A Comparison of Remote Robot Teleoperation Interfaces for General Object Manipulation

David Kent
Georgia Institute of Technology
Atlanta, Georgia 30332
dekent@gatech.edu

Carl Saldanha
Georgia Institute of Technology
Atlanta, Georgia 30332
csaldanha3@gatech.edu

Sonia Chernova
Georgia Institute of Technology
Atlanta, Georgia 30332
chernova@cc.gatech.edu

ABSTRACT

Robust remote teleoperation of high-DOF manipulators is of critical importance across a wide range of robotics applications. Contemporary robot manipulation interfaces primarily utilize a free-positioning pose specification approach to independently control each axis of translation and orientation in free space. In this work, we present two novel interfaces, *constrained positioning* and *point-and-click*, which incorporate scene information, including points-of-interest and local surface geometry, into the grasp specification process. We also present results of a user study evaluation comparing the effects of increased use of scene information in grasp pose specification algorithms for general object manipulation. The results of our study show that constrained positioning and point-and-click significantly outperform the widely used free positioning approach by significantly reducing the number of grasping errors and the number of user interactions required to specify poses. Furthermore, the point-and-click interface significantly increased the number of tasks users were able to complete.

Keywords

Robot Teleoperation; User Interface Design; Usability Study

1. INTRODUCTION

Robust remote teleoperation of high-DOF manipulators is of critical importance across a wide range of applications, including assistive robotics, space exploration, search-and-rescue, and web-based robot control. As use cases for remote teleoperation expand beyond pick-and-place to more complex manipulation tasks, there is a growing need to develop effective user interfaces that minimize operator error and improve task efficiency.

Most mouse-and-keyboard robot manipulation interfaces currently utilize a 6-DOF ring-and-arrow marker for independently controlling each axis of rotation and translation. The marker can be used to directly move an end-effector or set pose goals to which the robot can autonomously plan and move. Leeper et al. [13] evaluated multiple interfaces based on the ring-and-arrow marker design, showing that increasing robot autonomy results in better performance in object picking tasks.

In this work, we view interfaces for remote robot control as falling on a spectrum of reliance on scene information. The ring-and-arrow marker makes no use of scene information, giving robot operators complete control over the pose of their robot's end-effector in free space. While this degree of flexibility gives users the maximum level of control, the size of the workspace can become a burden that can increase the likelihood of operator error. Our work compares the ring-and-arrow marker to alternate approaches for human-in-the-loop robot manipulation that leverage depth information. We focus specifically on mouse-and-keyboard approaches, as these input devices are readily available to most of the population. We introduce two alternate approaches to the ring-and-arrow marker – one that uses depth information at a single point in the environment, and another that proposes grasp poses based on local 3D scene geometry – to explore the trade-offs inherent to scene information use in a general robot manipulation context. The presented techniques do not use scene or object recognition; as a result, despite using scene information, they can be immediately deployed in new environments without training. We evaluate all three interaction approaches over a variety of manipulation tasks, from pick-and-place to more complex tasks including opening containers and pouring.

This paper makes three contributions to teleoperation for manipulation. The first is a novel interaction method for positioning a 6-DOF end-effector that we call *constrained positioning*. We designed this method to incorporate a small amount of scene information into an approach that provides users with fine-grained control over grasp pose specification. The result is a constrained step-by-step approach that allows users to set a grasp point, approach angle, and grasp depth with a reduced number of mental transformations.

The second contribution is an extension to the AGILE grasp approach [22] for general object manipulation. We present a heuristic-based approach that replaces AGILE's classification step, resulting in an autonomous grasp calculator better suited to manipulating man-made objects during human-in-the-loop operation. We leverage the resulting approach, which we call *point-and-click*, to create an interaction method that incorporates a large amount of scene information, requiring only supervisory control from a user.

Our final contribution is a usability comparison of the

HRI '17, March 06-09, 2017, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2909824.3020249

above human-in-the-loop object manipulation approaches. We present the results of a remote manipulation user study that compare the efficiency and effectiveness of the *constrained positioning* and *point-and-click* approaches with the ring-and-arrow marker approach. Our results show with statistical significance that *constrained positioning* and *point-and-click* outperform the widely used ring-and-arrow marker approach. Both approaches significantly reduce the number of grasping errors and the number of interactions required by the user. Furthermore, *point-and-click* significantly increases the number of tasks completed and significantly decreases average task completion time.

2. RELATED WORK

Unconstrained methods allowing individual control of task space translation and rotation are widely used in robotics. Leeper et al. present and evaluate an interactive marker composed of rings and arrows (Figure 1) for just such a purpose [13], which can be rendered in a 3D scene such as rviz [19] or online via Robot Web Tools' ros3djs [24]. The marker can be used at varying levels of robot autonomy, including (in order of increasing robot autonomy) a) real-time arm position control, b) setting waypoints executed through linear interpolation, c) setting pose goals followed by autonomous trajectory execution with obstacle avoidance, and d) adjusting autonomously calculated grasp poses for fully autonomous grasp execution. Leeper et al. found the ring-and-arrow marker to be an effective tool for an object picking task, noting that users of the interface made fewer mistakes as the autonomy of the robot increased. Ciocarlie et al. showed the same marker could be used for general mobile manipulation in household environments [2]. The Robots for Humanity project [1] also uses these 6-DOF markers for teleoperating a robot to perform everyday tasks. Many teams competing in the DARPA Robotics Challenge used this same marker for multiple purposes, including positioning key points of a humanoid robot for balance or manipulation [26, 12, 3], positioning 3D object templates over sensor data [12], and correcting point cloud alignment [18]. The ring-and-arrow marker has also found use in web interfaces. In previous work, we used the marker in an interface for crowdsourcing object recognition and picking [11]. More recently, Grice and Kemp showed similar controls superimposed on a camera stream displayed in a web interface, rather than rendering them in a 3D environment [10].

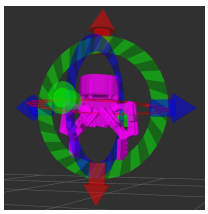


Figure 1: 6-DOF positioning ring-and-arrow marker

Despite its widespread use, the ring-and-arrow approach requires the user to perform many mental 3D rotations, which Zhai and Milgram show to be mentally taxing [25]. When the marker is rendered in a 3D environment, the user has to control both the transformation of the camera within the 3D environment and the transformation of the 6-DOF marker itself. When projected onto a 2D camera feed, manipulating the marker causes it to become unaligned with the user's viewing angle, which Parsons identifies as a particularly difficult case requiring complex mental transformations [17]. There are two approaches to alleviate this difficulty: isolate transformations where possible (e.g. separating rotation and translation) [16], or reduce the number of transformations that the user is required to make [4]. Both of these approaches can be accomplished by incorporating more scene information into grasp pose specification methods, and as such they inspire the *constrained positioning* and *point-and-click* methods contributed in our work.

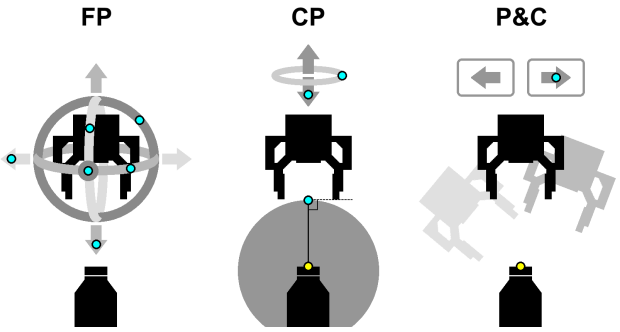


Figure 2: Interaction approaches arranged by number of major interaction points. Yellow points represent setting initial grasp poses; blue points represent pose refinement

Alternative control methods for 6-DOF object positioning have been studied primarily outside the field of robotics. By using a smartphone as an input device, researchers have shown multiple interfaces for controlling objects in 3D video games [9] and editing meshes in 3D space [7]. These approaches typically involve separating translations and rotations by assigning them to either the touch screen or the motion of the phone itself. Navidget, a tool for free-space camera positioning developed specifically for 2D input devices, provides another method of 6-DOF control. It separates translations and rotations into a multi-step process, while additionally reducing the total number of positioning operations by fixing the interaction to a local area and a selected approach angle [5].

Another approach to grasp pose specification is to completely remove the requirement for transformations by autonomously calculating grasp poses. This is a difficult problem for novel and unrecognized objects, particularly because point cloud data provides only partial object shape information. Hsiao et al. present a heuristic-based approach, biasing grasps to orthogonal overhead or side poses [8]. Other work uses classifiers with hand-specified features. One example uses contacts, symmetry, and force-closure as features[20], used to classify grasp stability. Another example is the AGILE grasp approach [22], which classifies antipodal grasps. Other work detects grasping affordances to identify handle-like geometry [23]. Deep learning provides another classification approach which learns the grasp features [15, 14].

3. INTERACTION APPROACHES

In this work, instead of focusing on a single interface, we examine a spectrum of grasp pose specification approaches for remote manipulation interfaces. Based on our literature review, we identified three interaction approaches for specifying 6-DOF end-effector poses[1]. The first approach, *free positioning* (*FP*), leverages the ring-and-arrow marker frequently used in manipulation teleoperation. The approach

[1]A video of our three approaches can be found here: https://youtu.be/ySgwyTsOkkY

uses no scene information, instead providing the user with full control over transformations about every Cartesian axis. The second approach, *constrained positioning* (*CP*), leverages a point-of-interest in the scene to constrain the pose specification process. Our third method, *point-and-click* (*P&C*), enables the robot to autonomously calculate and propose grasp poses by incorporating a local surface extracted from the scene.

The relationship between the three approaches is shown in Figure 2. We selected these three techniques to incorporate increasing use of scene information into the grasp specification process. We hypothesize that incorporating more scene information into the pose specification algorithms will reduce user effort by reducing the number of interactions and the time required to specify a grasp pose, as well as lower the required situational awareness of the user by offloading some of the decisions onto the algorithm.

After pose specification, the three approaches handle grasp execution in the same manner. Once the user confirms the pose, the robot uses the same motion planning algorithm and execution control loop to move to the specified location. All three approaches require at most a point cloud as input, and do not rely on object recognition. As such they can be used in novel environments with no need for training time on new objects.

3.1 Free Positioning

Our first approach allows the user to position the robot's end-effector into any pose. We use the standard ring-and-arrow marker and base the approach on the Grasp Execution strategy described by Leeper et al. [13], rendered in a 3D viewer. As such, the user first sets a pose by clicking and dragging on any of the three arrows or rings corresponding to the three Cartesian axes of translation and rotation, respectively. Once the grasp pose is set, the arm autonomously plans and executes an obstacle-avoiding trajectory resulting in a grasp at the specified pose.

We chose this supervisory autonomous planning and execution approach over direct control for two reasons. First, Leeper et al. showed approaches in which the robot performs more autonomous trajectory planning and execution result in better performance (i.e. more successful grasps and fewer collisions). Second, since our system is targeting remote teleoperation, latency is an important concern. Direct control is highly sensitive to latency as it requires real-time visual feedback from the robot's sensors, whereas supervisory methods are virtually invariant to latency, since all of the execution is performed on-board [21]. All ring-and-arrow interactions are performed client-side, so they are not limited by latency.

3.2 Constrained Positioning

Our second approach seeks to separate the required positioning transformations in an intuitive manner, as well as to reduce the number of transformations wherever possible. We base our approach on the Navidget interaction technique [5] designed for 3D camera positioning. Camera positioning is a useful analogy for end-effector positioning, as both involve specifying a 3D transformation using 2D inputs. Extending a Navidget-like approach to robot grasping results in a novel interaction method for constrained end-effector positioning. The constrained positioning approach is a 3-step process, visualized in Figure 3 and explained below.

1. *Select a grasp point.* The user first specifies a point-of-interest, in this case a grasp point, by clicking on a camera stream. This constrains the final grasp pose to one that must pass through the selected point.

2. *Set an approach angle.* Once the grasp point is selected, *constrained positioning* renders a sphere around the point-of-interest. The user can then set an approach angle with a single click on the sphere's surface. The approach angle is determined by calculating a pitch and yaw angle that create an orthogonal vector to the surface of the sphere. In this way, the user can specify two degrees of rotation in a single action.

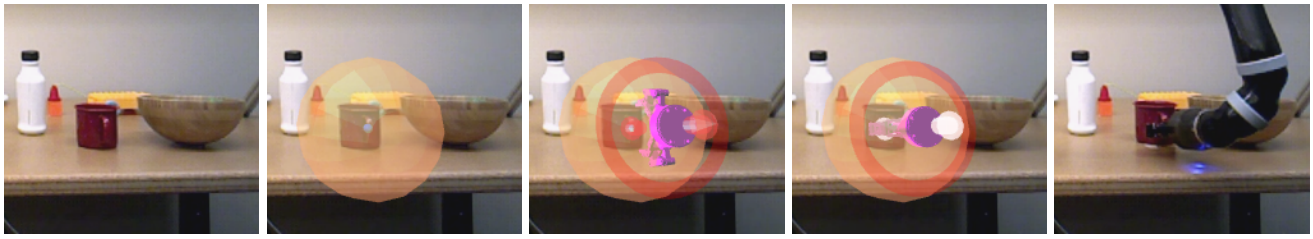
3. *Adjust the roll and grasp depth.* Navidget does not include a method for controlling the camera's roll, and while it does include a method for setting the depth of the camera, it does not translate well to robot grasping. As such, we include a single ring and arrow marker so that the user can adjust the wrist roll and the depth of the grasp by clicking and dragging the ring and arrow, respectively.

By using a small amount of scene information (the selected point-of-interest), we are able to base the approach around a grasp point and an approach angle, reducing the number of transformations the user is required to input from 6 degrees of freedom (3 rotations and 3 translations) to 4 degrees of freedom (3 rotations and a single translation). Furthermore, the sphere metaphor allows us to group 2 rotations into a single action with an intuitive visual-spatial representation. Since Navidget was designed with 2D input in mind, the entire process naturally lends itself to superimposition over a camera stream. As such, we treat this method as an Augmented Reality (AR) approach that does not require a separately rendered 3D scene. By eliminating the 3D rendered scene, we save screen real estate, reduce the amount of context-switching required by the user, and reduce latency by eliminating the need to stream point clouds.

3.3 Point-and-Click

Our third approach autonomously calculates grasp poses based only on a single mouse click. We base the grasp calculation on the AGILE grasp method [22]. The original AGILE algorithm has two steps. First, it samples a set of antipodal gripper hypotheses from a point cloud within a given workspace. The sampling algorithm samples the space around the point cloud to find gripper poses that do not intersect the point cloud, but do contain a subset of points between the gripper's fingers. Since these gripper poses are calculated from a point cloud, the algorithm assumes some shape information is missing due to occlusion. To address this, the second step of the algorithm is to classify the gripper pose hypotheses as antipodal or not, using a HOG descriptor to encode the points contained within each gripper pose hypothesis.

Our extension to AGILE replaces the antipodal gripper pose classifier with a heuristic-based approach. We found that the antipodal classifier was too conservative for our general manipulation tasks, frequently resulting in only one or zero grasps. Since we designed a human-in-the-loop system, we require an autonomous grasp calculator that provides more grasp options for the user to choose from. With that in mind, we designed a set of heuristics that result in



(a) Original image stream (b) Clicking the mug handle selects it as the grasp point (c) Clicking on the sphere sets an approach angle (d) Ring and arrow markers can be used to adjust wrist roll and grasp depth (e) The robot executes the final grasp

Figure 3: Constrained positioning steps

Algorithm 1 Autonomous Grasp Planning and Ranking

Require: Point *pointOfInterest*, PointCloud *cloud*
1: Workspace w = createWorkspace(*pointOfInterest*);
2: List<Pose> *hands* = findGraspsAGILE(*cloud*, w);
3: List<Pose> *grasps* = cluster(*hands*);
4: PointCloud pc = crop(*cloud*, w);
5: Plane p = planeSegmentation(pc);
6: Orientation n = surfaceNormal(p)
7: List<PointCloud> *clusters* = cluster(pc);
8: PriorityQueue<Pose> *rankedGrasps*;
9: **for all** g in *grasps* **do**
10: PointCloud c = nearestCluster(*clusters*, g);
11: $h_n = n_n$*distance(*g.orientation*, n);
12: $h_o = n_o$*dstToOrthogonal(*g.orientation*, PCA(c));
13: $h_p = n_p$*distance(*g.position*, *pointOfInterest*);
14: $graspRating = \alpha * h_n + \beta * h_o + (1 - (\alpha + \beta)) * h_p$
15: *rankedGrasps*.push(g, *graspRating*);
16: **end for**
17: **return** *rankedGrasps*;

approximately 5 to 20 potentially effective grasps for the objects used in our general manipulation tasks. We introduce three metrics:

- h_n minimizes the distance between the orientation of the grasp's approach angle and the orientation of the dominant plane's surface normal. Most manipulation features, such as handles and knobs, are situated such that perpendicular grasp poses are more effective.
- h_o represents whether the grasp's orientation is orthogonal to the principal direction (computed with Principal Component Analysis) of the local geometry around the grasp point. This comes from our observation that most man-made objects are designed to be grasped with poses orthogonal to their principal direction, i.e. poses that are "aligned" with the object.
- h_p minimizes the proximity of the grasp point to the algorithm's input point. Since the user specifies a grasp area by inputting an initial grasp point, the third heuristic prioritizes grasps that are closest to the user input.

Each heuristic is multiplied by a normalizing constant (n_n, n_o, and n_p) to place them on a [0,1] scale.

The full algorithm for leveraging the above grasp pose estimates and heuristics is shown in Algorithm 1. The grasp planner takes as input a point-of-interest and a point cloud. All calculations occur within a local workspace (in our case, a 10 cubic centimeter volume) situated around the point-of-interest. The algorithm first uses AGILE grasp to compute a set of hand hypotheses, which we then cluster by position and orientation into a set of unranked grasps (lines 2-3). Note that the hand hypotheses from the AGILE grasp pipeline are not classified by the AGILE classifier. Next, the algorithm performs a few calculations required to compute the heuristics, including cropping the point cloud to the workspace, segmenting the dominant plane, computing its surface normal, and separating the cropped point cloud into local geometry clusters based on Euclidean distance (lines 4-7). For each grasp, the algorithm computes a ranking based on our three heuristics (lines 10-14). The final ranking is determined from a linear combination of the three heuristics, which can be assigned different priority by adjusting the constants α and β (line 14). The ranked grasps are then returned in a priority queue where lower rankings are preferred. For our experiments, we used experimentally determined values of $\alpha = 0.6$ and $\beta = 0.25$, which prioritize perpendicular grasps most, and user input least.

We leverage the above algorithm to create a point-and-click user interface. The user first inputs the point-of-interest by clicking on a point within a camera stream. Algorithm 1 then autonomously calculates a ranked list of grasp points within the local area around the clicked point. The ranked list is presented to the user, who then selects a final grasp to execute. As with *constrained positioning*, even though the algorithm relies on 3D data, we can show only the 2D camera stream to the user as the input to the interface requires only a single click.

4. EVALUATION STUDY DESIGN

We performed an evaluation study in order to compare the effectiveness of the three interaction approaches identified in the previous section. Our system is designed for collecting robot demonstrations from remote users. The robot itself consists of a Kinova JACO2 arm with a Robotiq-85 2-finger gripper and two Asus Xtion PRO LIVE 3D cameras mounted to a workbench. The user interfaces are implemented using Robot Web Tools [24] so that they can be displayed in a web browser.

4.1 Procedure

We recruited 45 participants from a college campus to take part in a between-subjects study. Participants ranged in age from 20-39, with 30 males and 15 females. Each participant received $10 as compensation. Participants were asked to rate their experience on a 1 (no experience) to 5 (very experienced) scale with robotics (2.8±1.2), video games (3.4±1.3),

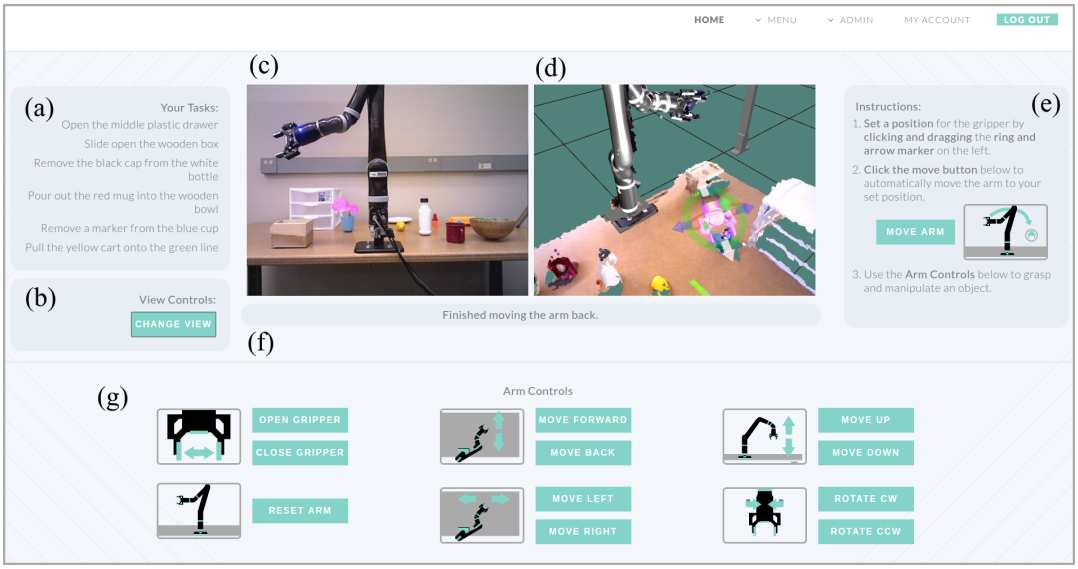


Figure 4: Complete study interface showing elements for the *free positioning* condition. (a) List of tasks (b) Controls to cycle through the top and side camera streams (c) Live camera stream with AR overlay; AR overlay contents and interactivity depend on the condition (d) Rendered 3D viewer, only used in *free positioning* (e) Manipulation instructions specific to the interface condition (f) Feedback bar displaying the current action being executed, successful execution of an action, or failed execution of an action with suggestions for improvement (g) Primitive actions for controlling the arm and gripper

and 3D software (3.0 ± 1.3). Participants were then assigned to one of three conditions, with 15 participants per condition, balanced by gender and level of robotics experience.

Upon arriving, participants were brought to a computer out of sight and sound of the lab where the robot was operating. After completing the demographics and experience survey, participants were given up to 10 minutes to complete two training tasks using the interface to which they were assigned. During training, they could ask the researchers questions about using the interface. Once training was complete and participants had no further questions, they were instructed to complete as many tasks as possible within a 30 minute time period. Upon completion of the tasks or at the end of the allotted time, participants completed a NASA TLX [6] exit survey to evaluate workload while using their respective interfaces.

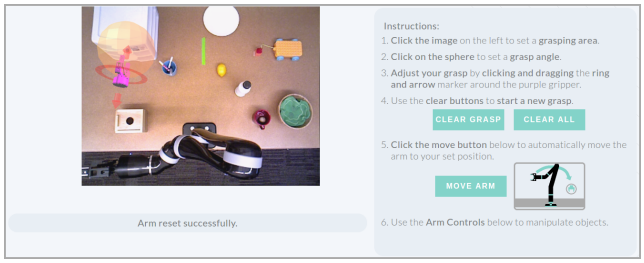
Each participant was given the same list of tasks to complete during the study. They could complete the tasks in any order, and skip or return to previously attempted tasks at any time. The tasks were as follows:

- *(Training) Move the lemon onto the white plate.* A simple pick-and-place task.
- *(Training) Reset the arm when you're finished.* An introduction to the reset primitive action, which moves the arm to a preset position that does not occlude the objects in either camera view. This also ensured the main part of the experiment would begin with the arm in a consistent starting pose.
- *Open the middle plastic drawer.* A task to grasp the correct handle in a set of stacked drawers.
- *Slide open the wooden box.* Manipulation of an uncommon object that typically required some exploration.
- *Remove the black cap from the white bottle.* A precision task requiring rotation and translation.
- *Pour out the red mug into the wooden bowl.* A task that could only be completed by performing an appropriate grasp for pouring on the mug's handle.
- *Remove a marker from the blue cup.* A cluttered-environment manipulation.
- *Pull the yellow cart onto the green line.* A task involving manipulating an object by pulling a string.

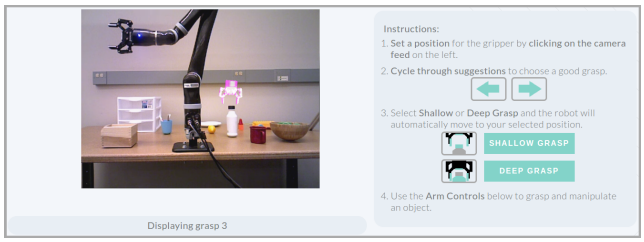
4.2 Conditions and Interfaces

The interface used for the study can be seen in Figure 4. In every condition, the interface included the same presentation of the tasks, a camera stream that can be switched between the top and side camera, a text-based feedback bar, and a set of primitive actions. The content of the AR camera stream overlay (c), inclusion of the 3D viewer (d), and the set of manipulation instructions (e) varied among the conditions.

The primitive actions included generally useful actions such as fully opening or closing the gripper and resetting the arm to a position that did not occlude the workspace from either camera view. The remaining actions were selected as the minimum set of primitive actions required to complete any task after performing a grasp. This included fixed-distance 10 centimeter translations in each Cartesian direction (with respect to a coordinate frame aligned with the cameras) and 90 degree clockwise and counter-clockwise gripper rotation with respect to the JACO's wrist. We specifically chose these actions and fixed distances so that the tasks could be completed, thus evaluating the quality of the grasps for each task, while also making it virtually impossible to skip the initial grasp specification by using only the primitive actions to complete a task. In this way we



(a) Camera stream with AR overlay and manipulation instructions used for the *constrained positioning* interface. These replace elements *c*, *d*, and *e* in Figure 4. This example shows a grasp being specified on the drawers using the top camera view



(b) Camera stream with AR overlay and manipulation instructions used for the *point-and-click* interface. These replace elements *c*, *d*, and *e* in Figure 4. This example shows a grasp calculated for the bottle cap displayed on the side camera view

Figure 5: Differences between the three interface conditions

could ensure the users would need to use the interface elements specific to their condition, which were the main focus of our evaluation.

4.2.1 Free Positioning Interface

The first condition uses the *free positioning* approach, described in Section 3.1. As such, it is very similar to our previous remote pick-and-place interface [11], containing a 3D viewer side-by-side with the camera stream. Along with the ring-and-arrow marker, the 3D viewer includes renderings of point clouds from the two depth cameras and a model of the robot in its current pose, to provide context and increase the user's situational awareness. The user can adjust the viewpoint of the 3D viewer by rotation, translation, and zoom. The AR overlay also displays a rendering of the pose goal with respect to the currently selected camera view, but all of the interaction occurs within the 3D viewer.

4.2.2 Constrained Positioning Interface

The second condition corresponds to the *constrained positioning* approach, described in Section 3.2. Since this method is designed specifically for 2D inputs, it does not require a 3D viewer, and so the user is provided with only the switchable camera stream and an AR overlay. The unique components of the *constrained positioning* interface are shown in Figure 5a, replacing elements (c), (d), and (e) in Figure 4. The initial grasp point is set by clicking directly on a point in the camera stream, and the interaction then follows the step-by-step process outlined in 3.2, with all interactions occurring within the AR overlay.

4.2.3 Point-and-Click Interface

The third condition corresponds to the *point-and-click* approach described in Section 3.3. The user inputs a point-of-interest by clicking on the camera stream. The autonomous grasp planner then calculates a set of suggested grasps based on the local scene geometry around the point-of-interest. The AR overlay displays one suggested grasp at a time, starting with the best ranked grasp, which the user can change by clicking the next and previous arrows. If they see a grasp they like, the user can initiate grasp execution by selecting a preset grasp depth of "shallow" or "deep", depending on the task. If they do not find a suitable grasp, the user can click on another point at any time to calculate and rank a new set of grasps. Since the only interaction comes from clicking on a point in the camera streams, this condition does not require a 3D viewer. The unique components of the *point-and-click* interface are shown in Figure 5b, replacing elements (c), (d), and (e) in Figure 4.

Table 1: Measures and descriptions

tasks completed	$number\ of\ completed\ tasks$
task success rate	$\frac{completed\ tasks}{attempted\ tasks}$
time per completed tasks	$\frac{\sum elapsedTime(completed\ tasks)}{completed\ tasks}$
errors per task	$\frac{misses+bad\ grasps}{attempted\ tasks}$
interactions per task	$\frac{interactions}{attempted\ tasks}$

4.3 Measures

We used multiple measures to evaluate users' performance, efficiency, and workload between the three interfaces.

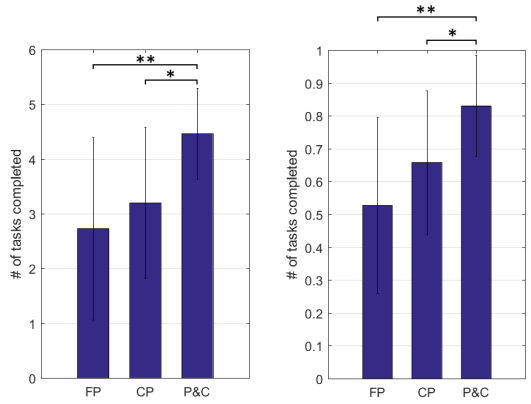
4.3.1 Observed Measures

We identify a set of observable measures intended to measure each participant's performance (*tasks completed*, *task success rate*, *errors per task*) and efficiency (*time per completed tasks*, *interactions per task*). These measures are all calculated from values logged during the main portion of the study, after the training period concluded. Definitions of the measures are listed in Table 1.

We define a *completed task* as any task in which the goal was met at some point within the 30 minute time period, a *failed task* as any task which was rendered impossible to complete (e.g. pouring out the mug somewhere other than the bowl), and an *attempted task* as any task the user devoted time towards completing. We define a *miss* as an attempted grasp that does not contact an object or the environment, and a *bad grasp* as any grasp that does make contact but cannot be used to complete the task. We define an *interaction* as any mouse button press that involves specifying or executing a grasp. This includes dragging a ring-and-arrow marker (*free positioning* and *constrained positioning*), setting or clearing a grasp point or approach angle (*constrained positioning*), calculating a new set of grasps or viewing a different calculated grasp (*point-and-click*), executing a grasp (all conditions), repositioning the virtual camera in the 3D viewer (*free positioning*), and changing camera streams (all).

4.3.2 Self-Reported Measures

Each participant completed a NASA TLX form upon completion of the study, to measure workload for each of the interfaces, including *mental demand*, *physical demand*, *temporal demand*, *performance*, *effort*, and *frustration*.



(a) Mean number of tasks completed (±1 SD)

(b) Mean task success rate (*tasks completed* / *tasks attempted*, ±1 SD)

Figure 6: Task completion measures

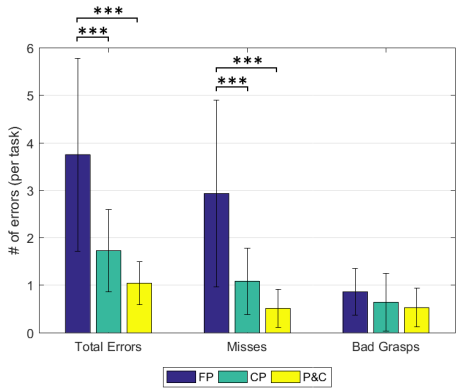
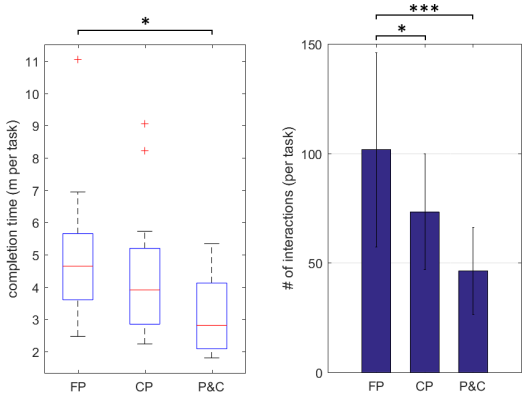


Figure 7: Breakdown of mean errors (±1 SD)



(a) Mean time per task (±1 SD)

(b) Mean number of interactions (±1 SD)

Figure 8: Efficiency measures

5. RESULTS

For each of our performance and efficiency measures described in Section 4.3.1, we conducted a one-way between subjects Analysis of Variance (ANOVA) to compare the effects of user control level on each measure. If a significant effect was found, we performed a Tukey HSD post test to determine which conditions significantly differed. We present the results for the set of observed measures below.

We first present task completion measures, shown in Figure 6. ANOVA showed a significant effect of scene information incorporation for both *tasks completed* ($F(2,42) = 6.75, p = 0.0029$) and *task success rate* ($F(2,42) = 7.20, p = 0.0021$) for the three conditions. *Point-and-click* significantly outperformed both *free positioning* ($p = 0.0027$) and *constrained positioning* ($p = 0.034$) in number of tasks completed. We note a similar result with task success rate, where users of the *point-and-click* interface had significantly higher completion rates over their attempted tasks than users of the *free positioning* interface ($p = 0.0014$), and the same trend appears between *point-and-click* and *constrained positioning* ($p = 0.09$). Taken together, we observe that incorporating more scene information into the grasp pose specification algorithm results in both a greater quantity of tasks being completed with a greater quality of attempt than algorithms incorporating little to no scene information. We found no significant difference between *free positioning* and *constrained positioning*, although we note that they have comparatively high variance (see Section 6).

Number of errors made, including a breakdown of errors into misses and bad grasps, is shown in Figure 7. Our analysis showed a significant effect of scene information incorporation on number of errors for the three conditions ($F(2,42) = 18.39, p < 0.001$). Both *constrained positioning* and *point-and-click* differed significantly from *free positioning* ($p < 0.001$ in both cases), although there was no significant difference between the two. The same relationship can be seen when accounting for time, shown in Table 2. As such, we observe that grasp specification algorithms that do not use any scene information introduce a risk of increased operator error.

The average time per completed task was 5.04 ± 2.19 min, 4.41 ± 2.03 min, and 3.17 ± 1.16 min for *free positioning*, *constrained positioning*, and *point-and-click*, respectively. We note that for the *free positioning interface*, two users were unable to complete any tasks, and so their data is omitted from the completion time analysis. Our analysis showed a significant effect of scene information incorporation on completion time ($F(2,40) = 3.87, p = 0.029$) for the three conditions. *Point-and-click* users completed tasks significantly faster than *free positioning* users ($p = 0.026$). Similar to the task completion measures, *free positioning* and *constrained positioning* had higher variance in task completion time, resulting in more outliers than *point-and-click* (Figure 8a).

Our analysis showed scene information incorporation had a significant effect on the number of interactions ($F(2,42) = 11.31, p < 0.001$) for the three conditions, shown in Figure 8b. *Free positioning* significantly differed from *constrained positioning* ($p = 0.049$) as well as *point-and-click* ($p < 0.001$). *Constrained positioning* and *point-and-click* exhibit the same trend ($p = 0.065$), favoring greater use of scene information. *Interactions per minute* are shown in Table 2. We observe a direct relationship where incorporating more scene information into grasp specification algorithms reduces the number of interactions required by the user, resulting in interfaces that are more efficient to use.

Based on all of the results, it is apparent that both of the novel conditions outperform the commonly used *free positioning* condition. *Constrained positioning* and *point-and-click* significantly reduced the number of grasping errors per task, and they require significantly fewer interactions per

Table 2: Time-controlled measures

measure	*FP*	*CP*	*P&C*
errors/min	0.58 ± 0.19	0.33 ± 0.14	0.28 ± 0.19
interact./min	15.1 ± 4.6	11.5 ± 3.3	10.1 ± 4.8

Table 3: NASA TLX Results

measure	*FP*	*CP*	*P&C*
total workload	64.5 ± 10.9	64.1 ± 14.1	57.5 ± 10.6
mental demand	12.9 ± 8.4	11.9 ± 9.1	12.1 ± 8.4
phys. demand	4.5 ± 6.6	2.5 ± 5.1	0.9 ± 2.7
temp. demand	9.1 ± 7.7	9.2 ± 7.4	8.2 ± 6.9
performance	10.5 ± 6.9	11.8 ± 8.5	13.1 ± 8.5
effort	13.3 ± 5.4	12.6 ± 5.3	11.9 ± 4.7
frustration	14.2 ± 10.6	16.0 ± 11.4	11.2 ± 10.5

task. Between *constrained positioning* and *point-and-click*, the *point-and-click* interface has the advantage. Along with a reduction in errors and interactions, *point-and-click* also results in a significantly better task completion rate, a significantly higher number of tasks completed, and a significantly lower completion time per task as compared to *free positioning*.

We found no significant correlations between robotics experience, video game experience, or 3D software experience and any of our measures. We also found no statistically significant differences between the workload reported by participants in each of the three conditions, or between any of the individual components contributing to workload. The results of the NASA TLX are reported in Table 3.

6. DISCUSSION AND FUTURE WORK

Our results consistently show that interaction approaches with greater incorporation of scene information result in improved performance for manipulation tasks across all of our objective measures. In particular, the *point-and-click* interface allowed users to complete a greater number of tasks more quickly, complete tasks more consistently, and make fewer mistakes than users with the *free positioning* and *controlled positioning* interfaces, all while performing fewer interactions.

The *point-and-click* condition had notably smaller variance than the other two conditions. The cause of the large variance in the *free positioning* and *constrained positioning* conditions comes from the potentially high number of errors. The full 6-DOF control of *free positioning*, and to a lesser extent the 4-DOF control of *constrained positioning*, create such a large workspace of possible grasp poses that non-carefully-planned movements will usually result in a failed grasp. The outliers in the *free positioning* and *constrained positioning* conditions spent the majority of their time attempting a single task, moving the gripper around the object without successfully grasping it. For the *point-and-click* interface, however, simply clicking on the object to be manipulated often resulted in a successful grasp, thus lowering the variance among non-expert users. Non-expert user variance is an important concern when designing interfaces meant to be deployed to the wider population. We leave identification of the causes of the non-expert user performance variance to future work.

While *point-and-click* had the clearest advantages over the other interfaces, *constrained positioning* does have a significant advantage over *free positioning* in reducing the number of errors made by users. The most frequent type of error was missing a grasp, in which the gripper failed to make contact with any part of the environment, object or otherwise. Both *constrained positioning* and *point-and-click* require the user to focus their end-effector positioning around a point cloud by clicking on a point to initiate the interaction. Constraining the interaction to a physical surface significantly reduces the number of missed grasps, resulting in more efficient use of the arm.

Dependency on a point cloud does create limitations for the new interaction approaches, however. *Point-and-click* requires an accurate point cloud. For difficult to detect objects, such as highly specular objects, it will not be able to calculate a good set of grasps. For objects undetectable by depth camearas, such as transparent objects, neither *constrained positioning* nor *point-and-click* will work, since the initial point of interest cannot be selected. An ideal interface would include an option to switch to *free positioning* for redundency in these cases.

A final interesting point comes from the NASA TLX data. While there are clear advantages to incorporating more scene information in manipulation interfaces (including a significant reduction in number of interactions), users do not appear to feel any reduction in workload. To further explore this, we suggest conducting a within-subjects evaluation study so that more subjective measures about preferred interaction methods can be collected.

7. CONCLUSION

This work introduces alternative 6-DOF end-effector positioning approaches to the widely used ring-and-arrow marker for remote object manipulation, by incorporating scene information into the grasp specification algorithms. The results are a constrained positioning approach which reduces pose specification to a 4-DOF process performed in 3 steps, and a point-and-click approach which leverages autonomous grasp planning to simplify pose specification to a single click.

The results of our evaluation study show that the common ring-and-arrow marker free positioning approach is significantly less effective than our novel approaches. Our constrained positioning and point-and-click approaches translate to more efficient user interfaces, resulting in significantly fewer errors and requiring significantly fewer interactions than the free positioning interface. Further, the point-and-click interface allowed users to complete a significantly greater number of general manipulation tasks with a significantly higher success rate per attempted tasks than the common ring-and-arrow approach. Despite measurable improvements in performance, however, users did not report a reduction in workload for either the constrained positioning or point-and-click approaches.

8. ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation award number IIS 13-17775 and the Office of Naval Research award number N000141410795.

9. REFERENCES

[1] T. L. Chen, M. Ciocarlie, S. Cousins, P. M. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C.-H. King, D. A. Lazewatsky, H. Nguyen, et al. Robots for humanity: A case study in assistive mobile manipulation. 2013.

[2] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow. Mobile manipulation through an assistive home robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5313–5320, Oct 2012.

[3] M. DeDonato, V. Dimitrov, R. Du, R. Giovacchini, K. Knoedler, X. Long, F. Polido, M. A. Gennert, T. Padır, S. Feng, et al. Human-in-the-loop control of a humanoid robot for disaster response: A report from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):275–292, 2015.

[4] B. P. DeJong, J. E. Colgate, and M. A. Peshkin. Improving teleoperation: reducing mental rotations and translations. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3708–3714. IEEE, 2004.

[5] M. Hachet, F. Decle, S. Knodel, and P. Guitton. Navidget for easy 3d camera positioning from 2d inputs. In *Proceedings of the 2008 IEEE Symposium on 3D User Interfaces*, 3DUI '08, pages 83–89, Washington, DC, USA, 2008. IEEE Computer Society.

[6] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[7] A. Henrysson and M. Billinghurst. Using a mobile phone for 6 dof mesh editing. In *Proceedings of the 8th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI*, pages 9–16. ACM, 2007.

[8] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones. Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1228–1235. IEEE, 2010.

[9] N. Katzakis, M. Hori, K. Kiyokawa, and H. Takemura. Smartphone game controller. In *Proceedings of the 74th HIS SigVR Workshop*. Citeseer, 2011.

[10] C. C. Kemp and P. M. Grice. Assistive mobile manipulation: Designing for operators with motor impairments. In *RSS 2016 Workshop on Socially and Physically Assistive Robotics for Humanity*, 2016.

[11] D. Kent, M. Behrooz, and S. Chernova. Construction of a 3d object recognition and manipulation database from grasp demonstrations. *Autonomous Robots*, 40(1):175–192, 2016.

[12] S. Kohlbrecher, A. Romay, A. Stumpf, A. Gupta, O. Von Stryk, F. Bacim, D. A. Bowman, A. Goins, R. Balasubramanian, and D. C. Conner. Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials. *Journal of Field Robotics*, 32(3):352–377, 2015.

[13] A. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow. Strategies for human-in-the-loop robotic grasping. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 1–8, March 2012.

[14] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[15] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*, 2016.

[16] M. R. Masliah and P. Milgram. Measuring the allocation of control in a 6 degree-of-freedom docking experiment. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 25–32. ACM, 2000.

[17] L. M. Parsons. Inability to reason about an object's orientation using an axis and angle of rotation. *Journal of experimental psychology: Human perception and performance*, 21(6):1259, 1995.

[18] C. Phillips-Grafflin, N. Alunni, H. B. Suay, J. Mainprice, D. Lofaro, D. Berenson, S. Chernova, R. W. Lindeman, and P. Oh. Toward a user-guided manipulation framework for high-dof robots with limited communication. *Intelligent Service Robotics*, 7(3):121–131, 2014.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[20] A. Saxena, L. L. Wong, and A. Y. Ng. Learning grasp strategies with partial shape information. In *AAAI*, volume 3, pages 1491–1494, 2008.

[21] T. B. Sheridan. Space teleoperation through time delay: review and prognosis. *IEEE Transactions on robotics and Automation*, 9(5):592–606, 1993.

[22] A. ten Pas and R. Platt. Using geometry to detect grasp poses in 3d point clouds. In *IntâĂŹl Symp. on Robotics Research*, 2015.

[23] A. Ten Pas and R. Platt. Localizing handle-like grasp affordances in 3d point clouds. In *Experimental Robotics*, pages 623–638. Springer, 2016.

[24] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova. Robot web tools: Efficient messaging for cloud robotics. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4530–4537. IEEE, 2015.

[25] S. Zhai and P. Milgram. Quantifying coordination in multiple dof movement and its application to evaluating 6 dof input devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 320–327. ACM Press/Addison-Wesley Publishing Co., 1998.

[26] M. Zucker, S. Joo, M. X. Grey, C. Rasmussen, E. Huang, M. Stilman, and A. Bobick. A general-purpose system for teleoperation of the drc-hubo humanoid robot. *Journal of Field Robotics*, 32(3):336–351, 2015.