Constraint-Based Haptic Rendering of Point Data for Teleoperated Robot Grasping

Adam Leeper*
Department of Mechanical Engineering
Stanford University

Sonny Chan†
Department of Computer Science
Stanford University

Kaijen Hsiao, Matei Ciocarlie‡
Willow Garage Inc.

Kenneth Salisbury§
Depts. of Computer Science and Surgery
Stanford University

Abstract

We present an efficient 6-DOF haptic algorithm for rendering interaction forces between a rigid proxy object and a set of unordered point data. We further explore the use of haptic feedback for remotely supervised robots performing grasping tasks. The robot captures the geometry of a remote environment (as a cloud of 3D points) at run-time using a depth camera or laser scanner. An operator then uses a haptic device to position a virtual model of the robot gripper (the haptic proxy), specifying a desired grasp pose to be executed by the robot. The haptic algorithm enforces a proxy pose that is non-colliding with the observable environment, and provides both force and torque feedback to the operator. Once the operator confirms the desired gripper pose, the robot computes a collision-free arm trajectory and executes the specified grasp. We apply this method for grasping a wide range of objects, previously unseen by the robot, from highly cluttered scenes typical of human environments. Our user experiment (N=20) shows that people with no prior experience using the visualization system on which our interfaces are based are able to successfully grasp more objects with a haptic device providing force-feedback than with just a mouse.

Index Terms: I.2.9 [Robotics]: Operator Interfaces; H.5.2 [Information Interfaces and Presentation]: User Interfaces–Haptic I/O

Figure 1: An operator using a haptic device to direct a robot performing a grasping task. The algorithm presented is fully 6-DOF, although the user-study was conducted using a 3-active, 3-passive DOF device that rendered forces only.

1 Introduction

This paper presents a method for 6-DOF haptic rendering of unordered point data. We motivate this work by its application to robotic grasping; in our work a remote operator programs a desired grasp pose on a previously-unseen object using a haptic device with force feedback. In particular, we are interested in situations typical of unstructured, human environments. In such situations, a robot can be required to handle a wide range of previously unseen objects, for which the robot has no internal model. Furthermore, the operating scene can be heavily cluttered with objects to be manipulated and with other obstacles (e.g. tables, cabinets, and vases). A remotely operated robot can handle such difficult environments thanks to the scene understanding and general cognitive abilities of its operator. However, the overall efficiency of the task can potentially be increased by assisting the operator using autonomous algorithms where possible and by improving the interface between the human and robotic components of the system.

Haptic rendering can provide the operator with force feedback regarding potential contacts and collisions between a desired robotic end-effector pose and the environment, which is particularly valuable in the context of a grasping task. Imagine an operator trying to position the robotic end-effector in a scene in order to grasp a desired object. This task, by definition, requires making contact with the environment, or at least with the target object. However, it is also often desirable to avoid contact between the robot gripper and other parts of the environments, such as other objects or obstacles; these types of contacts can be communicated to the operator through haptic feedback.

In our system, the operator uses a haptic device to select the 6D position and orientation of a virtual model of the robotic gripper in a cluttered scene, as shown in Figure 1. Three-dimensional point cloud data, acquired at run-time through the robot's sensors, serves as a minimally-processed model of the world. Running our haptic rendering algorithm with this data, the operator can interact with this world model while receiving force feedback on collisions between the virtual gripper and the environment. Once the user has selected a desired grasping pose for the virtual gripper, an autonomous motion planning module attempts to compute a collision-free arm trajectory for the desired grasp, and, if one is found, the robot proceeds to execute the grasp.

This paper presents three main contributions. First, we integrate portions of the authors' previous work [7, 14] to arrive at a haptic algorithm for 6D rigid proxy interactions with unconnected and unordered point cloud data. Second, we use this algorithm as part of a novel system for remote grasp selection based on either haptic device or mouse input. Third, we study the effect on user performance of using either mouse or haptic device with force-feedback as input to the grasp selection system. We believe that combining our haptic algorithm with run-time sensor data can enable teleoperated grasping in unstructured, human-like settings. The remainder of this paper discusses each of the above points in turn.

*e-mail: aleeper@stanford.edu
†e-mail: sonny@cs.stanford.edu
‡e-mail: {hsiao, matei}@willowgarage.com
§e-mail: jks@robotics.stanford.edu

2 Related Work

Haptic interfaces for robot manipulation have been used in diverse applications such as surgical, space, undersea, and rescue robotics. Many such interfaces include an aspect of shared control. These include shared execution of motion trajectories with moderate time delays (as encountered in space) [11] [16], virtual fixtures [3], and other methods for constraining motions of the master [17]. For a thorough review of the related literature, see Sheridan [21].

The interfaces for grasping used in this paper allow the operator to specify a grasp goal, using both a camera view of the scene and a virtual, 3D rendering of a stereo point cloud of the scene; the grasp is then executed autonomously. Other interfaces have similarly provided virtual renderings of the scene and allowed the user to specify manipulation goals that are then executed autonomously [12] [4]. However, all of these interfaces require full models of the scene and the objects to be manipulated, which are not available in most real-world tasks, including ours.

With the advent of depth imaging technology, modern teleoperation systems are able to supply the operator with a stream of RGB-D image data. Commodity RGB-D cameras such as the Microsoft Kinect™ have made this kind of data more accessible than ever. However, haptic interaction methods and computation of force feedback on RGB-D data largely remain to be explored.

Cha *et al.* pioneered some of the work on haptic interaction with depth video media in [6] and later improved their method in [5]. They tessellated the organized grid of depth values in 3D, then haptically rendered the resulting terrain mesh using an adaptation of the proxy graph algorithm [22].

Interpreting the RGB-D image data as an *unorganized* point set in 3D space can have advantages over explicit tessellation, and is often standard practice. Several authors, including Lee & Kim [13], El-Far *et al.* [8], and Rydén *et al.* [20], have proposed haptic rendering methods based on this interpretation. These approaches generally rely on rendering a penalty force computed from a neighborhood of points near the haptic interaction position. Recently, Leeper *et al.* described a three degree-of-freedom (3-DOF) algorithm for constraint-based haptic rendering of arbitrary point data [14]. They used an implicit surface rendering approach and explored two different formulations of an implicit representation from the point set: *metaballs* [23] and *surfels* [1].

A grasping task such as the one described in this paper is inherently a 6-DOF task that involves controlling both the position and orientation of a robotic gripper in an environment with obstacles. Although haptic interaction with such a scene would require a 6-DOF rendering algorithm, to date, no 6-DOF haptic rendering method for RGB-D sensor data has been demonstrated.

Popular methods for 6-DOF haptic rendering of rigid-body interactions include the Voxmap PointShell™algorithm by McNeely *et al.* [15] for a voxel representation of the scene geometry, and the methods of Gregory *et al.* [9], Hasegawa & Sato [10], and Otaduy & Lin [19] for polygonal mesh representations. In a grasping task, we are nominally interested in configurations of the end-effector that are free from interpenetrations with objects in the environment. Thus, constraint-based rendering algorithms that enforce non-penetration between objects are perhaps more suitable for this type of application than those which compute reaction forces from object penetration depth. Ortega *et al.* described a 6-DOF version of the god-object algorithm that simulated the motion of a virtual proxy object under contact constraints [18]. Later, Chan *et al.* presented an algorithm for rendering isosurfaces implicitly defined within sampled volume data that uses similar principles [7]. The ideas presented in these latter works, along with [14], form the basis for the haptic rendering methods we describe in this work.

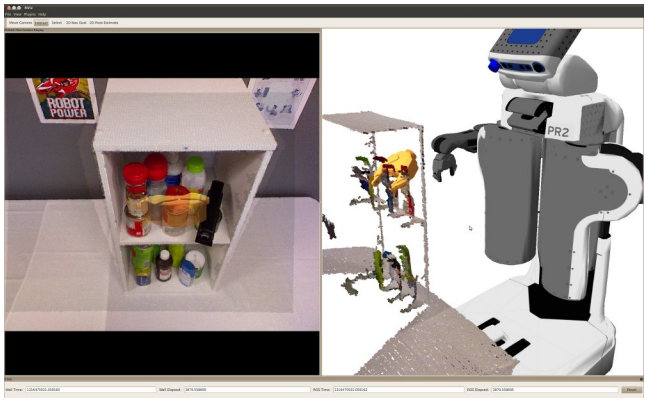


Figure 2: The Graphical User Interface used for the teleoperation tasks in this study. On the left is a real-time feed from the Kinect camera; on the right is a user-controllable, rendered view of the robot and the point cloud, which is incomplete due to occlusions.

3 Interfaces for Grasping

The purpose of our system is to allow a physical robot to perform a number of grasping tasks of common household objects in a complex environment, by allowing the user to specify desired grasp poses for the robot to execute autonomously. The system is designed for remote operation; the operator controls the robot through a separate desktop computer without directly viewing the robot.

3.1 System Overview

The hardware we used was the PR2 personal robot, shown in Figure 6. The PR2 has two backdriveable 7-DOF arms with parallel-jaw grippers. For range sensing, we used two sensors: a Kinect™ mounted on the robot's head (used as the main sensor, providing both range and color images), and a tilting laser rangefinder mounted on its chest (used only for autonomous collision avoidance). During the study, the PR2 communicated with the computer running the teleoperation interface via a commodity wireless network. We chose this setup as we expect that any mobile robot in real households or offices will have to be untethered in order to perform useful tasks.

We developed a Graphical User Interface (GUI) implemented using `rviz`, a 3D robot visualization and interaction environment in ROS (www.ros.org/wiki/rviz). A screenshot of our interface is shown in Figure 2. It presents the user with two main displays: on the left, a real-time feed from the Kinect camera mounted on the PR2; on the right, a rendered image of the PR2 in its current posture, along with a 3D point cloud showing a snapshot of the world as seen by the Kinect. The user can point the robot's camera by left-clicking anywhere in the camera view, changing the point of view of the live camera feed shown on the left. Since the right image is rendered, its viewpoint can be moved to any position by rotating, translating, and zooming the scene.

In order to generalize to unstructured settings, we make no initial assumptions about the contents of the scene. The only world model that we use is an unconnected point cloud, as can be acquired at run time through the robot's sensors. In this paper, we use the widely available Microsoft Kinect™ to provide this data. Using only run-time data of a scene presents an important challenge. In many cases, obstacles in the environment prevent the robot from seeing a scene from multiple viewpoints. The resulting point cloud will thus exhibit significant "blind spots", or occlusions, with the robot unable to see the back sides of objects (Fig. 3). This problem could be alleviated by attempting to recognize the objects from this incomplete data and filling in the missing parts from a database of

Figure 3: Raw 3D sensor data is, by its nature, incomplete and noisy, as seen in this side rendering of point data for objects on a shelf.

known models; however, this approach assumes that the robot has such an extensive database available and is able to recognize most objects of interest.

For the sake of applicability in general, unstructured environments, we avoid these assumptions and use partial, single-view point clouds of the scene as input to our algorithm. In our interface, we use a static snapshot instead of a continuous feed of the Kinect™ range data. This choice was made because it reduces network traffic, yet the user can refresh this static snapshot with a button click at any time.

3.2 Grasping Strategy

Our grasping strategy is based on two main concepts:

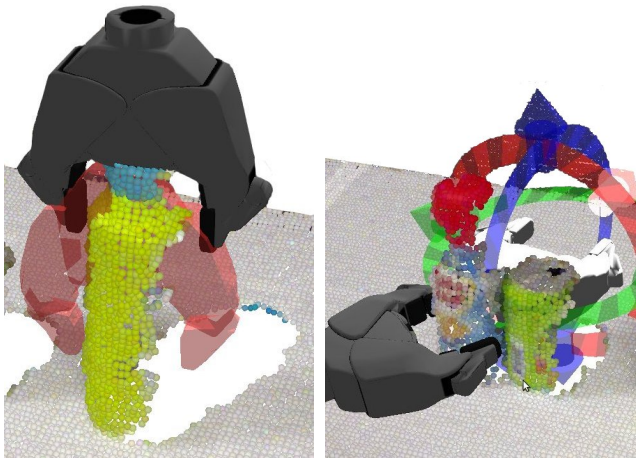
- The operator is only required to specify the pose of the gripper for grasping a desired object. An autonomous module is then in charge of computing an appropriate arm trajectory for executing the specified grasp. This removes the need for the operator to take into account the complex (and, in our case, non-anthropomorphic) kinematics of the arm, and to avoid collisions between the arm and the environment during the trajectory. The gripper itself, however, must come in contact with parts of the environment (especially the object itself), as well as avoid unwanted collisions while attempting to obtain a stable grasp of the target object. We thus attempt to focus the cognitive abilities of the operator on the gripper pose component of the task, offloading the other components to autonomous modules.
- The operator specifies a desired grasp pose using a *virtual* model of the gripper; once the operator has adjusted the pose of the virtual model he or she sends it to the robot for execution. This allows the operator to check the desired pose and perform fine adjustments before the robot starts to move.

We have implemented this strategy using two variants for the user interface, described in the following subsections.

3.3 Haptic Interface

To position the virtual gripper model in the scene, the operator uses a Phantom Omni™ haptic device, with the haptic algorithm described in the next section. As the virtual gripper always displays the pose of the collision-free haptic proxy, it is never in collision with any of the points that make up our scene model. In addition, rendering the forces output by the haptic algorithm allows the operator to "feel" the potential contacts between the hypothetical grasp denoted by the virtual gripper and the environment. This interface is illustrated in Figure 4.

In addition to the haptic device, the operator has access to a number of functions using a computer mouse, operated with the other hand. The mouse can be used to change the point of view of the



(a) Haptic Device Input (b) Mouse Input

Figure 4: The solid-gray virtual gripper (representing the pose of the collision-free proxy) is positioned for grasping. (a) The transparent red model represents the pose of the haptic device, not displayed to the user but shown here for illustration purposes. (b) The colored rings and arrows are manipulated using the mouse to set the substitute device pose. The algorithm does not know (or care) what is the input modality.

virtual camera. The operator can also re-center the workspace of the haptic device at any location in the scene by clicking on the associated point in the scene point cloud using the mouse.

Once the operator is satisfied with the virtual gripper pose, he or she simply clicks on the virtual gripper itself using either the mouse or a button on the haptic device pen. At that point, the robot attempts to compute an arm trajectory for executing the specified grasp. If one is not found, the virtual gripper turns red, informing the operator that a different pose is needed. If a trajectory is found, the virtual gripper turns blue, informing the operator that the grasp is ready to be executed. After a final confirmation from the operator, the robot executes the grasp and attempts to lift the object.

3.4 Mouse Interface

One of the goals of this study is to analyze haptic feedback in the context of teleoperated grasping tasks. To isolate the haptic component and quantify its benefits, we compare the interface described above against a variant where the haptic device is replaced with a regular computer mouse. The operator controls the substitute haptic device pose via a set of ring-and-arrow controls that can be dragged on the screen, as shown in Figure 4. The resulting pose is then used as input to the same haptic algorithm, which in turn produces a pose for the collision-free proxy and thus the virtual gripper used to specify the desired grasp.

When using this variant of the interface, the operator can see both the substitute device (which can be controlled directly through the rings and arrows) and the collision-free proxy. The operator is allowed to move the substitute device pose anywhere in the scene, with the collision-free proxy attempting to follow as in the case of the haptic interface. Once the operator releases the controls of the substitute device, the controls snap back to the pose of the collision-free proxy. The goal of this approach is to provide the operator with the same benefits of collision avoidance as in the previous case, but in the absence of the 6D input modality or force feedback.

All the mouse functions relating to movement of the virtual camera are also available, unchanged relative to the previous interface. The process of completing a grasp is also unchanged. Once the operator is satisfied with the pose of the virtual gripper, he or she

asks the robot to perform the feasibility test and, if the test passes, confirms the grasp for execution.

4 Overview of Haptic Algorithm

In the context of remote grasp planning with haptic feedback, we require an algorithm that can perform haptic rendering of potential contacts between a robotic gripper and its environment, represented as an unordered point cloud. By nature, this task requires reasoning about contacts anywhere on the gripper's surface, rather than just at a single point approximating the tooltip, as a 3-DOF rendering algorithm does. Our haptic algorithm thus uses a complete mesh model of the gripper for reasoning about collisions.

The haptic rendering algorithm presented here is based primarily on the methods described in [7] and [14]. It first reads a pose configuration (position and orientation) from the haptic interface, then simulates the motion of a collision-free proxy model that is constrained by contacts detected between the model and obstacles in the scene, and finally renders a feedback force based on the difference between the poses of the simulated proxy and the actual device. This algorithm runs in a continuous servo loop at a rate of approximately 1 kHz in a separate thread of the application.

In this section, we describe the haptic rendering components in greater detail and, in particular, the adaptations made to the algorithms described in [7] and [14] to accommodate our present task and environment. We refer the reader to [7] for a full description of the 6-DOF haptic rendering algorithm.

4.1 Data Representation

The haptic rendering algorithm simulates the interaction between a user-controlled rigid body (in the case of a grasping task, the robot's gripper) and a static scene of rigid objects constituting the virtual environment (in our case, a point cloud of the scene acquired from the robot's sensors). The algorithm requires that the virtual environment provide two pieces of information:

- It must report whether a given 3D query location is in free space or in collision with geometry in the environment.
- It must provide a surface normal at any point of contact with environment geometry, to establish a contact constraint.

A surface described by an implicit equation is in many respects an ideal representation that meets the collision testing requirements of the rendering algorithm. Two different methods of formulating an implicit surface representation from unconnected point cloud data are described in [14]. In our present work, we elected to use the metaball representation (Fig. 5) wherein each point in the captured scene adds a contribution to the global implicit field at its position. The single-view depth image data we use in this application presents difficulties for the surfel representation because of the abundance of occlusions and partially-imaged objects in the scene.

The global implicit function that describes the scene geometry is a sum of finitely-supported radial basis functions centered at each of the input cloud points. We chose the Wendland function as our basis, expressed as

$$\psi(r) = \begin{cases} \left(1-\frac{r}{R}\right)^4 \left(\frac{4r}{R}+1\right) & \text{if } r < R \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where r is the distance to the point and R is a chosen radius of support for that point. The support radius of each point was set as a multiple (in our implementation, 2.5) of the computed average spacing between points in the cloud. The full implicit equation describing the scene is thus written

$$f(\mathbf{x}) = T - \sum_{i=1}^{N} \psi_i(||\mathbf{x}-\mathbf{c}_i||), \quad (2)$$

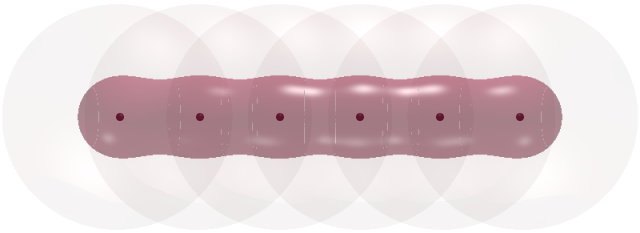


Figure 5: A simple illustration of a metaball surface. The individual points (dark red dots) are given a finitely-supported radial basis function (light overlapping spheres). The net scalar field is used to define an isosurface that is rendered as a rigid object (wavy red blob).

where $\mathbf{c}_i$ is the center of basis function ψ_i and T is a chosen threshold value. Consistent with the findings in [14], we used $T = 0.5$ in this work to ensure a smooth field. Picking a higher value of T would tend to ignore collisions with fringe points, which is undesirable in this setting since the Kinect™ sensor tends to miss edges of rounded objects while giving very few false positives. Positive values of f indicate free space, whereas negative values are in interference with the scene. The normalized gradient of the field function, $\nabla f/||\nabla f||$, serves as the surface normal for establishing contact constraints.

The geometry of the user-controlled object (the robotic gripper in our case) is represented as a point-sampled surface or point shell. A high-resolution CAD model of the gripper was down-sampled, and the vertices of the resulting mesh were taken to form the point shell. A denser point sampling was used in regions of the model deemed more important to the grasping task, such as the fingertips and finger pads. A surface of approximately 950 points represented the final gripper model.

4.2 Collision Detection

The role of the collision detector is to locate and report all instances of interference between the geometries of the user-controlled object and the scene. Using our data representation, this amounts to testing all points of the point shell against the scene geometry with one additional challenge. Because we are interested in interference-free configurations of the gripper, the collision detector must find the configuration of the gripper furthest along its movement trajectory that is free from interference, rather than simply reporting whether or not collision has occurred.

Details of the collision detection algorithm used in our implementation can be found in [7]. We used the method described exactly but for one modification: rather than sampling a value from a volumetric grid to determine interference for a point, we instead evaluate our implicit function (generated from the sensor point cloud) at that position.

During each cycle, the configuration solver determines a target pose for the collision-free gripper proxy based on the previous proxy pose, the haptic interface pose, and the currently active contact constraints. The collision detector must then determine whether or not the path between the proxy's previous pose and the new target pose has any collisions and, in the affirmative case, determine the pose furthest along the path that the proxy can reach before contact occurs. The algorithm accomplishes this by subdividing the motion path into segments such that no point on the proxy model moves more than a specified distance, f, which loosely indicates the size of the smallest feature in the virtual environment, in that segment. Knowing the radius of influence of each point cloud point, we can set f appropriately to some fraction of this radius. The collision detector tests each segment of the motion path by querying all points of the point shell against the scene for interference. Once a collision

is found, interval bisection is performed on the segment to refine the contact to within a desired error distance, ε, and the interference-free pose is reported.

An interference test for a point is performed by evaluating the implicit equation (2) at its position. Because the metaball equation used has a finite radius of support, only the small neighborhood of cloud points that have a non-zero contribution to the function at the query position need to be considered. In our implementation, we use a k-d tree constructed on the point cloud to accelerate the neighborhood search. For our point cloud with approximately 50,000 points, a typical query near the surface of a scene object will involve a neighborhood of approximately 25 points.

4.3 Configuration Solver and Virtual Coupling

The rendering algorithm tracks the pose of the haptic interface and simulates the pose of the collision-free virtual proxy; the two are attached by a 6-DOF virtual spring. The force from the virtual spring determines the motion of the virtual proxy during each frame, and the reaction force is rendered to the operator as the feedback force. The stiffness of this virtual spring can be set to control the amount of force rendered to the operator.

Every contact between the proxy and the scene introduces one degree of constraint on the proxy's 6D motion. Knowing the applied force on the proxy (from the virtual spring) and the contact constraints, Gauss' principle of least constraint can be used to solve for the motion of the proxy object. The constrained motion path of the proxy is found by solving an optimization equation as described in [18] and [7].

5 User Experiments

We performed user experiments to assess the performance and usability of our grasping strategy and to quantify how much the haptic interface vs. the mouse interface affects performance. Of 20 adult particpants, 10 had used rviz (the visualization environment on which our interface is based) before, and 10 had not. For the task, participants grasped objects from the shelf environment shown in Figure 6, chosen to simulate a typical situation that might be encountered in a household environment. Each user performed the same task with both interfaces, in a randomly-chosen order; interface types were balanced for order. Upon completion of the study, each participant was given a $20 gift card as a token of thanks.

5.1 Experimental Procedure

Participants were recruited from colleagues and local contacts. 12 participants were 20-29 years of age, 4 were 30-39, 2 were 40-49, and 2 were 50-59. The participants who had used rviz before were very familiar with robots (mean (M)=6.70, standard error (SE)=0.15; 1 = not familiar at all, 7 = very familiar), while the non-rviz participants were much less familiar (M=3.70, SE=0.68). For rviz users, 8 had 3D experience, defined as having played at least one 3D video game or used a CAD (computer-aided design) program, and 2 had only played 2D video games; 7 had used a haptic device before, while 3 had not. For non-rviz users, 6 had 3D experience, 2 had played only 2D video games, and 2 had no experience with video games or CAD programs; 3 had used a haptic device before.

After signing the study agreement form, each participant was taught to use first the mouse interface and then the haptic interface by grasping two objects placed on the same shelf used for the task. Training included how to move the robot camera, how to move the viewpoint for the virtual scene, how to drop off objects, and how to refresh point clouds. Data collection then occurred during the task, for which the participant had 10 minutes to grasp as many objects as possible from the environment shown in Figure 6, using both the mouse and haptic interfaces in turn (in the randomly-chosen order).

Figure 6: The PR2 robot preparing to execute a grasping task in our experiments. This difficult scene includes large obstacles and a highly-cluttered arrangement of objects in the shelf.

The grasping task simulated clearing the scene by grasping objects and dropping them into a container to the right side of the robot; however, because we are concerned only with the grasping of objects and not with transporting them after grasping, we manually removed from the gripper objects that were grasped and lifted. The arm started each grasp to the side of the robot as if the previous object had been dropped off into a container. To reset the arm to that position we provided a command that simply dropped the grasped object and moved the arm back to the initial side position.

Each scene contained more objects than could be grasped in the allotted time even by expert users, so no user was able to clear all of them. Grasping stacked or nested objects only counted as one grasped object since only one grasp was involved; the experimenter returned the additional objects to the scene. After each round of grasping, the participant was presented with a questionnaire about his or her experience during the task. At the end of both rounds, the participant filled out a demographics questionnaire and was then debriefed about the purposes of the study.

5.2 Metrics

The task performance metric for each 10-minute round was the number of successful grasps. For each round, we asked users to rate the level of intuitiveness, cooperativeness, and quality of the interface, the level of effort and frustration experienced, and the overall experience in interacting with the robot, using a 7-point scale. We also asked about how well a set of adjectives described the participant's user experience (easy, boring, engaging, difficult, simple, straightforward, fun, intriguing, or visceral), using a 5-point Likert scale. Finally, we measured demographics, including age, gender, video gaming experience, experience with CAD programs and haptic devices, and familiarity with robots.

5.3 Data Analysis

Because each participant experienced both interface types, we ran a mixed analysis of variance (ANOVA) on the data, using user type (two levels: rviz user vs. non-rviz user) as a between-participants independent variable and interface type (two levels: mouse point-and-click interface vs. haptic interface) as a within-participants independent variable. We ran a separate ANOVA for each of the performance metrics and each of the attitudinal metrics.

In the following section, we report upon the results that were found to be statistically significant at the $p<.05$ level. Analyses that did not produce statistically significant results are not reported.

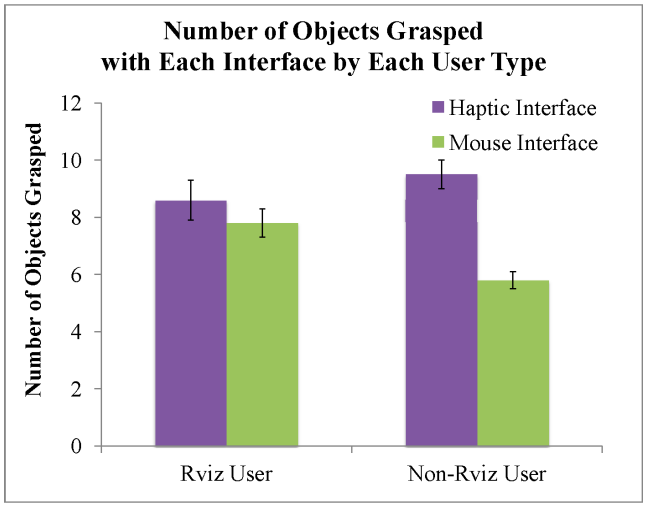


Figure 7: Mean and standard errors for number of objects grasped with each user interface by each user type.

5.4 Study Results

In terms of task performance, people grasped more objects when using the haptic user interface (M=9.1, SE=0.4) than when using the mouse point-and-click user interface (M=6.8, SE=0.4), $F(1,18)$=45.89, $p<.0001$.

There was also a strong interaction effect between user interface type and rviz expertise, $F(1,18)$=14.10, $p<.01$. Non-rviz-users grasped more objects when using the haptic user interface (M=9.5, SE=0.5) than when using the mouse point-and-click user interface (M=5.8, SE=0.3), $F(1,9)$=43.85, $p<.0001$. Rviz-users seemed to grasp slightly more objects when using the haptic user interface (M=8.6, SE=0.7) than when using the mouse point-and-click user interface (M=7.8, SE=0.5), but that difference was not found to be statistically significantly different, $F(1,9)$=2.25, p=.17 (not significant). See Figure 7.

In terms of user experience, rviz users vs. non-rviz users felt differently about each of the user interfaces. There was an interaction between user interface type and rviz experience, $F(1,18)$=4.64, $p<.05$. Rviz users felt that the haptic user interface provided a much more visceral experience (M=4.9, SE=0.3) than the mouse point-and-click user interface (M=3.3, SE=0.3), $F(1,9)$=22.15, $p<.01$. However, non-rviz-users did not notice a significant difference between the haptic user interface (M=4.3, SE=0.4) and the mouse point-and-click user interface (M=4.1, SE=0.2), $F(1,9)$=0.13, p=.73 (not significant).

In summary, people were able to grasp more objects using the haptic user interface than the mouse interface; however, the difference was not significant for users who had used rviz before. Rviz users found the haptic device to be more visceral than the mouse interface, whereas non-rviz users had no such difference.

5.5 Discussion

Overall, the controlled experiment showed that the system we have described enables users to direct a robot for grasping a wide range of objects, even in highly cluttered environments. Operators with no previous experience using rviz, the 3D visualization tool that our interfaces are based on, also performed significantly better when using a haptic input device, as compared to a regular mouse.

Specifying a desired grasp for a robot implies reasoning about the 6D pose of the gripper and possible contacts with the environment. Both of these aspects require a good understanding of the scene from a 3D perspective, which is difficult to obtain from a single viewpoint rendering. One option for operators to better understand the 3D scene was to move the virtual camera. Based on our qualitative observations, we believe that operators with no previous experience using our visualizer did not find this approach intuitive. Observation of mouse interface users from this category revealed that they avoided re-orienting the 3D camera, resulting in difficulty understanding why a particular pose was infeasible (to the autonomous grasp execution) or inaccessible due to the constraint algorithm.

On the other hand, a haptic device can also convey information about the 3D composition of a scene, in a way that proved more helpful to inexperienced operators. Haptic force feedback seemed to provide a more intuitive explanation for why the system was not allowing certain gripper poses. Haptic users quickly responded by trying other grasps, rather than fighting the constraint algorithm.

Finally, users experienced in using the 3D visualization tool were able to achieve the same level of performance using both interfaces. We believe this is due to the fact that our only source of information about the scene was ultimately a vision-based sensor, providing a single-viewpoint depth image. This choice makes our system applicable to a wide range of situations in unstructured environments; it also means that it must cope with the some of the drawbacks of single-view vision data, such as occlusions and incomplete scene models. For an operator experienced with 3D visualization tools, able to extract as much information as possible from such data using a vision-based interface, adding haptic input provided no additional benefits.

As an additional note, we recognize that these experiments were conducted using a device with only three active DOFs (the other three DOFs are tracked but passive), even though the task is 6-DOF. This asymmetry (having more tracked DOFs than actuated DOFs) can in some cases lead to unstable rendering effects as described by Barbagli and Salisbury in [2]. These potentially disruptive artifacts were not experienced in our experiments, likely due to stabilizing effects of mechanism friction and limited torque-inducing interactions. We would expect that using a true 6-DOF feedback device could lead to even more intuitive and accurate virtual prepositioning of the gripper for desired grasps because of the more complete geometric information conveyed by the three additonal channels of torque information. However, even using just three active DOFs is still fairly intuitive and appears to be helpful to users in completing the desired task.

6 Conclusions

In this paper, we have described a system than enables an operator to direct a robot performing grasping tasks. In particular, we focus on the operator's ability to specify an appropriate end-effector pose for picking up a desired object. This component of a grasping task requires selecting an end-effector pose that obtains stable contacts with the target, while avoiding collisions with other parts of the environment.

In our approach, the operator uses a haptic device to position a virtual model of a robot's gripper inside a scene, described using a point cloud obtained from a single image of a depth sensor. A constraint-based haptic algorithm is used to prevent the virtual gripper model from colliding with the environment, while providing force feedback as the virtual model contacts points corresponding to the objects in the scene. After the operator specifies a desired grasp pose for the gripper, the robot attempts to execute it using an autonomously-planned joint trajectory that avoids collisions between the rest of the arm and the environment.

To quantify our methods, we performed a user study where robot operators were instructed to grasp as many objects as possible in a limited amount of time, operating in a highly cluttered environment. We compared the haptic interface described above against a mouse interface, where the haptic device is replaced by a substitute op-

erated through a set of click-and-drag controls. This latter variant used the same algorithm for computing a collision-free pose of the virtual gripper based on operator input; however, being controlled via a regular mouse, it lacks the force feedback capabilities of the haptic device.

Our results showed that operators were able to grasp a wide range of objects, even in the presence of clutter and restrictive obstacles. Furthermore, using a haptic device significantly increased the performance of those users who had no prior experience with the visualization tool that both of our interfaces use for 3D rendering. We believe that, for this group, the haptic interface proved a more intuitive way for conveying spatial relationships between objects and potential collisions in a complex scene. Experienced 3D visualization tool users were able to obtain similar information by moving the virtual camera to change the viewpoint as needed, and achieved similar levels of performance using both interfaces. Our system requires no prior knowledge of the grasped objects or the scene, and we hope these results can be useful for general applications requiring remote manipulation in unstructured, human environments.

Acknowledgements

Many thanks to Leila Takayama for helping with data analysis, as well as to Günter Niemeyer and Reuben Brewer for helpful comments on the manuscript. A. Leeper is supported in part by a National Science Foundation GRFP Fellowship. S. Chan is supported in part by NIH Grant 1R01LM01067301A1 and by a post-graduate scholarship from the National Science and Engineering Research Council (NSERC) of Canada.

References

[1] A. Adamson and M. Alexa. Approximating and Intersecting Surfaces from Points. *Computer*, pages 230–239, 2003.

[2] F. Barbagli and K. Salisbury. The effect of sensor/actuator asymmetries in haptic interfaces. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 140 – 147, March 2003.

[3] A. Bettini, P. Marayong, S. Lang, A. Okamura, and G. Hager. Vision-Assisted Control for Manipulation Using Virtual Fixtures. *IEEE Transactions on Robotics*, 20(6):953–966, Dec. 2004.

[4] D. J. Cannon and G. Thomas. Virtual tools for supervisory and collaborative control of robots. *Presence: Teleoperators and Virtual Environments*, 6(1):1–28, 1997.

[5] J. Cha, M. Eid, and A. El Saddik. DIBHR: Depth Image-Based Haptic Rendering. In *EuroHaptics*, pages 640–650. Springer, 2008.

[6] J. Cha, S.-m. Kim, I. Oakley, J. Ryu, and K. Lee. Haptic interaction with depth video media. In *Advances in Multimedia Information Processing-PCM 2005*, pages 420–430. Springer, 2005.

[7] S. Chan, F. Conti, N. H. Blevins, and K. Salisbury. Constraint-based Six Degree-of-Freedom Haptic Rendering of Volume-embedded Isosurfaces. In *IEEE World Haptics Conference*, 2011.

[8] N. R. El-Far, N. D. Georganas, and A. El Saddick. An algorithm for haptically rendering objects described by point clouds. In *Canadian Conference on Electrical and Computer Engineering*, pages 1443–1448, 2008.

[9] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha. Six degree-of-freedom haptic display of polygonal models. In *Proceedings of the conference on Visualization'00*, pages 139–146. IEEE Computer Society Press, 2000.

[10] S. Hasegawa and M. Sato. Real-time Rigid Body Simulation for Haptic Interactions Based on Contact Volume of Polygonal Objects. In *Computer Graphics Forum*, volume 23, pages 529–538. Wiley Online Library, 2004.

[11] S. Hayati and S. Venkataraman. Design and implementation of a robot control system with traded and shared control capability. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 1310–1315. IEEE Comput. Soc. Press, 1989.

[12] G. Hirzinger. Advances in Robotics: The DLR Experience. *The International Journal of Robotics Research*, 18(11):1064–1087, Nov. 1999.

[13] J.-K. Lee and Y. J. Kim. *Haptic Rendering of Point Set Surfaces*. IEEE, Mar. 2007.

[14] A. Leeper, S. Chan, and K. Salisbury. Constraint-Based 3-DOF Haptic Rendering of Arbitrary Point Cloud Data. In *RSS Workshop on RGB-D Cameras*, 2011.

[15] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *Proceedings of SIGGRAPH '99*, pages 401–408, New York, New York, USA, 1999. ACM Press.

[16] M. Oda, N. Inaba, Y. Takano, S. Nishida, M. Kayashi, and Y. Sugano. Onboard local compensation on ETS-W space robot teleoperation. In *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 701–706. IEEE, 1999.

[17] M. K. O'Malley, A. Gupta, M. Gen, and Y. Li. Shared Control in Haptic Systems for Performance Enhancement and Training. *Journal of Dynamic Systems, Measurement, and Control*, 128(1):75, 2006.

[18] M. Ortega, S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties. *IEEE transactions on visualization and computer graphics*, 13(3):458–69, 2007.

[19] M. A. Otaduy and M. C. Lin. Stable and Responsive Six-Degree-of-Freedom Haptic Manipulation Using Implicit Integration. In *IEEE World Haptics Conference*, pages 247–256. Ieee, 2005.

[20] F. Rydén, H. J. Chizeck, S. N. Kosari, H. King, and B. Hannaford. Using Kinect TM and a Haptic Interface for Implementation of Real-Time Virtual Fixtures. In *RSS Workshop on RGB-D Cameras*, 2011.

[21] T. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA, 1992.

[22] S. P. Walker and J. K. Salisbury. Large haptic topographic maps. In *Proceedings of the 2003 symposium on Interactive 3D graphics - SI3D '03*, page 83, New York, New York, USA, 2003. ACM Press.

[23] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, Aug. 1986.