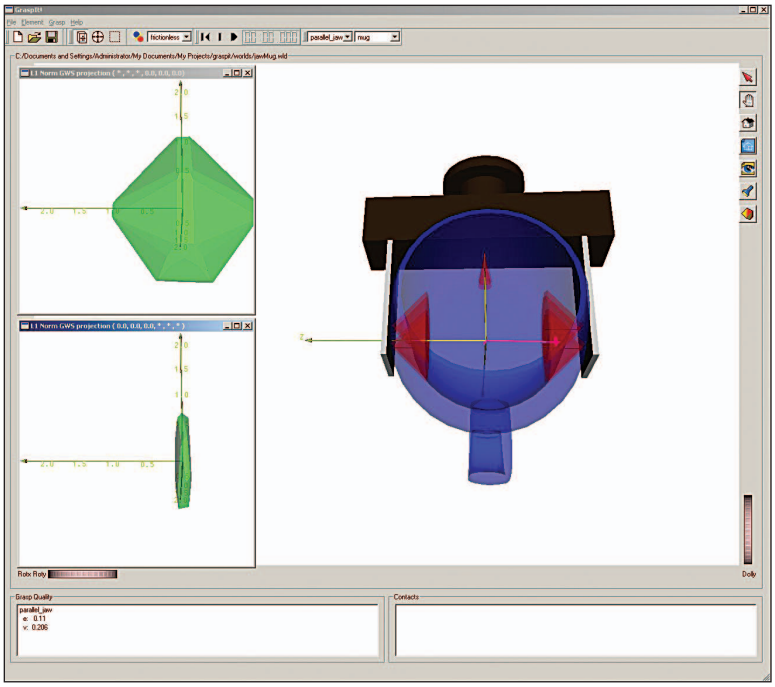

Figure 13. *A force-closure grasp of the mug using a parallel jaw gripper.*

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.

Joint Control

Without the application of external motor forces on the links of a robot, they would fall limply under the influence of gravity. So, in every iteration of the dynamics, a control routine is called for each robot. This routine can be written by the user, but we have provided some default routines. If given a new desired position for a robot, a trajectory generator creates a linear path in Cartesian space that has a continuous velocity and acceleration profile. The inverse kinematics of the parent robot are then computed for each sample along this path, and the joint trajectories are recorded. Or, if given a new set of joint positions, the trajectory generator creates the smooth joint trajectories individually.

We are currently using simple PD controllers to control the motor forces of each joint. Given a joint position set point from the trajectory, the controller for that joint determines the current error from that position and computes a joint force using gains defined within the robot configuration file. The problem with using only PD control is that the actual position never matches the desired position exactly, so in the future we plan to add a feed-forward component that would compute the inverse dynamics of the system using the recursive Newton-Euler formulation.

Simulating Grasp Formation

With the dynamics in place, it is possible to study the temporal formation of grasps. In this example, 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. Figure 14 shows the initial setup and five 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. The full movie of this experiment can be viewed at http://www.cs.columbia.edu/~amiller/graspit/movies.html.

Future Directions

We have presented a robotic grasping simulator that we believe could be a versatile tool for the grasping community. In its current state, GraspIt! is useful in a variety of applications, but there are still several features we would like to add.

- The focus of our grasp analysis has been on force-closure grasps, which are useful for pick and place type tasks. When a task involves manipulation of the object, it is necessary to plan a manipulable grasp where arbitrary velocities can be imparted to the object. By incorporating manipulability and dexterity metrics, these types of grasps could be analyzed as well.
- Currently, the force-closure quality metrics assume nothing is known about the space of external forces that might be applied to a grasped object during the execution of the task. If certain task forces were more likely to occur than others, it would be useful to provide an interface for the operator to specify this task wrench space. Then the quality metrics would more accurately reflect the quality of the grasp for the given task. One possible method, involving the use of wrench space ellipsoids, is described in [18], but in an interactive setting, there may be other possible approaches.
- The current trajectory generator is quite simplistic and does not avoid singularities or joint force limits. We would like to implement a more sophisticated trajectory generation scheme as well as a path planner that could plan an approach that avoids obstacles given a desired grasp and a starting point.
- We would like to move beyond rigid bodies and examine deformable models. Deformable fingertips is a key advantage in human grasping because it provides a larger contact surface area, and many robots are mimicking this feature by using rubber coated fingertips. To accurately compute the reaction forces will involve using finite element methods that would break up a finger link into discrete deformable mass elements connected to a rigid skeletal structure.

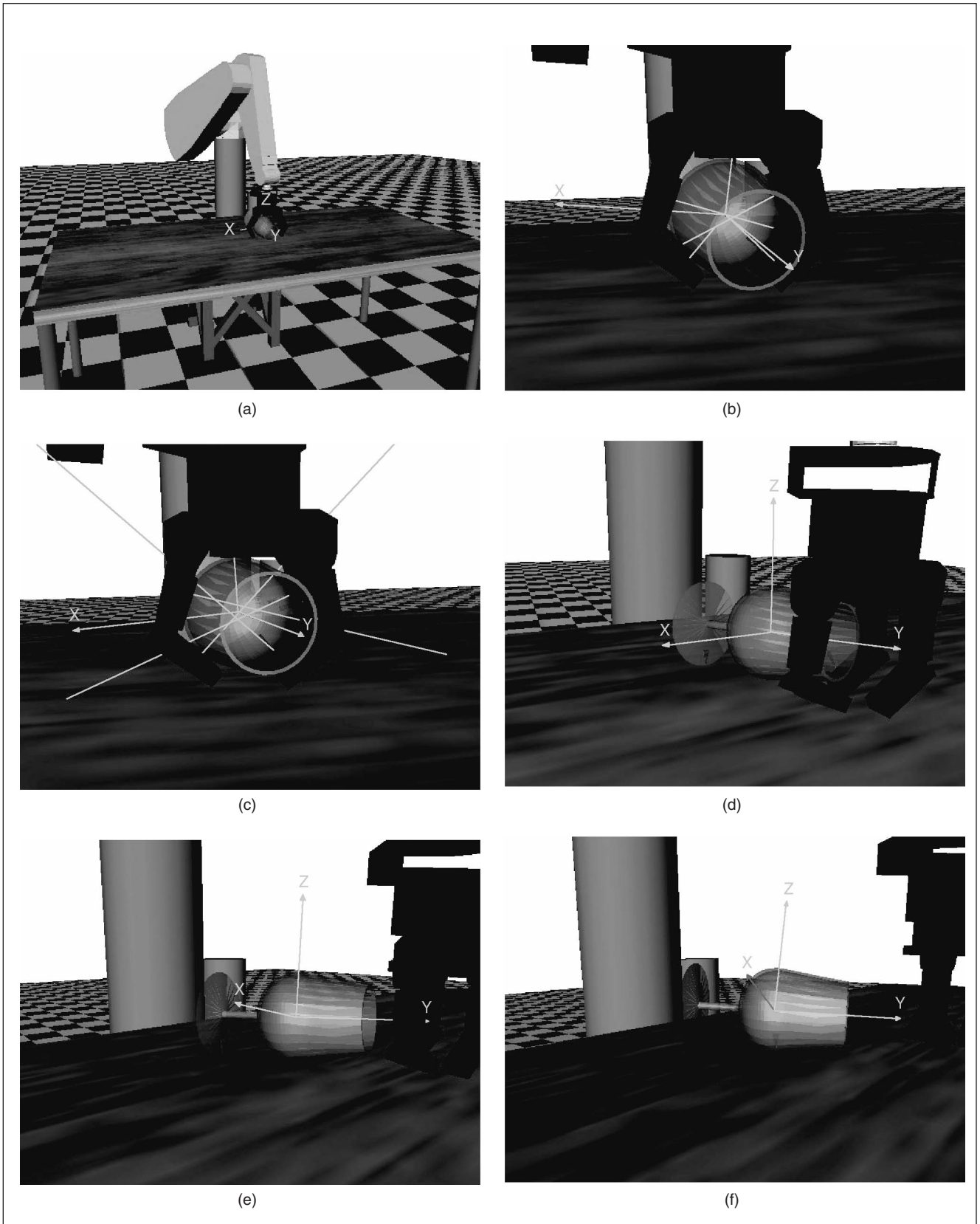

Figure 14. *The Barrett hand attempts a grasp of the wine glass, but due the low degree of friction at the contacts and the taper of the glass, the glass is squeezed out of the grasp. (a) shows the initial setup, and following from (b) to (f), snapshots are taken at 0.5103, 0.5425, 0.6148, 0.6586, 0.6956 seconds of simulation time. The full movie is available online at http://www.cs.columbia.edu/~amiller/ graspit/movies.html..*

- The last, and perhaps most exciting, area is automatic grasp selection. This is an extremely difficult problem given the large dimensionality of the search space of possible grasps, and although it has been studied extensively, little progress has been made for general grasps using complex articulated hands. We have implemented a grasp planner for the Barrett hand that uses a simplified version of an object built out of shape primitives, such as cubes, spheres, and cylinders, and uses heuristic grasp strategies defined for these shapes to generate a set of possible grasping configurations [19]. Each of these grasping configurations is tested rapidly within the simulator, and if the grasp is feasible (there is no conflict with the arm kinematics and the hand or arm is not blocked by an obstacle), it is evaluated. In this way, a few hundred grasps can be tested in under a minute and the best grasps can be presented to the user. We would like to try to generalize this system so that it could be applied to other hands as well, and we are currently examining a variety of machine learning techniques that would further optimize the grasps found with the current planner.

Acknowledgments

We gratefully acknowledge the support and guidance of Prof. Henrik Christensen at the Royal Institute of Technology. We would also like to thank Prof. Gerd Hirzinger and Dr. Max Fischer from the German Aerospace Center (DLR) for providing us with models of their robotic hand, Dr. Myron Diftler of NASA's Johnson Space Center for the Robonaut hand model, and Prof. Mavroidis from Rutgers University for his group's hand model. Finally, we would like to thank Prof. Jeffrey Trinkle for his helpful advice regarding grasp force optimization and dynamic simulation. This work was also supported in part by an ONR/DARPA MURI award (ONR N00014-95-1-0601) and an NSF ITR award (0312271).

Keywords

Grasping, grasp analysis, simulation, robot hands.

References

[1] A. Bicchi, "Hands for dextrous manipulation and robust grasping: A difficult road towards simplicity," *IEEE Trans. Robot. Automat.*, vol. 8, no. 5, pp. 560–572, 2000.

[2] T. Lozano-Pérez, J. Jones, E. Mazer, P. O'Donnell, and W. Grimson, "Handey: A robot system that recognizes, plans, and manipulates," in *Proc. 1987 IEEE Int. Conf. Robot. Automat.*, 1987, pp. 843–849.

[3] D. Kragić, A. Miller, and P. Allen, "Real-time tracking meets online grasp planning," in *Proc. 2001 IEEE Int. Conf. Robotics Automation*, 2001, pp. 2460–2465.

[4] S. Derby, "Simulating motion elements of general-purpose robot arms," *Int. J. Robot. Res.*, vol. 2, no. 1, pp. 3–12, 1983.

[5] S. Thomopoulos, Y. Papelis, and R. Tam, "IRODESS: Integrated robot design and simulation system," in *Proc. IEEE Control Systems Soc. Workshop Computer-Aided Control Systems Design*, 1989, pp. 117–122.

[6] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robot. Automat. Mag.*, vol. 3, no. 1, pp. 24–32, 1996.

[7] A. Speck and H. Klaeren, "RoboSiM: Java 3D robot visualization," in *IECON '99 Proc.*, 1999, pp. 821–826.

[8] D. Ruspini and O. Khatib, "Collision/contact models for the dynamic simulation and haptic interaction," in *Proc. 9th Int. Symp. Robotics Research ISRR'99*, pp. 185–194.

[9] J. Chan and Y. Liu, "Dynamic simulation of multi-fingered robot hands based on a unified model," *Robot. Autonomous Syst.*, vol. 32, pp. 185–201, 2000.

[10] J. Butterfass, G. Hirzinger, S. Knoch, and H. Liu, "DLR's multisensory articulated hand, part I: Hard- and software architecture," in *Proc. 1998 IEEE Int. Conf. Robotics and Automation*, 1998, pp. 2081–2086.

[11] C. Lovchik and M. Diftler, "The Robonaut hand: A dexterous robot hand for space," in *Proc. 1999 IEEE Int. Conf. Robotics Automation*, 1999, pp. 907–912.

[12] K. DeLaurentis, C. Pfeiffer, and C. Mavroidis, "Development of a shape memory alloy actuated hand," in *Proc. 7th Int. Conf. New Actuators*, 2000, pp. 281–284.

[13] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Dept. of Computer Science, UNC, Chapel Hill, NC, Tech. Rep. TR99-018, 1999.

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

[15] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. 1992 IEEE Int. Conf. Robotics and Automation*, 1992, pp. 2290–2295.

[16] A. Miller and P. Allen, "Examples of 3D grasp quality computations," in *Proc. 1999 IEEE Int. Conf. Robotics and Automation*, 1999, pp. 1240–1246.

[17] A. Miller and H. Christensen, "Implementation of multi-rigid-body dynamics within a robotic grasping simulator," in *Proc. 2003 IEEE Int. Conf. Robotics and Automation*, 2003, pp. 2262–2268.

[18] Z. Li, P. Hsu, and S. Sastry, "Grasping and coordinated manipulation by a multifingered robot hand," *Int. J. Robot. Res.*, vol. 8, no. 4, pp. 33–50, 1989.

[19] A. Miller, S. Knoop, P. Allen, and H. Christensen, "Automatic grasp planning using shape primitives," in *Proc. 2003 IEEE Int. Conf. Robotics and Automation*, 2003, pp. 1824–1829.

[20] C.B. Barber, D.B. Dobkin, and H. Huhdanpaa, "The (Q)uickhull algorithm for convex hulls," *ACM Trans. Mathematical Software*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

[21] R. Pelossof, A. Miller, P. Allen, and T. Jerba, "An SVM learning approach to robotic grasping," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3215–3218.

Andrew T. Miller is a research scientist in the Computer Science Department at Columbia University where he earned his Ph.D. in 2001. After graduating, he spent one year working with Henrik Christensen at the Royal Institute of Technology in Stockholm, Sweden. He received his B.A. in computer science from Hamilton College in 1995. His research interests include robotic simulation, grasp analysis, computational geometry, and computer vision. He is currently working on a collaborative project to develop a biomechanically realistic model of the human hand.

Peter K. Allen is a professor of computer science at Columbia University. His current research interests include 3-D modeling, real-time computer vision, robotic hand-eye coordination, and model-based sensor planning. He received the A.B. degree from Brown University in mathematics-economics, the M.S. degree in computer science from the University of Oregon, and the Ph.D. in computer science from the University of Pennsylvania.

Address for Correspondence: Andrew T. Miller, Dept. of Computer Science, Columbia University, New York, NY 10027-7003 USA. E-mail: amiller@cs.columbia.edu.