Autonomous Robots
https://doi.org/10.1007/s10514-021-09983-8

Annotation scaffolds for manipulating articulated objects

Pablo Frank-Bolton[1] · Roxana Leontie[2] · Evan Drumwright[3] · Rahul Simha[2]

Received: 1 October 2018 / Accepted: 9 May 2021

Abstract
We present and evaluate an approach for human-in-the-loop specification of robot–object interactions. Our method is based on the idea of model annotation: the addition of simple cues to an underlying object model to delineate a complex task. The goal is to explore simplified CAD-like interfaces to permit novice users to describe and annotate manipulation tasks that are then carried out by a robot. The constructed models meet precision requirements for modeling a variety of joint types and kinematic chains, and can be re-used after their initial design. The approach is contrasted with teleoperation and tested with a user study. We found that untrained users can create object models whose structure can be readily used to initialize joint constraints to guide the user in designing successful end-effector trajectories. We see this approach as an alternative to direct teleoperation for cases where it is more natural or practical to store the action sequence with the object as the reference frame. The approach was evaluated using the PR2 robot platform.

Keywords Articulation modeling · Annotation · Human-in-the-loop · HCI

1 Introduction

High level robotics tasks, such as preparing a meal or cleaning a room, might require robots to interact with new or complicated objects in potentially unstructured environments. In particular, pick and place actions and the handling of articulated objects are typical tasks for a robotic system that is expected to operate in human-centric environments. These tasks are often, in themselves, composed of a sequence of low-level actions such as detecting objects, choosing and executing grasps, and deciding how to move held objects. These low-level actions are often the bottleneck to accomplishing more complex activities due to the sheer combinatorial variability of the cases that the robot might have to encounter.

Significant progress has been made in teaching robots to deal with manipulation challenges autonomously: examples of these are (Ciocarlie et al. 2014), where vision is used to recognize previously seen objects and grasp them based on physical appearance; (Sung et al. 2015), where manipulation behaviors are learned from human demonstrations; and (Kim and Sukhatme 2014), were behaviors are defined as a reaction to object features. Despite these advances, many of these low level tasks often prove difficult to solve without human participation.

An alternative to autonomous techniques for object interaction, such as the ones mentioned above, is having a human operator provide assistance to the robot in order to accomplish difficult tasks. While many circumstances might preclude a human from doing the task themselves (such as a dangerous environment, or being physically unable to do so), an operator can direct the actions of a robot using an effective interface. A human operator has the great advantage of being able to assist in the full perception-action loop, from interpreting noisy or obstructed sensory data, to making decisions and indicating ways to carry them out. In *teleoperation*, a person controls the actions of the robot using some type of remote control. This approach could be called *actor-relative*, since the motions are directed from the point of view of the robot (or the operator controlling it).

One issue with this approach is that the operator might get overburdened by attempting to simultaneously achieve the task outcomes (by cognitively managing micro-actions) while overcoming difficulties that arise from interacting with the environment through the robot interface and limited perception (Kent et al. 2017). We consider the approach of

✉ Pablo Frank-Bolton
pfrank@smith.edu

[1] Smith College, 10 Elm Street, Northampton MA 01063, USA

[2] The George Washington University, 800 22nd St NW, Suite 4000, Washington, DC 20052, USA

[3] Dextrous Robotics, 1350 Concourse ave, Memphis, TN 38104, USA

Published online: 04 July 2021

reducing the burden of operators by moving away from full teleoperation (as in bomb-disposal robots) to a low-demand interface where simple gestures or commands from the operator are translated into complex sequences of actions to be performed by the robot. This has the potential to reduce the amount of low-level controlling actions that the operator needs to think about, allowing them to focus instead on the high level goals. In fact, efforts have been made to construct robotic languages that link these high-level actions into complex activities (Tenorth et al. 2013). However, using such a language to indicate a detailed sequence of low-level actions can become cumbersome, especially when complex spatial understanding is required, as for example, when trying to program the sequence of joint rotations and translations to reach for an object and grasp it.

Instead of having the human grapple with a complex language to fully specify the step-by-step instructions to complete an interaction, an alternative is to capture these from a real or virtual demonstration. This gentler (on the human) approach focuses on generating and storing the most basic and relevant information that can be used to complete a task, leaving the low-level control to the robot platform instead. The human's cognitive burden can be reduced to making *annotations* layered on top of a visual representation of the scene. For example, the human can apply annotations to a manipulation task by adding simple markers on top of the visual field in order to create a set of *object-relative* instructions that can be directly carried out by a robot. Annotations have been used in a wide variety of object-related robotic tasks, such as shape reconstruction (Berger et al. 2017), rigid object grasping (Leeper et al. 2012), and learning articulated models (Sung et al. 2015).

In this work, we explore the potential and limitations of human annotation as a way for novices to program the steps required for a robot to *manipulate complex articulated objects*. The notion that novice users should be able to use the interface is important as robots start to be used as consumer devices and in home environments.

While complex objects might be difficult to completely describe (due to complicated geometry or complex internal mechanics), they usually have a reduced set of modes of operation related to their *function* (e.g., a drawer is slid open or closed, a window is rotated or slid open, a lid is unscrewed from a jar). In addition, a reduced set of simple actions is often all that is needed to allow the completion of a given task: specifying the bounding box of an object to avoid a collision; indicating a valid contact area for grasping; or defining the axis of rotation for a given revolute joint. The challenge lies in capturing the particulars of each articulated object's modes of operation and linking those to the actions a robot must follow to manipulate that object. Moreover, the (novice) operator should not need a lengthy training phase to construct the instructions for a robot–object interaction.

A properly designed annotation strategy (for specifying high-level interaction instructions as embedded cues inside the scene representation) should allow the operator to spend a minimum amount of effort to specify the annotations that will allow a task to be completed. In addition, these annotations should be multi-purpose and reusable so that the object, once annotated for a particular robot and task, can carry the annotations with it for successful manipulation of any instance of the object and robot. In addition to being applicable in multiple situations, the model should allow for simple extension to new situations.

In this work, we present a method that allows a user to quickly annotate instructions to manipulate a complex object by attaching a sequence of virtual actions to a model of the object. The goal of the model annotation interface is to have naive users be able to achieve robot manipulation tasks with little or no training, and to demonstrate sufficient precision and repeatability for their annotations to have generality and application in new situations.

We are motivated by recent results in the DARPA Robotics Challenge (DRC), summarized in Krotkov et al. (2016), which found, over the course of 3 years of competition that (1) human input is extremely useful in unstructured tasks, (2) using object models to prime interaction is comparatively better than the alternative, (3) there seems to be an optimal level of immersion for a human operator that is between full teleoperation and full abstraction through automation, and (4) the most successful interfaces incorporated multiple data streams (like point clouds, and object and robot models). The findings for the DRC are further discussed in Sect. 2.4.

Our method is composed of three steps: (1) building models of articulated objects (which we call *scaffolds*) guided by captured point clouds of these objects, (2) annotating those scaffolds with user-defined task frames, and (3) using these task frames to determine robot's grasping poses and trajectories to actuate the object's internal degrees of freedom. The annotations are performed by changing the configuration of scaffolds through the use of simple widgets and saving the intermediate steps.

This approach is validated through user testing which also shows that reusing these constructed models on new scenarios takes very little time and practically no training.

We envision this system as an approach that will allow untrained operators to *model once* and *reuse anywhere* to accomplish complex robotic actions with very little overhead. We believe it constitutes an attractive alternative to teleoperation for circumstances in which that approach is less viable, such as when live-remote control causes the point of view to be occluded, or when the manipulation is a complex one, requiring great precision or involving several steps. The other application scenario is for consumer robotics where end consumers wish to achieve a manipulation task through a simple

interface without having to learn the equivalent of 3D object design software.

In the following section we describe previous work focusing on annotating robot–object interactions. Section 3 contains the description of the methods used for specifying annotations for object modeling, including articulations. Section 4 contains a description of the user study performed to demonstrate the strengths of our approach. Results are presented in Sect. 5, and discussed in Sect. 6. We discuss the limitations of our approach in Sect. 7 and present our conclusions in Sect. 8. Finally, in Sect. 9, we discuss future work.

2 Related work

In this work, a simple GUI framework and graphical annotation scheme is employed to allow novice users to indicate the steps needed for a robot to interact with articulated objects. In particular, we wish to introduce a way to use human perceptual capabilities to sidestep the difficulty of estimating articulation models in the presence of noise, or in the case where demonstrations are not a viable option. In the following subsections, previous work is presented that address the tasks of inferring object model parameters from interactions, using human cues, and constructing annotation platforms based on human-centered design.

2.1 Extracting object parameters

Some studies have looked at specific solutions for inferring parameters in pick-and place-tasks (Jiang et al. 2012; Dragan and Srinivasa 2012), or inferring articulation parameters from human demonstrations and applying the obtained insights to robotic manipulation (Sturm et al. 2010; Sturm 2013; Katz et al. 2013; Hausman et al. 2015; Pillai et al. 2015; Huang et al. 2014).

Other work focuses on inferring possible object function from shape analysis. A survey of methods for inferring functionality from shape is presented in the work of Hu et al. (2018). In particular, the work of Hu et al. (2017) aims to predict mobility of parts from an input 3D model.

In contrast, our work uses a human in the loop and a simple annotation mechanism to bypass the need for inference. We instead use human experience to directly indicate a preferred mode of interaction. Having a human in the loop can help counteract the problems that automatic methods have when considering the imperfect nature of the input. For example, monocular images lack depth information while stereo images rely on using features that might be sparse. Point clouds allow the use of registered depth and color images, but often contain noisy or missing data that can negatively affect autonomous methods.

2.2 Human in the loop

The main goal of this paper is to examine the manner by which a human can effectively take part in the perception, planning and execution process. Prior work has focused on remote control, which may be accomplished through the use of a wide variety of interfaces (Boboc et al. 2012). These have changed considerably with the advent of *human-centered design*, and have evolved from joystick-and-button black boxes to force-feedback devices (Farkhatdinov et al. 2010), smart gloves (Ekvall and Kragic 2007), body-suits (Ramos et al. 2015), verbal instructions (Johnson-Roberson et al. 2011), and vision-based analysis of natural body motions (Lipton et al. 2018; Pollard et al. 2002; Fritsche et al. 2015; Ishiguro et al. 2017).

While attractive in terms of the natural interaction they afford, an immediate complication in using these methods is obtaining and learning to use the required equipment. In addition, mixed results from immersive teleoperation have been reported, resulting in efforts to reduce the cognitive load of the operators (Hart and Staveland 1988). In the work of Martins et al. (2015) a clear trade-off was detected: greater control and situational awareness came at the cost of possible cognitive overload and impaired performance. In contrast, we seek to use off-the-shelf sensing equipment as well as a traditional monitor-based GUI.

With regard to reducing the cognitive load on the user in monitor-based GUIs, Kent et al. (2017) found that simple constraining of motions was effective in reducing some of the cognitive effort. The familiarity and widespread use of smartphones and tablets has also brought attention to touch-based control. Singh et al. (2013) used a simple touch-based interface in order to reduce operator fatigue. Interfaces that use Monitor, Mouse, and Keyboard (MMK) are simple and cheap alternatives that have been applied to a wide variety of robotics applications such as robot navigation (Osentoski et al. 2010), grasping (Miller and Allen 2004; Sorokin et al. 2010; Sucan and Chitta 2013), or object manipulation (Leeper et al. 2012; Sung et al. 2015). Consistent with these, our model-creation tool employs constraining motions that allow quick specification of parameters using an MMK interface.

One important consideration is the level of human control during the interaction. In the work of Leeper et al. (2012), a variety of strategies involving different levels of human control were compared. They determined that shared workload approaches that pair high-level decisions by humans and low-level motion planning by the robot were able to achieve the best results. This is supported by previous work (Dragan and Srinivasa 2012; Hertkorn 2016), where collaborative robot–human workload generated better results than pure automation; or in the work by Balasubramanian et al. (2014), where a strategy involving human supervision surpassed the

quality of automatically suggested poses, especially if the purpose of the grasp was taken into account. With this in mind, the task specification in our work is based on high-level annotations whereas the low level motions are left to the automatic planner.

2.3 Simplified task specification

Our approach is designed with the idea that grasp suggestions are to be embedded in the simple object models and employed as a possible initial grasp point for the completion of a manipulation task. This section reviews work of a similar nature.

Instead of asking the operator to solve the whole manipulation problem, one option is to obtain the parameters of an action by inferring them from the object they are to affect. In the context of grasping, Ciocarlie et al. (2007) and Ciocarlie and Allen (2009) developed a technique called *Eigengrasps*, where low-dimensional grasp subspaces are computed that match the object shape. These, in turn, may be used for seeding interactive grasping. Some approaches are designated hybrid, since they use appearance features or 3D-model detection depending on the situation (Brook et al. 2011). Another procedure is to match and apply grasping templates to the sensed input (Herzog et al. 2012). One option is to detect known objects and execute a specific grasp linked to that particular instance or category of objects (Azad et al. 2007; Dang and Allen 2012). In the work by Miller et al. (2003), shape primitives are linked to the target object and used to seed appropriate contact-level grasping. In our work, we use the structure of the object models to prime the grasp position and let the human operator complete the initial placement.

Databases of grasping examples may be used as a starting point for automatically generating grasp hypotheses (Li and Pollard 2005). While these approaches are not fully automatic, they already contain important semantic information that might be germane to a multitude of context-dependent grasping tasks. In Dang and Allen (2012); Nikandrova and Kyrki (2015), shape and objective constraints are used to specify appropriate grasps. Large annotated databases (Goldfeder et al. 2009; Chang et al. 2015) exist that could be mined for data that could be applied to objects involved in the robot's task. These annotated objects might be hand-crafted using Computer Assisted Design software or CAD (such as Solidworks, or 3DsMax), or constructed by integrating features from repeated interactions (Huang et al. 2014; Sturm et al. 2010; Martín-Martín et al. 2016). They can be later refined and reused for different task instances or as a reference to inform on new events (Dang and Allen 2012). We built our own framework with the idea of (1) generating structured output that can be used to build such a database, and (2) using pre-made models from these databases to bootstrap our own model construction. Integrating these databases with the task specification described here constitutes future work for this project.

Some studies focus on creating interfaces that fulfill requirements of versatility and ease-of-use: on the one hand, they allow the annotation of raw data to solve multiple robotic tasks; on the other, the interface must be precise without demanding a large effort from the user. In Sorokin et al. (2010), crowdsourcing was used to segment, classify and evaluate 2D and 3D objects for grasping. Also for grasping, some work has focused on achieving grasp refinements based on appearance and tactile feedback (Hsiao et al. 2010), the detection and fitting to predefined models (Dang and Allen 2012), or a combination of both (Brook et al. 2011). One approach (Sung et al. 2015) uses crowd-sourcing of human manipulation annotations to gather information that may recognize affordances for objects that share similar features to the ones used in the manipulation trials. While that work concentrates on extracting possible affordances, in our work, we focus on attaching annotations for manipulation sequences that relate a specific object to a full task.

One popular tool for motion planning is Moveit! (Sucan and Chitta 2013). This tool can be used within the larger Robot Operating System (ROS) to perform automatic motion planning with collision avoidance. In the RViz project (Leeper et al. 2012), an interface was provided for humans to record robot manipulations. In this context, our work is aimed a providing a simple interface for novice users to specify object manipulation, which we further test with Moveit! and the PR2 robot.

2.4 DRC: lessons learned

In the DARPA Robotics Challenge (DRC) humanoid robots were tested in disaster response scenarios where human assistance could be limited by faulty communications. Teams created shared-autonomy systems that could complete a sequence of mobility and manipulation tasks (Fallon et al. 2015; Atkeson et al. 2015; Marion et al. 2017). Strategies were ranked according to task and sub-task completion ratio and timing, and the results were analyzed to extract recommendations for the design of future systems. In both the 2013–2014 (Yanco et al. 2015) and 2015 (Norton et al. 2017) analyses, the findings indicate the following with respect to complex object manipulation:

- Despite advances in AI research, having robots perform tasks in unstructured environments almost always requires human supervision.
- The most successful strategies made use of integrated data displays (point clouds, object and robot models, etc.), which allowed visualizing the scenario in a single display.

- No correlation was found between situational awareness (fraction of the environment that the operator can be aware of through the interface) and task speed, indicating no clear benefits from using interfaces that attempt to increase awareness (through the use of feedback-gloves or VR) over those consisting of the classic monitor-mouse-keyboard (MMK).
- Decreasing the amount of operator input needed to control the robot is desirable, (moving away from teleoperation) (Yanco et al. 2015). However, more user interaction is recommended for more complex manipulation tasks (Norton et al. 2017), suggesting the existence of an optimal region between teleoperation and fully abstracted user interaction.
- The number of teams (of those included in the analysis) using object-models to help with motion planning went from 25% in 2014 to 50% in 2015. Those teams were consistently among the most successful.
- Teams using autonomous pose estimation of objects by fitting shape templates performed worse than teams where the user participated in refining the objects pose (Norton et al. 2017).

One thing to note is that surprise challenges were offered to each team competing in the DRC. Several teams chose to skip the surprise challenges because they had no pre-computed motions or object models to deal with these challenges. This highlights the importance of having a strategy to quickly incorporate new object models and interaction routines. In Marion et al. (2017), an object model with linked constraints for motion planning is automatically fitted to the scene, and operators were able to refine the model poses when necessary. These objects, however, were limited to those present in the competition.

In the work by Fallon et al. (2015), object models constructed in a modified version of Unified Robot Description Format were used to indicate manipulation actions. In that system a combination of automatic model fitting and user refinements were used to indicate affordances in the environment, which were then acted on by the robot according to the results of their planning. Their experiences fit with the lessons learned mentioned above. In their design, the user provides an initial search region in which an automatic fitting algorithm would place one of three simple affordances. The human operator could do final adjustments and accept the fitting. The system is quite powerful in scenarios for which these three affordances are required, but might be severely challenged if surprise tasks (with new or more complex affordances) are presented. In our work, the focus is on the inclusion of an integrated interface for the creation of such complex objects from raw input clouds, with the fitting left to the user.

An integrated interface for model-construction and manipulation annotation, as we will describe in this article, could be very useful in extending these types of shared-autonomy system architectures.

2.5 Automatic methods and human-in-the-loop

The use of methods that involve a human operator, or Human-In-the-Loop (HIL) need not be mutually exclusive with autonomous ones. In the aforementioned studies, several of the automatic methods were developed by working with training data annotated by or performed with human guidance. We believe that the construction of such annotated models can seed the development of automatic algorithms that can solve complicated articulation challenges without the use of visual aids (Baum et al. 2017), or require a learning phase to operate complicated machinery from inferred object-relative interactions (Sung et al. 2015; Kappler et al. 2015).

In the following sections, we present an annotation strategy that allows the creation of simple object models with instructions that, when followed by a robot, can perform complicated manipulation tasks.

3 Methods for specifying interaction annotations

In this study, we describe a method to quickly record a set of virtual manipulation actions to accomplish a real robot–object interaction. We do this by annotating the motion of a virtual gripper with respect to the model of an object that has been previously constructed. The model annotations describing its joints are linked to the gripper's pose, thereby constraining its motion to the correct path.

3.1 Annotations for object modeling

As mentioned above, we wish to obtain an *object-relative* annotation that can be constructed incrementally and applied to multiple scenarios, including: object detection, grasping, and manipulation. Our approach uses the concept of *tracing*, where an original item is copied (to some desired precision) with the use of an overlay, such as when tracing the lines of a drawing on superimposed translucent paper. Our goal has been to develop the 3D equivalent of translucent paper-tracing using an inexpensive sensor and its data stream: a Kinect, and the popular Point Cloud Library (PCL) (Rusu and Cousins 2011). The Kinect is used to capture a point cloud of a scene which can be a snapshot from a single point of view, or an integrated scan of the full scene, and which itself can be accomplished using any number of methods including the Kinect-Fusion algorithm (Newcombe et al. 2011). We use PCL's version, called *KinFu*. At the moment, the point-cloud

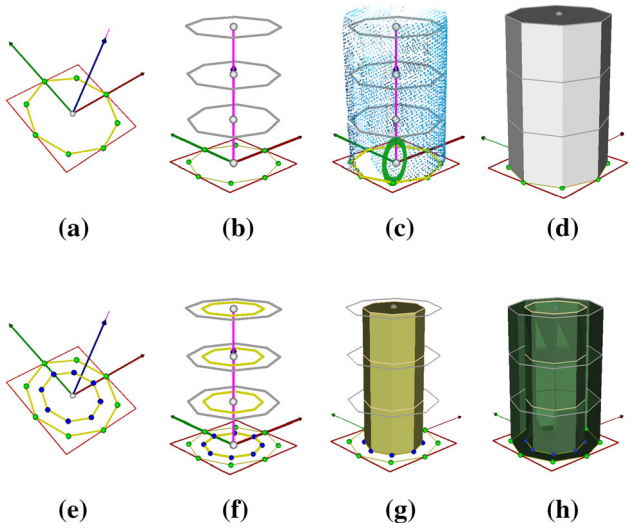


Fig. 1 Scaffolds: **a** single slice with green vertices and yellow contour; **b** full scaffold with one highlighted slice; **c** scaffold on top of its point cloud; **d** generated mesh for the enclosed volume; **e** slice with a hole contour defined by blue handles; **f** sequence of slices with hole contours; **g** mesh for the negative volume; **h** combined final mesh with the negative volume extracted from the external volume(Color figure online)

capture is carried out before users annotate but in the future can be integrated into the annotation procedure.

At this stage, the point cloud is loaded into our user interface and annotation tool, which we call the Point Cloud Prototyper (PCP). In PCP, the annotator can use our simple set of tools to perform various tasks depending on the annotation objective. If the object model is to be constructed for the first time, the annotator can use the tools to quickly build object representations we call *scaffolds*. If the objective is to record a manipulation sequence, a different set of tools are used to indicate the grasp point and the constrained motion of the parts. In the following sections, we provide a description of the scaffold construction process.

3.1.1 Scaffolds

Scaffolds are compact shape representations composed of a series of contours defined on a control plane or *slice* (highlighted in red for the base slices in Fig. 1).

They stem from a type of representation called swept-surfaces or generalized cylinders (Binford 1971), which use sequences of polygons or closed planar spline contours to describe a wide variety of solids. In our system, scaffolds are composed of closed polygonal contours defined by vertices (shown in green in Figure a). The base contour is highlighted in yellow for all the images in Fig. 1 (a single contour is shown in Fig. 1a), and highlighted in silver for the remaining contours of each scaffold shown in Fig. 1b–h. They are connected in a sequence to form the full scaffold (Fig. 1b). Scaffolds can be visualized as an overlay on top of point clouds (Fig. 1c) and can have subsequent slice vertices (or *handles*) connected to define a mesh that encloses a volume (Fig. 1d). The number and position of handles on each contour, and the placement of the slices that contain them, define the shape of the enclosed volume.

Scaffolds may be combined using Boolean operations to define complex shapes. In addition, a second set of *hole contours* in each slice (gold contour with blue handles in Fig. 1e), can be added and connected to define an *internal volume* (Fig. 1g) which can be removed from the external volumes to create cavities or internal spaces (Fig. 1h).

Slices and handles are interactive. Each slice can be independently translated, rotated, and scaled using an arrow and disk transform widget (the y-axis rotation disk for the first slice can be seen in Fig. 1c). Each handle may be dragged along its constraining slice. Figure 1a shows a highlighted slice with its local reference frame. When initially created, slices are placed in sequence along an axis of displacement or *sweep axis* (highlighted in Fig. 1 in magenta).

The goal of a scaffold is for the user to quickly go from a point cloud of a scene to a rigid or articulated object representation to better represent the structure of objects. The following section shows an example of a full object modeling sequence.

3.1.2 Modeling objects with scaffolds

In PCP we can work with point clouds obtained from a single point of view, or composed of the fusion of many such clouds.

A scanned point cloud from a single point of view may appear as a complete point-based representation of a scene (Fig. 2a) but it usually contains large gaps or artifacts (Fig. 2b). In PCP, we can load the cloud and segment the various objects of interest through the use of a variety of point-editing tools (Fig. 2c–e) such as using frustum or polygonal selection; or using cut, copy, or deletion actions of point selections.

The modeling process continues once we have an isolated object (Fig. 2f). All other parts of the scene may be hidden and stored for future use. For the case of the mug shown, we have split its points into a “cup” and a “handle” (Fig. 2g). The insertion of the scaffold is done by using a *crosshairs insertion widget* that requires placing the point of view (POV) of the modeling window along the desired *sweep-axis* of the inserted scaffold. In Fig. 2h we insert a scaffold (with regular polygon as predefined contours) along the cup’s longitudinal axis. The result of the insertion is a scaffold that encloses a point cloud (Fig. 2i). PCP automatically fits the scaffold to the underlying point cloud for some predefined cross-section shape (rectangle or regular polygon) by scaling the shape so no point is outside the scaffold. The number of slices and handles may be changed to generate the external volume of the cup (Fig. 2j).

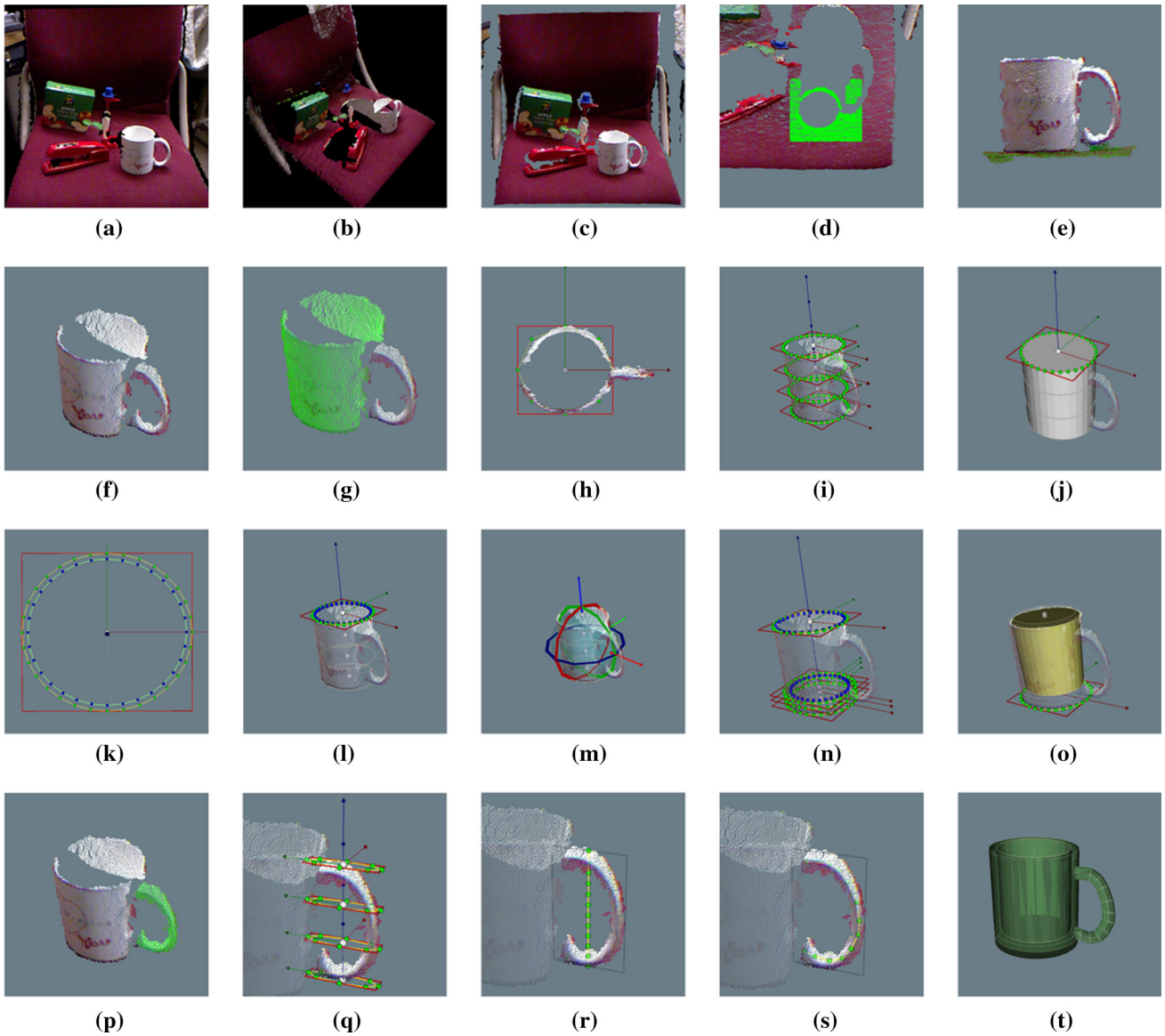


Fig. 2 Modeling process: **a**–**e** Isolating the point cloud of interest; **f**–**g** segmentation into a cup and handle; **h**–**i**the scaffold is inserted along the medial axis of the desired point cloud; **j** connecting the external contour handles results in an exterior (positive) volume; **k**–**n** insertion and placement of internal contours; **o** connecting the internal contour handles results in an interior (negative) volume; **p**–**q** insertion of the cup-handle scaffold; **r**–**s** use of the axis-drawing widget to draw the sweep-axis; **t** final union of meshes (the cup mesh has the negative volume removed)

In order to create the cavity in the cup, the user may insert an internal contour (Fig. 2k) that can be placed on any continuous sequence of slices (Fig. 2l). All external or internal contours can be modified (translated, rotated, and scaled) individually or as a whole (Fig. 2m). In the example, the top two slices have a *hole contour* (curves with blue vertices in Fig. 2n). Finally, these are connected into a negative volume (Fig. 2o).

Since some shapes have a curved sweep-axis, such as the mug handle (Fig. 2p), we have supplemented the normal automatic POV insertion (Fig. 2q) with an axis-drawing widget (Fig. 2r) that allows the user to reposition the slices along a path (Fig. 2s). The final object is the union of all difference meshes (Fig. 2t). The slice contours may also be automatically fitted to the underlying point clouds in a process we call *shrink-wrapping*.

Shrink wrapping is done by following these steps for each control plane (that holds each contour): (1) obtain the point cloud points near the plane (within the thin prism bisected by the plane); (2) fit a NURBS curve to the points projected on the contour plane; (3) reposition the contour vertices along the fitted curve without altering the initial vertex

order; (4) adjust the vertex positions to disallow topological irregularities when connecting corresponding vertices from neighboring contours.

For a more detailed explanation of the modeling, see (Frank-Bolton 2018)[1] where we concluded that when using PCP, novice users could reconstruct shapes quickly and with high accuracy. Novice users could quickly learn to use these shape proxies to represent a wide range of objects. The same study indicated that simple grasp widgets could be precisely placed using the GUI. This result agrees with previous findings (Balasubramanian et al. 2014) that indicate that human pose specification surpasses the quality of automatically suggested poses, especially if the purpose (functional objective) of the grasp is taken into account.

In the following section, we will explain how to attach articulation annotations to the constructed scaffolds.

3.2 Annotations for articulation

In this section we present an interactive method of specifying a task-frame for the manipulation of objects that contain simple joints. We make use of the scaffolds described in the previous section and attach additional annotations that allow the specification of articulations between the parts that they represent.

3.3 Frame of reference and control strategies

In addition to the creation of these annotated objects, it is important to define the reference frame for the annotation of actions and manipulations. One option is to define a task-frame for an object. The task-frame is a coordinate frame attached to a manipulated object which allows certain actions to be defined with respect to the object, rather than the world-frame. Several formalisms have been developed to specify this task frame in a way that may allow different control strategies. Mason (1981) described a task-frame formalism for active control of trajectories by employing force-sensing information. A survey of formalisms is presented in Bruyninckx and De Schutter (1996), together with a synthesis of control-oriented strategies (focusing on force-control), and task-planning strategies (focused on sequences of fine-motions). In the survey, strategies that use geometric features in the manipulated objects are categorized under the so-called *CAD-based planning* (Bruyninckx and De Schutter 1996). Our approach is based on this formalism, and is extended to ease the interactive specification of the object and the task.

[1] This paper is based on the first author's Ph.D. thesis. This paper would be the first peer-reviewed publication for this work.

3.4 Annotation scheme for articulations

We represent complex objects as an assembly of rigid links, each of which can be fixed or "jointed" with respect to the others. The construction of a complex shape consists of (1) point cloud segmentation into links, (2) the insertion of the scaffold for each link, and (3) the addition of joint constraints. Once an object model has been constructed, the linking of parts (scaffolds) into rigid assemblies or articulated chains is a simple matter of specifying the grouping of parts into links and placing the joint annotations.

For joint placement, we make use of the observation that *most of the articulated objects that the robot might manipulate have joints whose axes of motion align to their linked shapes*: doors and windows rotate around an axis that is perpendicular to the floor; a drawer slides along an axis that is perpendicular to its front and parallel to the ground; a lid unscrews around the container's central axis. This means that we can take advantage of the scaffold proxy itself to initialize the joint axis.

The way we implement this principle is by using the link's position to prime the pose of an articulation widget that we call the *joint mover*. This widget controls the desired motion of the link, and can later be used to constrain the movement of an interacting gripper (highlighted in cyan in Fig. 3). The joint axis that the *joint mover* will represent is defined as a point and vector, where the point is the origin of the coordinate frame, and the orientation is the z axis of that frame. The *joint mover* must be placed so that it lies on top of the main axis of motion (translation or rotation). If the *joint mover*'s pose is not ideal, the annotator can easily adjust its pose. For 1-DoF joints, it is sufficient to place the *joint mover* along the correct vector with the origin lying anywhere along that axis.

Figure 3 shows the steps to specify a rotation with the *joint mover*. The interaction can be recorded by the user by saving the configuration of the *joint mover* and gripper with respect to the initial scaffold placement. Waypoints are then exported from the sequence of gripper poses generated by the interaction.

3.4.1 Annotated object format

The final annotated object is a combination of individual scaffolds which may be rigidly linked or jointed. For each scaffold, we used a custom format where the pose of each slice is recorded with respect to the center of the base slice using the top 3×4 components of a homogeneous transformation matrix where the last row is always (0, 0, 0, 1) (unless otherwise specified all poses use this format), and each handle within its slice using polar coordinates with respect to the slice center.

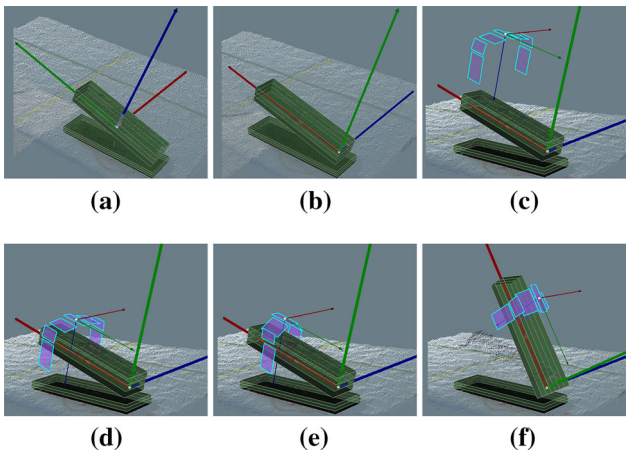


Fig. 3 Articulation of a stapler: **a** initial *joint mover* coordinate frame. The joint axis is defined by a point and vector, where the point is the origin of the coordinate frame, and the orientation is the z axis of that frame; **b** correctly placed *joint mover*, with the z-axis along the rotation axis; **c** gripper pre-grasp pose; **d** gripper on stapler top; **e**, **f** joint motion provokes the constrained gripper motion. The user may record a waypoint at any stage in the process

Rigid links are an assembly of scaffolds where their relative poses are recorded with respect to the base scaffold. Kinematic chains are sequences of links that can be saved in different configurations by also saving the sequence of relative transforms between them. In our current implementation, we save the motion of an articulated object as a sequence of chain configurations where each relative motion between links is controlled by the user's interaction with the *joint mover*. The single assemblies or kinematic chains contain all the information necessary in order to be exported to various other formats, for example the Unified Robot Description Format (Sucan last accessed: March 2019) as well as the one used in the Graspit! platform (Miller et al. 2003), which is similar to the Virtual Reality Modeling Language (VRML)

3.5 Articulation annotations for robot interaction

An additional annotation that can be informed by the object's shape (scaffold) is the motion of a gripper that manipulates it. Here, the task is to get the robot to manipulate the mechanism from one joint configuration to another. While the precise grip force and orientation can be refined automatically, the high level decision of where to grasp and how to actuate a joint can be left to the operator.

In this work, we make use of the scaffold structure and a simple set of tools to (1) place a virtual gripper, and (2) specify its motion by recording waypoints to reconfigure an articulated object. The motion is easy to constrain once the joint is engaged to cause a change in the object's configuration.

This procedure is shown in Fig. 3e, f, which consists of the following steps:

1. Select the joint mover widget for the first joint to reconfigure (stapler top in Fig. 3e).
2. Since the joint motion is constrained by the widget, a valid motion is achieved by rotating or translating the widget along the desired constraining axis (blue axis rotation in Fig. 3e).
3. As explained in Sect. 3.2, this will cause a constrained motion of the attached gripper widget and the linked elements down the articulation (gripper in Fig. 3f).
4. A waypoint is recorded by storing the configuration of the articulation chain and the connected gripper.

When evaluating our approach we used a grasp evaluation mechanism to test the validity of a candidate grasp. This mechanism uses the *GraspIt!* API (Buehler 2015). It computes the volume and ϵ-distance of the Grasp Wrench Space obtained from the chosen grasp. We use these two metrics to compare and contrast grasps obtained from different users and approaches.

In order to annotate more complex objects, like a 3-link chain with two revolute joints (see Fig. 5), or one link with a multi-degree-of-freedom (multi-DOF) joint, the steps are the same: (1) create the links by inserting scaffolds for each one; (2) indicate the link order within the articulation chain; (3) activate and place the *joint mover* widget for each link in the chain; and (4) indicate the motion using the steps discussed above. Since complex joints can be created by composing revolute and prismatic ones, a multi-DOF joint is represented by a single joint mover widget (for a joint that translates and rotates along its z-axis), or by multiple superimposed joint-mover widgets placed with z-axis in different directions.

To summarize, the whole process consists of the following steps: (1) construct an articulation model based on scaffolds, (2) annotate the joint mover poses (primed by the shape itself), and (3) specify the sequence of Cartesian-space waypoints that the object should follow. This causes the virtual gripper to follow a constrained path that is then exported to the actual robot for execution.

4 Experimental design

We carried out a user study to assess the ease of use, precision level, and quality of our approach when performed by untrained human subjects. We created a set of scaffolds based on actual objects and had 10 novice users annotate a manipulation virtually by using our object-relative waypoints under two scenarios: *articulation annotation* and *unconstrained gripper*. These annotations were then exported to be executed by the PR2 robot. As an experimental control, we had

10 novice users perform articulation tasks using teleoperation, and we then compared the results against those obtained though our approach.

4.1 Robotic platform and control strategy

For this project we used the PR2 robot platform and a position based control strategy. The trajectories are generated from the poses of the user-provided waypoints, and the PR2 native PID (proportional integral derivative) controller determines the torque, velocity, and acceleration necessary to execute it. We use the standard gains for the PR2. We are able to use this control schema successfully for this application due to the actuators in the PR2, which are mechanically compliant.

The PR2 robot motion is effected using Open Motion Planning Library (OMPL) and Kinematics and Dynamics Library (KDL). We used KDL to generate IK solutions for the end effector waypoints and final configuration in Cartesian space. We then used OMPL to generate spline-based trajectories between the resulting joint-space configurations.

Due to the natural stochasticity of the OMPL planners, it is desirable to specify paths at a finer resolution (Rovida et al. 2017). We performed the interpolation by subdividing the waypoint-to-waypoint path position (to a maximum of 10 cm) using simple linear interpolation, and the orientations (to a maximum of 15°) by using spherical linear interpolation, also known as slerp (Shoemake 1985). The constraints for the motion are set so that every step (after interpolation) is covered in 4.0 s and must end with zero acceleration and velocity.

While this added computation extended the execution times, it minimized the effects of the planning and execution control strategy on the results, thus permitting a better evaluation of the user-annotated motions.

4.2 Measurements

We measured the precision of the interactions by comparing actual link motions with respect to desired ones and accumulated translational and rotational errors. Baselines for measurements were determined by fixing a harness between the robot and the table that the objects were placed on. The door/window object was affixed to the harness to keep it from moving. We measured linear distances (with respect to the fixed table) using calipers and angles using a combination of angle finders and marks added to the objects for this specific purpose. We also measured the time it took to complete each full articulation annotation.

We evaluated the quality of the resulting robot interaction by assigning four levels to the complete manipulation, including grasping and motion. The four levels are:

- *fail* There is at least one major collision, or the grasp misses completely.
- *incomplete* The full task is not completed due to the object slipping out of the grasp or critical waypoints being skipped for exceeding the robot's reach.
- *rough* There is at least one minor collision, or the end-effector motion is misaligned with the degree of freedom of the articulation so that it displaces the whole object, rather than the intended link.
- *good* There are no major or minor collisions, the task is completed, and only the desired links are moved.

Lastly, we evaluated the reuse of *articulation annotation* using two steps: In the *initial annotation stage*, the annotator uses the scaffold assembly to annotate grasps and joints, and proceeds to record the sequence of steps (joint motions) to achieve a target configuration. This is then exported to be executed by the robot. In the *reuse stage*, a previously constructed model is applied to a new scene. To do this, a new cloud is obtained with the original object having been placed in a new location and configuration. The annotator must then follow the next steps: (1) insert the existing model and place it in the right spot; (2) adjust its starting configuration and desired motions. For the reuse stage, we measure the time needed to annotate, and the resulting precision and quality of the interaction (as described above).

4.3 Test objects

We constructed a set of simple physical devices for the robot to interact with. In addition, we included a set of every-day objects with various types of articulations and articulation chains. These were represented by models whose joint pose parameters are described in Table 1 and shown in Fig. 4.

In order to minimize the effect of modeling inaccuracy on interaction precision (just for the articulation evaluation), the object scaffold models were created beforehand by an expert. In our current implementation, the waypoints are recorded in Cartesian space (6 DoF poses for the end effector with respect to the object) and are exported to be executed by the controller. This approach was used for all trials and therefore, the results may vary given a different planning or execution control strategy.

4.4 Constrained versus unconstrained gripper motion

Under the *articulation annotation* approach, users were asked to record a pre-grasp, and grasp locations as well as a sequence of *joint mover* configuration changes that caused the gripper to move in a constrained way. Figure 5 shows an example of the *articulation annotation* approach using PCP,

Table 1 Description of the objects used in the trial

Object	Joint-type	Joint-definition	Interpolation method	Figure
pwood	Prismatic 1-DoF joint	Defined by its axis (using a point and vector) and the displacement magnitude from its initial location	For inserting intermediate points, simple linear interpolation was used	4a
rwood	Revolute 1-DoF joint	Defined by its axis and the magnitude of revolution. This joint has a maximum of 35° rotation	Simple linear interpolation between angles every 15°	4b
prwood	Cylindrical (prismatic and revolute) 2-DoF joint	Defined by its axis of motion (translation and revolution), and the magnitude of revolution and of prismatic displacement	When needed, we interpolated in the same way as with the *pwood* object	4f
bwood	Ball joint or 3-DoF revolute joint	Defined by the position of the center of the spherical joint, the rotation of the second link (the stick in the figure) from the canonical starting position (pointing straight up). This joint has a maximum deviation of 25-degrees from the vertical	Spherical linear interpolation (Shoemake 1985) was employed whenever two ball-joint poses differed in more than 15°	4i
drawer	Prismatic 1-DoF joint	Defined by its axis of displacement and the magnitude of the prismatic motion	Same interpolation as *pwood*	4c
stapler	Revolute 1-DoF joint	Defined by its axis and the magnitude of revolution. This joint has a maximum of 180° rotation	Same interpolation as *rwood*	4d
tmug	Screw 1-DoF joint	Defined by its axis of motion and the magnitude of the relative rotation. The screw pitch is hardcoded, which is something that could also be specified by a user, but is beyond the scope of this paper	Same interpolation as *rwood*	4e
door	Kinematic chain of two 1-DoF revolute joints	Defined by a combination of the configuration of two joints (same as with *rwood*), with a rigid transformation between the first joint (door), and the second (handle)	Independent interpolation (single joint articulation per motion) and in the same way as with *rwood*	4g
window	Kinematic chain of two 1-DoF revolute joints	This is similar to the *door* chain, but with a different revolution axis for the first joint (parallel to the ground instead of perpendicular)	Same interpolation as for the *door*	4h
tripod	Kinematic chain of a cylindrical 2-DoF joint with a revolute 1-DoF joint	Defined by the pose of the first (cylindrical) joint (defined by its axis of motion), and by the translation and rotation magnitudes of all three joints. The revolute joint's axis of revolution is fixed with respect to the cylindrical one	Interpolation was performed independently similar to that of the *door*	4j

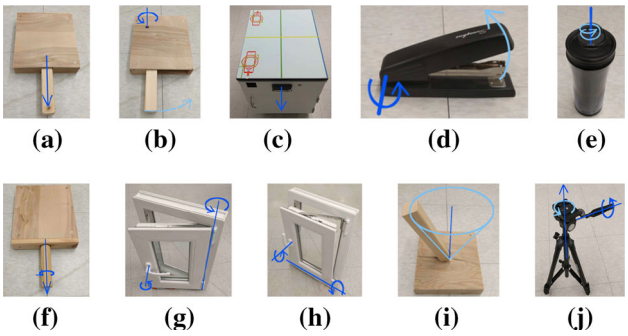


Fig. 4 Objects with general joint characteristics highlighted in blue: **a** artificial prismatic; **b** artificial revolute; **c** drawer; **d** stapler; **e** travel mug; **f** artificial cylindrical; **g** door; **h** window; **i** artificial ball; **j** tripod (Color figure online)

as well as the equivalent moments during the execution of the PR2.

We evaluated the constrained articulation approach using a matched pairs experiment. For the control, users annotated a gripper manipulation path in a manner similar to the manipulations evaluated in (Frank-Bolton 2018). In this approach, which we call *unconstrained gripper*, users use PCP to mark the path by using a sequence of gripper pose waypoints (by eyeball estimation only), with no joint motion to constrain the object's path. For the experimental group, the same group of users would then use the *articulation annotation* approach explained above. This was done to verify the effect of constrained motion on the annotation, while using the same GUI. Figure 6 shows a diagram for each approach.

For both our *articulation annotation*, and *unconstrained gripper* approaches, users were taught to move the virtual gripper and joint widgets (which took 2 min) and asked to complete a sequence of actions for each approach. We then recorded how long it took to annotate the motion, and the precision and success rate of the specified manipulation when executed by the PR2. These tests were carried out on four objects (*drawer*, *prwood*, *bwood*, *door*), with a combined total of six joints: prismatic (*drawer*); cylindrical (prismatic and revolute for *prwood*); ball (*bwood*), and two revolute (*door*). For the case of the *door*, extreme misalignment of the gripper motion with the degrees of freedom of the articulations resulted in the gripper slipping out, or the motion sequence stopping (due to the robot compliance).

4.5 Teleoperation

To compare our approach with an alternative teleoperation method, we had 10 novice users perform articulation tasks using direct PR2 teleoperation. This was done using the motion planning plug-in in the RViz GUI, which lets a user set waypoints (in Cartesian space) to control the motions of the PR2. This allowed us to contrast an *actor-relative* approach

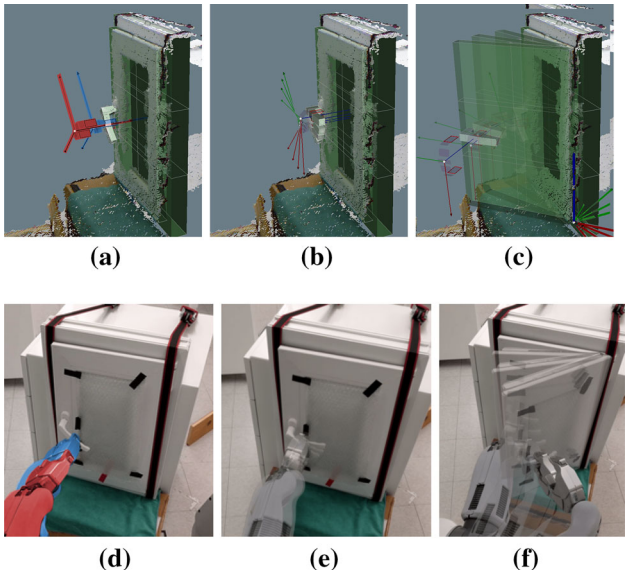


Fig. 5 Articulation Annotation for the action of rotating a handle and opening the door. The steps annotated in PCP are shown in **a**–**c** while the execution of the same steps is shown in **d**–**f**. The *pre-pose* (red) and *grasp-pose* (blue) are shown in (**a**) and (**d**). The joint mover change for the handle is shown in (**b**) and (**e**); The joint mover change for the door is shown in the bottom right corner of (**c**), and executed in (**f**) (Color figure online)

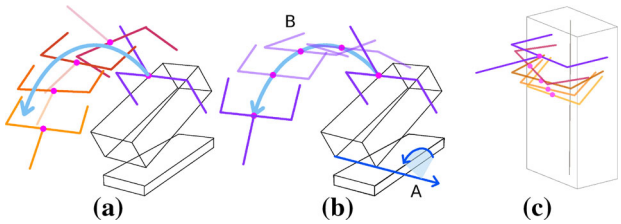


Fig. 6 Matched pairs experiment using PCP: **a** we use *unconstrained gripper* as the control group, where the user must move the gripper manually from waypoint to waypoint while attempting to visually match a desired motion (blue arrow); **b** we use *articulation annotation* as the experimental group, where changing the mover configuration **A** causes the constrained motion of the gripper **B** to a new pose (along the desired arc). **c** shows an example drift of the gripper poses relative to the rotated link for the *unconstrained gripper* (drifting orange tones); the *articulation annotation* gripper pose (purple) remains fixed with respect to the link (Color figure online)

(teleoperation) with our *object-relative* strategy. In order to ensure the best possible conditions for teleoperation, users received a 15 min training session that familiarized them with the interface; once the trials were performed, several attempts were completed for each task. In addition, teleoperation was performed in the presence of the robot and its workspace, and users were allowed to approach and visually verify their positioning. This increased the level of accuracy and spatial awareness for the operators. This situation is unusual in teleoperation tasks, but it allowed us to contrast our approach against teleoperation when performed in very favorable con-

ditions. These tests were performed on two objects (*prwood* and *rwood*) with a combined total of three joints (cylindrical, or prismatic and revolute; and revolute).

Finally, to provide an additional baseline for comparison, an expert in the use of both the PCP virtual annotation and the RViz-teleoperation GUI performed the same tests on the complete set of objects.

4.6 Data analysis

We used the *stats* package in R (R-Core-Team 2017) to analyze our data. To determine statistical difference between approaches, we employed the Kruskal-Wallis test (Kruskal and Wallis 1952) with post-hoc pairwise Chi-squared tests with Bonferroni correction (Field et al. 2012) for multi-way comparisons.

5 Results

5.1 Annotated articulation

Figure 7 illustrates the relative durations of modeling, articulation annotation and its execution, and reuse annotation and its execution. For the design of the articulation experiments we prioritized accuracy over speed when creating the object scaffolds.

The figure shows that after the initial modeling is performed, annotating and executing manipulation takes little effort to complete, and so does the reuse and its execution. The modeling, manipulation and reuse stages are performed (and timed) using PCP. The execute and re-execute phases are performed (and timed) using OMPL in MoveIt!, which is used to perform the annotated motions on the PR2. The *window* object is a special case because it has two different ways of being opened. When turning the handle half way, it can open as a door (Fig. 4g). If the handle is turned all the way up, it can open as an awning window (Fig. 4h), which in this case opens at the top. The scaffolds and *joint mover* annotations for the two modes of use were placed when constructing the first object (*door*), causing a slight increase in the modeling and annotation time. On the other hand, for the annotation of the interaction with the *window*, no modeling, and very little manipulation was required. This means that a modeled object with multiple uses requires no more than a single modeling but can be annotated for multiple interactions.

5.2 User study

Figure 8 shows the results for *articulation annotation* (art), *unconstrained gripper* (eye), and teleoperation (tele). As seen in the figure, *articulation annotation* is significantly

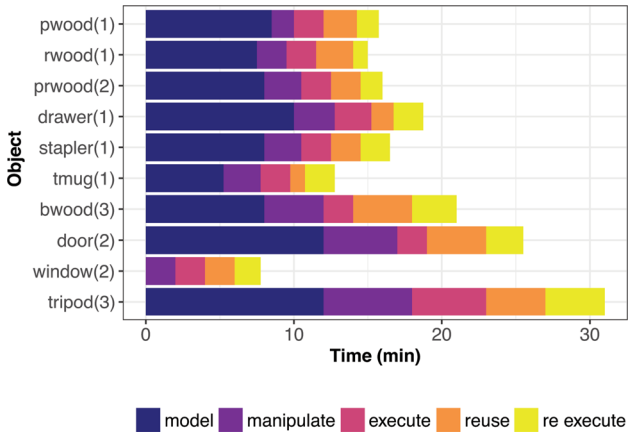


Fig. 7 Duration of stages of annotation for an expert user. The number of configuration changes is indicated in parenthesis (Color figure online)

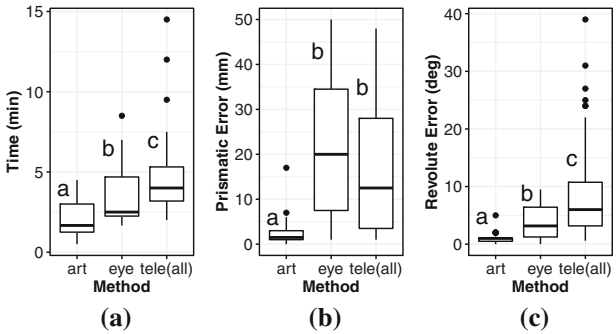


Fig. 8 Manipulation times (**a**) prismatic errors (**b**), and revolute errors (**c**) for each method. All attempts are included for *teleoperation*

faster and more accurate than both *unconstrained gripper* and *teleoperation*.

In addition, the quality distribution for each approach can be seen in Fig. 9. Even the best teleoperation attempts do not reach the quality of the *articulation annotation* approach. In terms of the initial placement of the *joint mover*, out of the six joints that required a change, only two (the door handle and the door itself) required an adjustment so that the virtual (*joint mover*) axis would match the real joint axis. These two joints are those that cause the gripper to follow an arc.

We also compared speed and accuracy when reusing annotations (i.e. in contrast to constructing them from scratch).

As can be seen in Fig. 10, the *reuse annotation stage* is significantly faster than the *initial annotation stage* with acceptable errors.

The qualities for both stages were identical, with all attempts except one marked as having *good* quality, and a single *rough* interaction under the *reuse stage*. Since the expert and novice users performed similarly, the above results hold even when including the expert to the user study results.

Lastly, when the expert user attempted all ten shapes under the three approaches, the overall pattern is maintained. Figure

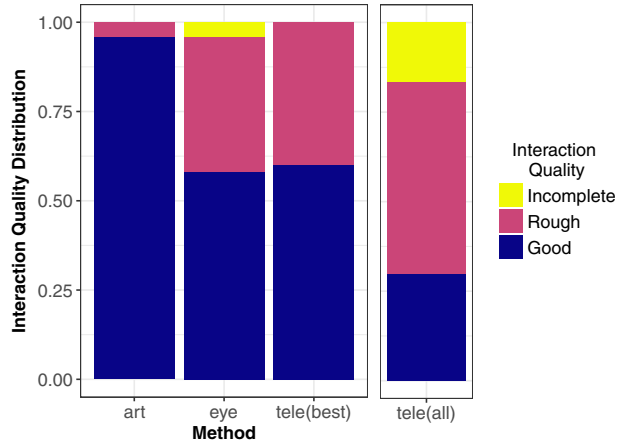


Fig. 9 Comparison of quality distributions including only best attempts (best) and all attempts (all) for teleoperation

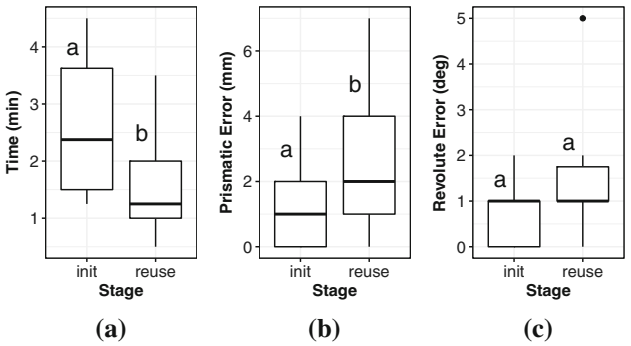


Fig. 10 Manipulation times (**a**) prismatic errors (**b**), and revolute errors (**c**) for the initial (init) and reuse (reuse) stages

11 shows the time as well as the prismatic and revolute errors for each approach. Figure 12 shows the quality distribution, which was also very similar to that of the novice users. While in general, faster times and lower errors were obtained compared to novices, the qualities were overall slightly lower than for novice users because of the inclusion of the additional objects, in particular, the *travel mug*, the *window*, and the *tripod*. Each of these had a particular difficulty with respect to manipulation:

- the travel mug had a cap that was almost as wide as the gripper's maximum aperture;
- the window required large forces when opening as an awning window. The PR2, being compliant, did not always complete the full motion as annotated by the user;
- the tripod was an articulation chain of degree three. Each joint change caused all outboard links to change locations. Since each joint change required a new grasp to be employed, this exacerbated any inaccuracies of the annotated articulation. On the other hand, as mentioned in the discussion section, this is an example of why the reuse capability is powerful.

For the expert *articulation annotation* we noted that out of the 15 joints that had to be annotated for all the objects (10), the joint mover had to be adjusted only for six of them: those causing the gripper to follow an arc. These were the *rpwood* object, the *stapler*, the *door* and its handle, and the *window* and its handle.

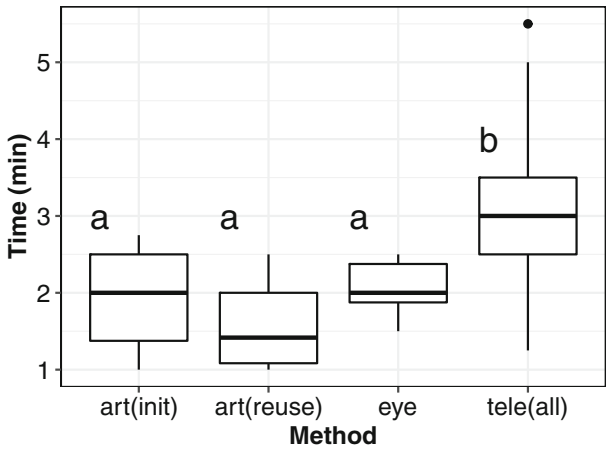
6 Discussion

We found that the scaffold structure itself provides an initial joint placement for links that is often immediately usable or that requires minimum effort to adjust. This follows from the structural properties of manufactured objects (where there is often an alignment of surfaces to joint axes). For the cases where the joint mover placement was not initially correct (in 6 out of 15 joints for the whole set), a single displacement or right angle rotation (trivial to apply with our approach) proved sufficient (Fig. 3a, b). All of these motions corresponded to actions that caused the gripper to follow an arc.

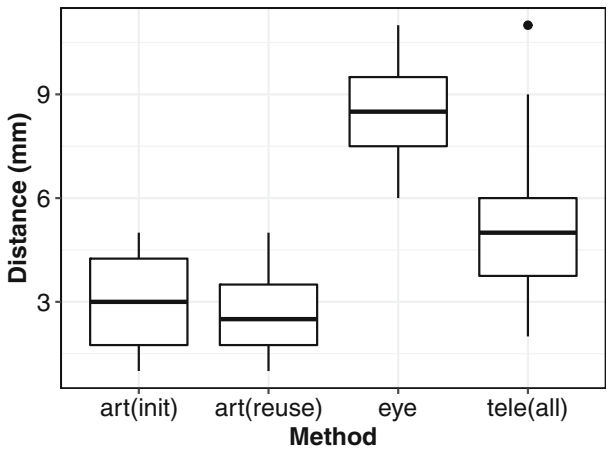
The object modeling stage required little time for constructing each of the chosen set of objects. Once these models were created, very little time was needed ($\approx$ 2 min) to annotate a manipulation per joint motion. Strikingly, novice users performed the annotation in a very similar amount of time as the expert (as can be seen from the time plots of Figs. 8a, 11a), indicating an interface that is simple to learn.

Reuse times were shorter than the initial annotations since, in most cases, only the placement of the object in the scene was necessary. In addition, it was clear from the user study that the constrained *articulation annotation* (moving the part would cause a motion on the gripper) helped shorten annotation times when compared to *unconstrained gripper* annotation (moving the gripper directly). The case of the *window* object annotation shows that, for any object whose model is available, the whole annotation and execution sequence can take fewer than 5 min (it took 8 min for both the initial and reuse stages, each with two joint manipulations). These times are comparable to the times spent solving manipulation sub-tasks in the DRC (Norton et al. 2017).

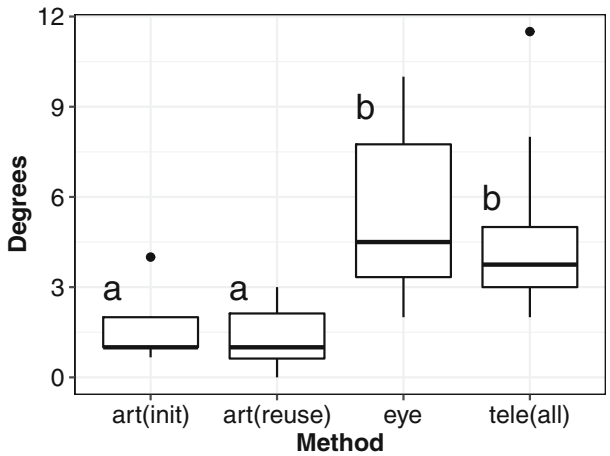
Prismatic motion for the *articulation annotation* approach was significantly more accurate than teleoperation and *unconstrained gripper*. The object-relative approach helps maintain precision and reduce the application of force against joint constraints during linear displacements. This was seen in the PR2 robot action when attempting to move an object: the moved link would get stuck along the chosen path and the gripper would slip out or the compliance mechanism would prevent the whole motion. This also indicates that *unconstrained gripper* annotation, with no constraints for motion



(a)



(b)



(c)

Fig. 11 Manipulation times (**a**) prismatic errors (**b**) (there were insufficient points to determine significant differences), and revolute errors (**c**) for the expert

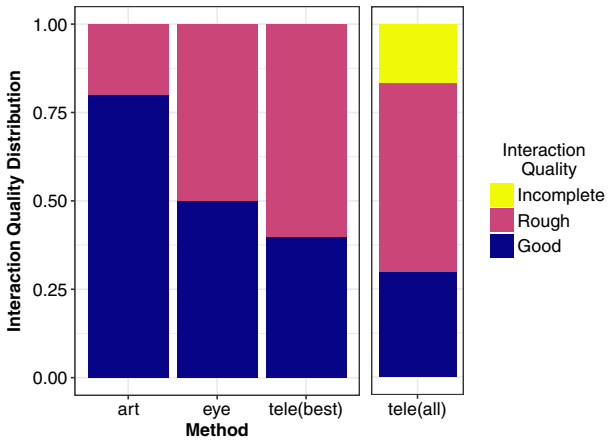


Fig. 12 Quality for the expert attempts on all objects including only best attempts (best) and all attempts (all) for teleoperation

or feedback for distance, is insufficient for precise distance estimation, which supports previous work indicating the difficulties of perceiving precise relative positions of objects in 3D (St. John et al. 2001; Tory et al. 2006).

The strongest benefits were noted with respect to the precision of manipulating a mechanism with revolute joints, where *articulation annotation* was significantly more precise than teleoperation. With no constraints, teleoperation of roll-rotations of the gripper were hard to specify because of the difficulty of aligning the gripper and joint axes of rotation. This problem does not arise in the *object-relative* approach, where joint motion happens with respect to the annotated axes of motion. For *articulation annotation*, all revolute motions were precise, whereas for *unconstrained gripper*, motions that required both the position and orientation to change (motion along an arc), obtained lower manipulation quality and precision.

The quality of grasps throughout the interaction was higher in the *articulation annotation* approach (for both expert and novice users). For *articulation annotation*, "rough" articulations stemmed from slight misalignment of the joint-mover's axis with the actual joint's axis. In one case, the interaction problems were due to the difficulty of the grasp (the gripper was almost the same size as the travel mug lid). On the other hand, "rough" articulations under teleoperation or *unconstrained gripper* would contend with more points of failure: in addition to the misalignment between the joint's degree of freedom and the application of force, each motion between waypoints would have a slightly different arc, causing the relative grip point to vary between trajectory segments.

Finally, the benefits of reusing the annotated scaffolds become apparent when considering that (1) objects with multiple uses can be modeled once and annotated several times (such as the window object in Fig. 7); and (2) there is a sig-

nificant time gain with a minimal precision loss when simply placing the annotated object into a new scene.

7 Limitations

The presented approach is most effective when used for objects with rigid links in relatively short articulation chains. For example, deformable bodies have not been considered in this study. The annotation scheme itself, while simplified, currently depends on sequences of actions that can be sometimes confusing or that take time to remember.

We noticed that, in some instances, annotators had to spend some time to regain the sense of direction and position of the indicated gesture. The interface itself could be made more intuitive by incorporating visual reference points (like the ground or fixed furniture) so that the user can quickly orient themselves.

Grasping the lid of the travel mug was difficult to annotate virtually before seeing the actual robot motions. This is due to the relative size of the gripper to the lid. While grasps were successfully placed by the expert, we decided to discard this item from the user study in order to focus on the annotation. While we do not directly address this issue, grasp-adjustment methods exist (Hsiao et al. 2010; Chebotar et al. 2016; Hogan et al. 2018) that may help deal with this low-level problem.

The worst quality in the *articulation annotation* approach was seen when actuating the last joint of the tripod: a 3-DOF kinematic chain where re-grasping was used. Small inaccuracies were propagated along the chain, causing visible deterioration of precision as the action chain was extended. This points to a limit in the length of action chains that can be specified in a single annotation. While problematic, this only means that complicated mechanical interactions that require re-grasping of the object would benefit from an intermediate scan and adjustment of the model's pose with respect to the actual object. This underlines the power of the reuse capability, which would allow the annotation of longer sequences of actions by interleaving saved actions with minor position adjustments provided by a user.

One other aspect to consider is that our results are tied to the chosen control strategy. A better control strategy could conceivably obtain even better results. In addition, the current implementation does not integrate the articulation annotation and the MoveIt! KDL solver, which might not find an IK solution.

Lastly, objects might have stiff joints, causing certain annotated motions to be rough despite being accurate according to geometry. Again, the use of adaptive methods during the execution of the manipulation could counter these issues.

8 Conclusions

This study introduced a simple method for rapidly annotating and executing manipulation tasks involving articulated objects. The approach was tested with the PR2 robot platform and showed a high success rate even for novice users. The constructed models could be quickly reused and refined for new circumstances and exhibited sufficient precision for manipulating articulated kinematic chains with up to three degrees of freedom.

This approach is an alternative for teleoperation for the cases when an object-relative approach is better suited to the challenge. These include cases where the POV for teleoperation gets occluded due to the operation, where precise joint rotations are required, or when action duplication is expected to occur. The object-relative manipulation using scaffolds constitutes simple annotation that product makers can add to their objects, indicating recommended modes of use.

The *object-relative* approach permits successful interaction with a multitude of different joint types and kinematic chains, which can be found in common household objects.

9 Future work

The current interaction and annotation mechanism described here appears to work well for articulated objects but is not required for scaffold construction and annotation. It is possible to replace mouse-based interactions with touch-based interaction using a tablet, or even through gestures captured from a mixed-reality head-set. Another extension is to include the point-capture stage within PCP. This could be done in conjunction with the use of the mixed-reality head-set in order to have a user scan an object and interact with the resulting point cloud through gestures to construct and annotate the virtual models.

We are also interested in extending PCP with a point-cloud-based joint estimation method. This would help further refine the annotation scheme and allow its initial position to be estimated by providing sequences of point clouds (snapshots) for each configuration. This could be provided by a human demonstrator or by the action of an autonomous agent. Additionally, cues for the insertion of possible joints could be obtained by using techniques that can predict part mobility from an input 3D model (Hu et al. 2017). An additional improvement might be gained from using extracted features from the scene to bootstrap the position estimation of the annotated scaffold, thus speeding the reuse of previously constructed models. The aforementioned object databases could be integrated with PCP to obtain these bootstrapping cues.

Lastly, PCP could be integrated with the IK solver in order to quickly adapt the annotations if no IK solution was found.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-021-09983-8.

References

Atkeson, C. G., Babu, B. P., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., DeDonato, M., Du, R., Feng, S., Franklin, P., Gennert, M., Graff, J. P., He, P., Jaeger, A., Kim, J., Knoedler, K., Li, L., Liu, C., Long, X., Padir, T., Polido, F., Tighe, G. G., & Xinjilefu, X. (2015). No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge. In *IEEE-RAS international conference on humanoid robots* (Vol, 2015-Decem, pp. 623–630). https://doi.org/10.1109/HUMANOIDS.2015.7363436.

Azad, P., Asfour, T., & Dillmann, R. (2007). Stereo-based 6d object localization for grasping with humanoid robot systems. In *IEEE/RSJ international conference on intelligent robots and systems, 2007. IROS 2007* (pp. 919–924). IEEE.

Balasubramanian, R., Xu, L., Brook, P. D., Smith, J. R., & Matsuoka, Y. (2014). Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *Springer Tracts in Advanced Robotics*, *95*(4), 477–500. https://doi.org/10.1007/978-3-319-03017-3_22.

Baum, M., Bernstein, M., Martin-Martin, R., Hofer, S., Kulick, J., Toussaint, M., et al. (2017). Opening a lockbox through physical exploration. *IEEE-RAS international conference on humanoid robots* (Vol. Part F1341, pp. 461–467). https://doi.org/10.1109/HUMANOIDS.2017.8246913.

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., et al. (2017). A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, *36*(1), 301–329. https://doi.org/10.1111/cgf.12802.

Binford, T. O. (1971). Visual perception by computer. In *Proceedings of the IEEE conference on systems and control*.

Boboc, R., Moga, H., & Talaba, D. (2012). A review of current applications in teleoperation of mobile robots. *Bulletin of the Transilvania University of Brasov Series I: Engineering Sciences*, *5*(54), 9–16.

Brook, P., Ciocarlie, M., & Hsiao, K. (2011). Collaborative grasp planning with multiple object representations. In *Robotics and automation (ICRA)* (pp. 2851–2858). https://doi.org/10.1109/ICRA.2011.5980490.

Bruyninckx, H., & De Schutter, J. (‘). Specification of force-controlled actions in the “task frame formalism”-a synthesis. *IEEE Transactions on Robotics and Automation, 12*(4), 581–589.

Buehler, J. E. (2015). *Capabilities in heterogeneous multi robot systems*. Ph.D. thesis, The University of New South Wales, Sydney, Sydney

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). ShapeNet: An information-rich 3D model repository. https://doi.org/10.1145/3005274.3005291. arXiv:1512.03012.

Chebotar, Y., Hausman, K., Su, Z., Sukhatme, G. S., & Schaal, S. (2016). Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1960–1966). IEEE.

Ciocarlie, M., Goldfeder, C., & Allen, P. (2007). Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Proceedings of the robotics: Science & systems 2007 workshop-sensing and adapting to the real world, electronically published*.

Ciocarlie, M., Hsiao, K., Jones, E. G., Chitta, S., Rusu, R. B., & Şucan, I. A. (2014). Towards reliable grasping and manipulation in household environments. *Springer Tracts in Advanced Robotics*, *79*, 241–252. https://doi.org/10.1007/978-3-642-28572-1_17.

Ciocarlie, M. T., & Allen, P. K. (2009). Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, *28*(7), 851–867.

Dang, H., & Allen, P. K. (2012). Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *2012 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1311–1317). IEEE.

Dragan, A. D., & Srinivasa, S. S. (2012). Assitive teleoperation for manipulation tasks. In *2012 7th ACM/IEEE international conference on human–robot interaction (HRI)* (pp. 123–124).

Ekvall, S., & Kragic, D. (2007). Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Proceedings—IEEE international conference on robotics and automation* (pp. 4715–4720). https://doi.org/10.1109/ROBOT.2007.364205.

Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., et al. (2015). An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics*, *32*(2), 229–254.

Farkhatdinov, I., Ryu, J. H., & An, J. (2010). A preliminary experimental study on haptic teleoperation of mobile robot with variable force feedback gain. In *Haptics symposium* (pp. 251–256). IEEE.

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. Thousand Oaks: Sage Publications.

Frank-Bolton, P. (2018). *Annotation scaffolds for robotics*. Ph.D. thesis, The George Washington University.

Fritsche, L., Unverzag, F., Peters, J., & Calandra, R. (2015). First-person tele-operation of a humanoid robot. In *2015 IEEE-RAS 15th international conference on humanoid robots (humanoids)* (pp. 997–1002). IEEE.

Goldfeder, C., Ciocarlie, M., Dang, H., & Allen, P. K. (2009). The columbia grasp database. In *Proceedings—IEEE international conference on robotics and automation* (pp. 1710–1716). https://doi.org/10.1109/ROBOT.2009.5152709.

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

Hausman, K., Niekum, S., Osentoski, S., & Sukhatme, G. S. (2015). Active articulation model estimation through interactive perception. In *Proceedings—IEEE International conference on robotics and automation* (Vol. 2015-June, pp. 3305–3312). https://doi.org/10.1109/ICRA.2015.7139655.

Hertkorn, K. (2016). *Shared grasping: A combination of telepresence and grasp planning*. Karlsruhe: KIT Scientific Publishing.

Herzog, A., Kalakrishnan, M., Righetti, L., Bohg, J., Pastor, P., & Schaal, S. (2012). Template-based exploration of grasp selection. In *2012 IEEE international conference on robotics and automation (ICRA)* (Vol. 60, No. 3). https://doi.org/10.1007/s10514-013-9366-8.

Hogan, F., Bauza, M., Canal, O., Donlon, E., & Rodriguez, A. (2018). Tactile regrasp: Grasp adjustments via simulated tactile transformations (pp. 2963–2970). https://doi.org/10.1109/IROS.2018.8593528.

Hsiao, K., Chitta, S., Ciocarlie, M., & Jones, E. G. (2010). Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ 2010 international conference on intelligent robots and systems, IROS 2010—conference proceedings* (pp. 1228–1235). https://doi.org/10.1109/IROS.2010.5649494.

Hu, R., Li, W., Van Kaick, O., Shamir, A., Zhang, H., & Huang, H. (2017). Learning to predict part mobility from a single static snapshot. *ACM Transactions on Graphics (TOG)*, *36*(6), 227.

Hu, R., Savva, M., & van Kaick, O. (2018). Functionality representations and applications for shape analysis. *Computer Graphics Forum, Wiley Online Library*, *37*, 603–624.

Huang, X., Walker, I., & Birchfield, S. (2014). Occlusion-aware multi-view reconstruction of articulated objects for manipulation. *Robotics and Autonomous Systems*, *62*(4), 497–505. https://doi.org/10.1016/j.robot.2013.12.006.

Ishiguro, Y., Kojima, K., Sugai, F., Nozawa, S., Kakiuchi, Y., Okada, K., & Inaba, M. (2017). Bipedal oriented whole body master-slave system for dynamic secured locomotion with lip safety constraints. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 376–382). IEEE.

Jiang, Y., Lim, M., Zheng, C., Saxena, A., Lim, M., & Saxena, A. (2012) Learning to place new objects. In *Proceedings—IEEE international conference on robotics and automation* (pp. 3088–3095). https://doi.org/10.1109/ICRA.2012.6224581. arXiv:1105.3107.

Johnson-Roberson, M., Bohg, J., Skantze, G., Gustafson, J., Carlson, R., Rasolzadeh, B., & Kragic, D. (2011). Enhanced visual scene understanding through human–robot dialog. In *IEEE International conference on intelligent robots and systems* (pp. 3342–3348). https://doi.org/10.1109/IROS.2011.6048219.

Kappler, D., Pastor, P., Kalakrishnan, M., Wuthrich, M., & Schaal, S. (2015). Data-driven online decision making for autonomous manipulation. In *Robotics: Science and systems XI*. https://doi.org/10.15607/RSS.2015.XI.044.

Katz, D., Kazemi, M., Bagnell, J. A., & Stentz, A. (2013). Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In *2013 IEEE international conference on robotics and automation (ICRA)* (pp. 5003–5010). IEEE.

Kent, D., Saldanha, C., & Chernova, S. (2017). A comparison of remote robot teleoperation interfaces for general object manipulation. In *Proceedings of the 2017 ACM/IEEE international conference on human–robot interaction* (pp. 371–379). ACM.

Kim, D. I., & Sukhatme, G. S. (2014). Semantic labeling of 3D point clouds with object affordance for robot manipulation. In *Proceedings—IEEE international conference on robotics and automation* (pp. 5578–5584). https://doi.org/10.1109/ICRA.2014.6907679.

Krotkov, E., Hackett, D., Jackel, L., Perschbacher, M., Pippine, J., Strauss, J., et al. (2016). The darpa robotics challenge finals: Results and perspectives: The darpa robotics challenge finals. *Journal of Field Robotics*,. https://doi.org/10.1002/rob.21683.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, *47*(260), 583–621.

Leeper, A. E., Hsiao, K., Ciocarlie, M., Takayama, L., & Gossow, D. (2012). Strategies for human-in-the-loop robotic grasping. In *Proceedings of the seventh annual ACM/IEEE international conference on human–robot interaction* (pp. 1–8). ACM.

Li, Y., & Pollard, N. S. (2005). A shape matching algorithm for synthesizing humanlike enveloping grasps. In *2005 5th IEEE-RAS international conference on humanoid robots* (pp. 442–449). IEEE.

Lipton, J. I., Fay, A. J., & Rus, D. (2018). Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*, *3*(1), 179–186.

Marion, P., Fallon, M., Deits, R., Valenzuela, A., Pérez D'Arpino, C., Izatt, G., et al. (2017). Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics*, *34*(2), 262–280.

Martín-Martín, R., Höfer, S., & Brock, O. (2016). An integrated approach to visual perception of articulated objects. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 5091–5097). https://doi.org/10.1109/ICRA.2016.7487714.

Martins, H., Oakley, I., & Ventura, R. (2015). Design and evaluation of a head-mounted display for immersive 3d teleoperation of field robots. *Robotica*, *33*(10), 2166–2185.

Mason, M. T. (1981). Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, *11*(6), 418–432.

Miller, A. T., & Allen, P. K. (2004). Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, *11*(4), 110–122.

Miller, A. T., Knoop, S., Christensen, H. I., & Allen, P. K. (2003). Automatic grasp planning using shape primitives. In *IEEE international conference on robotics and automation* (Vol. 2, pp 1824–1829). https://doi.org/10.1109/ROBOT.2003.1241860.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., & Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality (ISMAR)* (pp. 127–136). IEEE.

Nikandrova, E., & Kyrki, V. (2015). Category-based task specific grasping. *Robotics and Autonomous Systems*, *70*, 25–35.

Norton, A., Ober, W., Baraniecki, L., Mccann, E., Scholtz, J., Shane, D., et al. (2017). Analysis of human–robot interaction at the DARPA robotics challenge finals. *The International Journal of Robotics Research*, *36*, 1–42. https://doi.org/10.1177/0278364916688254.

Osentoski, S., Crick, C., Jay, G., & Jenkins, O. C. O. (2010). Crowdsourcing for closed loop control. In *Proceedings of the NIPS workshop on computational social science and the wisdom of crowds, NIPS* (pp. 4–7).

Pillai, S., Walter, M. R., & Teller, S. (2015). Learning articulated motions from visual demonstration . arXiv:1502.01659.

Pollard, N. S., Hodgins, J. K., Riley, M. J., & Atkeson, C. G. (2002). Adapting human motion for the control of a humanoid robot. In *IEEE international conference on robotics and automation, 2002. Proceedings. ICRA'02* (Vol. 2, pp. 1390–1397). IEEE.

R-Core-Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Ramos, J., Wang, A., & Kim, S. (2015). A balance feedback human machine interface for humanoid teleoperation in dynamic tasks. In *IEEE international conference on intelligent robots and systems* (Vol. 2015-Decem, pp. 4229–4235). https://doi.org/10.1109/IROS.2015.7353976.

Rovida, F., Crosby, M., Holz, D., Polydoros, A. S., Großmann, B., Petrick, R. P., & Krüger, V. (2017). Skiros—A skill-based robot control platform on top of ros. In *Robot operating system (ROS)* (pp. 121–160). Springer.

Rusu, R. B., & Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE international conference on robotics and automation (ICRA), Shanghai, China*.

Shoemake, K. (1985). Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics* (Vol. 19, pp. 245–254). ACM.

Singh, A., Seo, S. H., Hashish, Y., Nakane, M., Young, J. E., & Bunt, A. (2013). An interface for remote robotic manipulator control that reduces task load and fatigue. RO-MAN (pp. 738–743). IEEE.

Sorokin, A., Berenson, D., Srinivasa, S. S., & Hebert, M. (2010). People helping robots helping people: Crowdsourcing for grasping novel objects. In *IEEE/RSJ 2010 International conference on intelligent robots and systems, IROS 2010—conference proceedings* (pp. 2117–2122). https://doi.org/10.1109/IROS.2010.5650464.

St John, M., Cowen, M. B., Smallman, H. S., & Oonk, H. M. (2001). The use of 2d and 3d displays for shape-understanding versus relative-position tasks. *Human Factors*, *43*(1), 79–98.

Sturm, J. (2013). Learning kinematic models of articulated objects. *Springer Tracts in Advanced Robotics*, *89*, 65–111. https://doi.org/10.1007/978-3-642-37160-8_4.

Sturm, J., Jain, A., Stachniss, C., Kemp. C., & Burgard, W. (2010). Operating articulated objects based on experience. In *2010 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2739–2744). https://doi.org/10.1109/IROS.2010.5653813.

Sucan, I. A. Urdf. Retrieved March, 2019, from http://wiki.ros.org/urdf.

Sucan, I. A., & Chitta, S. (2013). Moveit! http://moveit.ros.org.

Sung, J., Jin, S. H., & Saxena, A. (2015). Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3D Pointclouds. https://doi.org/10.1007/978-3-319-60916-4_40. arXiv:1504.03071.

Tenorth, M., Perzylo, A., Lafrenz, R., & Beetz, M. (2013). The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments. In *IJCAI international joint conference on artificial intelligence* (pp. 3091–3095). https://doi.org/10.1109/ICRA.2012.6224812.

Tory, M., Kirkpatrick, A. E., Atkins, M. S., & Moller, T. (2006). Visualization task performance with 2d, 3d, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, *12*(1), 2–13.

Yanco, H. A., Norton, A., Ober, W., Shane, D., Skinner, A., & Vice, J. (2015). Analysis of human–robot interaction at the DARPA robotics challenge trials. *Journal of Field Robotics*, *32*(3), 420–444. https://doi.org/10.1002/rob.21568.

Pablo Frank-Bolton is a Lecturer in Computer Science at Smith University. He received his PhD in Computer Science from the George Washington University in 2018, a Masters in Computer Engineering from the Universidad Nacional Aut ónoma de México (2009), and B.S. in Telecommunication and Informatics from the Instituto Tecnológico Autónomo de México (2006). His primary research has focused on the use of human cues for improving cyber-physical systems. His other research interests include STEM Education and the applications of computer science to biology. He is the recipient of the Philip J. Amsterdam Graduate Teaching Award from the The George Washington University, as well as the Graduate Teaching Assistant of the Year from the GWU chapter of the ACM.

Roxana Leontie is Postdoc at George Washington University. Her Ph.D. thesis focused on using alternative feedback modalities, like proprioception and kinesthetic feedback, for robotic manipulation. Currently, Roxana is working on applying machine learning techniques to solve computer vision problems with biological applications. She is also very passionate about teaching and has covered several undergraduate classes: from teaching freshmen how to program Lego robots to autonomously explore a maze, helping sophomores understand the principles of software engineering, to teaching juniors how to apply database theory to practice.

Evan Drumwright is the CEO of Dextrous Robotics, a startup focused on building a robotic system for manipulating objects at superhuman speeds. He was a Senior Research Scientist at Toyota Research Institute in Los Altos, CA. He has been faculty in Computer Science at George Washington University and the University of Memphis and was a visiting researcher at both Open Source Robotics Foundation (OSRF) and Honda Research Institute. His work has been funded by NSF, ARL, Willow Garage, Hewlett Packard, and OSRF. Evan completed his Ph.D. in Computer Science from the University of Southern California (2007) and B.S. degrees in Mathematics and Computer Science from the University of Memphis (1999). He investigates multibody dynamics and its applications to robotic simulation and control. Evan wrote the multibody dynamics simulator Moby and is now a key contributor to the Drake robotics library.

Rahul Simha is Professor of Computer Science at The George Washington University, with an additional courtesy appointment in GW's school of education. He received his PhD in Computer Science from the University of Massachusetts in 1990. From 1990-2000 he was first Assistant Professor of Computer Science, then Associate Professor. He joined GW in 2000. His research interests include security and embedded systems, applied AI, educational technology, as well as interdisciplinary areas such as biocomplexity and STEM Education. He is the recipient of funding from various agencies such as the National Science Foundation, DARPA, and the Air Force Office of Scientific Research. He was a founding member and first Faculty Co-Director faculty lead of the university's reformulated teaching center, serving from 2010-2017, and has been involved in and has led numerous educational activities on campus including: co-teaching interdisciplinary STEM courses, co-directing interdisciplinary STEM programs, undergraduate research, senior projects, workshops for K-12 teachers, the Faculty Learning Community for junior faculty, curricular development, assessment, and program development. He is the recipient of several teaching awards including: 2010 CASE U.S. Professor of the Year for the District of Columbia, the engineering school's first Distinguished Teacher Award, and several departmental teaching awards.