An Object-Centric Paradigm for Robot Programming by Demonstration

Di-Wei Huang[1], Garrett E. Katz[1], Joshua D. Langsfeld[2], Hyuk Oh[3], Rodolphe J. Gentili[4], and James A. Reggia[1]

[1] Department of Computer Science, UMIACS,
dwh@cs.umd.edu, gkatz12@umd.edu, reggia@cs.umd.edu
[2] Department of Mechanical Engineering,
jdlangs@umd.edu
[3] Neuroscience and Cognitive Science Program,
hyukoh@umd.edu
[4] Department of Kinesiology, Graduate Program in Neuroscience and Cognitive Science, Maryland Robotics Center
rodolphe@umd.edu
University of Maryland, College Park, MD 20742, United States

Abstract. In robot programming by demonstration, we hypothesize that in a class of procedural tasks where the end goal primarily consists of the states of objects that are external to a task performer, one can significantly reduce complexity of a robot learner by not processing a human demonstrator's motions at all. In this class of tasks, object behaviors are far more critical than human behaviors. Based on this virtual demonstrator hypothesis, this paper presents a paradigm where a human demonstrates an object manipulation task in a simulated world without any of the human demonstrator's body parts being sensed by a robot learner. Based on the object movements alone, the robot learns to perform the same task in the physical world. These results provide strong support for the virtual demonstrator hypothesis.

Keywords: programming by demonstration · imitation learning · human-robot interactions

1 Introduction

Programming by demonstration [1, 2], or imitation learning, has emerged as a popular approach for coding behaviors of robots. Essentially, a robot learner is to observe a human perform a task (i.e., the demonstration), after which the robot is to autonomously reproduce the human's behavior to complete the same task (i.e., imitation). This general approach is widely viewed as an intuitive and promising alternative to conventional manual programming of robots. However, the optimal way in which a demonstrator should interact with an observing robot has not been conclusively determined, and may vary depending on the task and/or the learning robot. Common criteria for evaluating such interactions include minimizing the barriers for humans to demonstrate, minimizing

the burdens for a robot's learning algorithms to parse/analyze a demonstration (since imitation learning in itself is already a very difficult problem), and maximizing the types of tasks that are able to be demonstrated.

Existing interaction methods in which a robot learner observes a human demonstration can be classified into two classes. In the first class, the human demonstrator directly drives or manipulates the states of the robot's body, during which the robot stores the state trajectories for learning. The way in which the human drives the robot can be kinesthetic teaching, where a human physically pulls a robot's arms to complete a task [3, 4], or teleoperating, where the robot is driven by remotely located controls [5, 6]. The robot being driven can also exist in augmented realities [7] or virtual realities [8]. In the second class of interfaces, the human demonstrator performs the target task on his/her own without intervening in creating the states of the robot. The demonstration is captured by a camera and/or spatial markers that detect human motion, the results of which are passed to the robot for learning [9, 10]. This method has been used in conjunction with kinesthetic teaching [11]. However, in many cases human motions on their own are vague and do not clearly convey the intention of a demonstration. To remedy this, human motion data are usually furnished with additional information, such as the states of task-related objects [12–14], from which the robot can learn the relations between hand motions and object movements, and speech descriptions about the motions, from which the robot can learn the intentions of the motions from symbolic annotations [15].

Although many demonstration interfaces in the past have emphasized conveying the trajectories of a human's body parts, learning from human motions in this way poses a major burden to the learner as it involves either difficult image processing or requires special equipment for capturing a human's movement. Complex processing is then needed to identify actions from the recorded motions, and to infer the effects the actions have on the task-related objects. Coordinate transformations must also be learned between the demonstrator's and learner's frames of references. Further, since a human's and a robot's bodies are usually very different from each other, it is likely that the robot has to act differently from the human to cause the same effect on an object. In this case, learning from human motions can be ineffective or extremely difficult. To address these issues, we are exploring the following alternative scenario:

> *Virtual demonstrator hypothesis*: In many situations, effective imitation learning of a task can be achieved when the demonstrator is invisible.

In other words, in contrast to past work that deals with humans' physical demonstrations, we hypothesize that for many tasks where the goal is external to a performer's body, most critical information about a target procedure can be learned by ignoring the demonstrator and focusing instead primarily on the behaviors of the objects that are being manipulated, such as assembling a car or fixing a computer. By making the demonstrator a virtual presence, one qualitatively simplifies the motion tracking and understanding needed during learning and eliminates the coordinate transformation problems involved. This approach effectively shifts the problem of programming by demonstration from human

motion understanding (i.e., how to do) to object behavior understanding (i.e., what to do). While similar ideas have been considered in a few past studies [12, 13], they have either used an over-simplified environment (2D simulated world) or required special equipment. While we do not definitively prove the virtual demonstrator hypothesis in the current paper, we do provide a basic proof-of-concept that this hypothesis is valid in at least some situations and provide strong support for the hypothesis.

In the following, we describe an object-centric paradigm for programming by demonstration based on the virtual demonstrator hypothesis, where a demonstration is composed of object movements only while the human actions that cause these movements are not sensed by a robot learner. The work reported here is mainly about the interaction paradigm, as opposed to describing any specific cognitive robotics approaches to imitation learning. The latter will be the subject of a future paper.

2 Methods

Our object-centric programming by demonstration paradigm can be outlined as the following steps. (1) First, a task space is set up in a software-based simulated environment. (2) A human then demonstrate a task by manipulating objects in this environment via "invisible hands", the process of which is recorded as an object-only demonstration "video" along with information describing the state of the objects. (3) A robot agent processes the object-only video as well as the accompanying information, and forms a plan to reproduce the demonstration. The plan can then be rehearsed and refined in the same environment. (4) Finally, the plan is executed in the real world.

Our initial efforts have focused on a tabletop environment for bimanual object manipulation tasks, one that is being intensely studied such as in [16]. As a result, our software platform contains a 3D simulated world where an invisible demonstrator, a robot, a table, and a variety of task-related objects coexist. An example view of the simulated world as seen by the human demonstrator is given in Fig. 1, which contains a 3D environment and overlaid graphical user interface (GUI) controls (Fig. 1; top-right). The environment can be observed through a freely navigable perspective (Fig. 1; main window) and/or through the robot's perspective (Fig. 1; bottom-right). Real-time rigid body physics simulation is included based on the Bullet physics library[5].

2.1 Task Space Initialization

Task-related objects can be specified by writing XML files in conformance to a predefined XML schema. Our software platform loads such XML files and generate objects in the simulated environment accordingly. The XML schema defines XML elements for generating simple objects including blocks, cylinders, spheres,

[5] `http://bulletphysics.org`

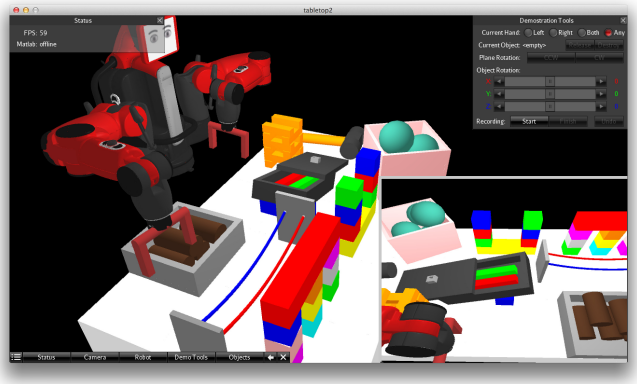

Fig. 1. An example view of the software simulation. The main screen shows a simulated environment containing a tabletop and a variety of objects (block, boxes, lids, strings, etc.), as they are seen by the human demonstrator. The avatar for a two-armed programmable robot is also embedded in the environment. The bottom-right corner shows the environment as it is seen by the robot. The top-right corner shows a sample GUI window. The user can manipulate the environment, including picking up, moving, rotating, and releasing objects, as well as changing the viewpoint and creating objects.

and containers (Fig. 2a–c), whose size, location, rotation, mass, color, etc., can be specified using XML attributes. Multiple simple objects can also be grouped hierarchically, via XML's hierarchical ordering, to form composite objects, whose center of mass can be specified. For example, the hammer shape in Fig. 2d is composed using three cylinders and a block. Further, more complex objects are included in the XML schema. A string object is simulated by connected solid links (Fig. 2e), which, when cut, will fall to the table (Fig. 2f). A container with a sliding lid can also be created (Fig. 2g). The lidded box can be positioned to become a container with a trap door (Fig. 2h).

2.2 Demonstration

A human demonstrator, while not embodied in the simulated environment, can manipulate objects using keyboard and mouse inputs through a GUI system, and therefore no special hardware is needed. For example, a user can virtually grasp an object by simply clicking on it, and move it by dragging the mouse cursor. A demonstration containing multiple object manipulations can be recorded as a video (a sequence of visual images) and optional text descriptions (object shapes, colors, positions, orientations, etc), which are then used subsequently for training a robot. Text descriptions enable researchers to opt to work at high-level reasoning about what happens in the scene instead of pure visual learning. A demonstration can be edited as it is created, by undoing unwanted actions. Since

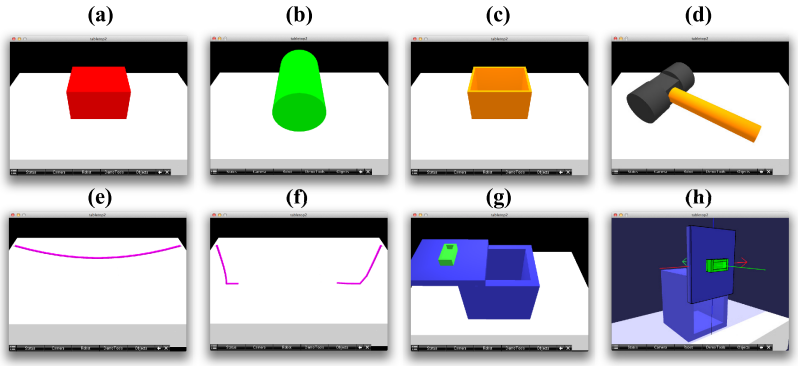

Fig. 2. Sample objects generated via XML files. (a)–(c) show simple objects: a block, a cylinder, and a box container. (d) shows a composite object, a hammer. (e), (f) show a string that when cut dynamically falls onto the table. (g), (h) show a box container with a sliding lid.

the demonstrator is not visible in the simulated world, it is therefore not necessary for the robot to capture and understand human motions or to transform between coordinate systems. Instead, learning can be focused entirely on the consequences of the demonstrator's manipulative movements in the task/object space. That is, from the robot's perspective, the objects in the environment appear to move on their own.

The demonstrator conceptually has two manipulators, or "hands", which can independently manipulate two objects. Each hand can manipulate an object by grasp, move, rotate, and release operations. A manipulation starts by grasping an object, from which point on the object is temporarily unaffected by gravity, i.e., it is held by an invisible hand. A user does so by simply clicking on an object to be moved. The object can then undergo a series of movements and/or rotations. The user can move the object by dragging it. In Fig. 3, the block on the right side is grasped and lifted above the tabletop. To guide the user through moving an object in 3D space using 2D mouse inputs, a rotatable restricting plane and a virtual shadow is added to the demonstrator's perspective (Fig. 3a–b). The object can also be rotated in the three primary axes using GUI controls (Fig. 3c–d). Finally, when the object has been moved/rotated to a desired destination/orientation, it can then be released by clicking the release button. The object resumes being affected by gravity after it is released. The demonstrator can switch between the two hands while manipulating objects. To manipulate two objects in parallel, the demonstrator needs to manually interleave the manipulative actions of the two hands. In the case where the user made an unwanted action, an undo function is provided to restore the previous world state and discard the corresponding segment in the recorded video.

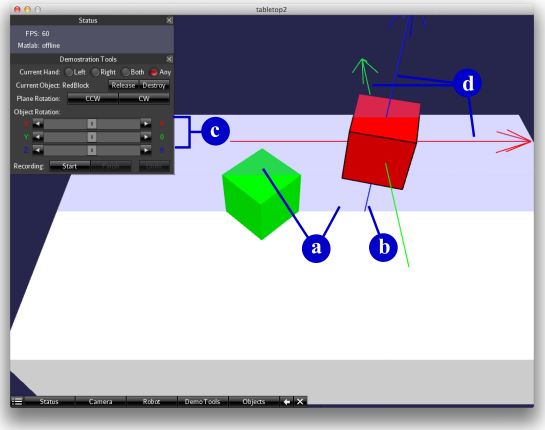

Fig. 3. The demonstration interface. The red block (right) is currently grasped by the demonstrator and raised above the tabletop. (a) A restricting plane perpendicular to the tabletop that guides movement of the grasped object in the 3D world. The plane can be rotated around the vertical axis to select different moving paths. (b) A virtual shadow indicating the location of the object. (c) Sliders that control object rotations around (d) the three primary axes (color coded).

2.3 Robot Agent

A Matlab application programming interface (API) for controlling a simulated robot is provided. The interface is designed such that a user program serves as the "brain" of the robot that communicates with the body via sensory inputs and motor commands. Specifically, the user program is invoked iteratively at approximately 60Hz (depending on the computing capacity and complexity of the callback script). In each iteration, the user program is supplied with sensory information in a Matlab variable such as time elapsed since last invocation, current joint angles, current gripper opening, visual images, and end effector positions. The user program can then specify motor commands in another Matlab variable containing joint and gripper velocities. This API resembles a typical perception-action cycle that is adopted by many bio-inspired intelligent agents.

The robot currently built into this system is modeled after Baxter®, a commercial bimanual robot by Rethink Robotics. Each arm of the robot consists of seven joints, two on the shoulder, two on the elbow, and three on the wrist. Physics effects are implemented for the arms to affect objects, making it possible to perform tasks such as pushing an object using a forearm. The robot "sees" the simulated world through a camera mounted on its head, which can pan horizontally. The 3D model and joint configurations of the robot are extracted from data released by the manufacturer[6]. A gripper is attached at the end of

[6] `https://github.com/RethinkRobotics/`

each arm, which can act on objects in the environment. The distance between the two "fingers" of a gripper can be widened or narrowed, so the gripper can hold or release objects. When a gripper is closing, a simple algorithm is used to determine if an object should be considered grasped by a gripper based on the contact points and normals. If so, the object is attached to the gripper and becomes unaffected by gravity. On the other hand, when the gripper is opening, any grasped objects are considered released and resume being affected by gravity. For debugging purposes, the software provides additional features such as manually controlling robot joints and drawing visual markers in the simulated environment.

2.4 Physical Performance

After a robot controller Matlab program is trained and tested with the simulated robot, it is straightforward to migrate it to control a real Baxter robot through ROS and Robot Raconteur[7]. The latter provides an interface containing methods to read sensor information, such as joint angles and camera images, and to write motor directives, such as new joint velocities or positions, in a similar way to our robot API. Some calibration and parameter tuning is needed to compensate for the differences between the simulated and the physical worlds, such as different camera parameters, lighting conditions, or gripper sizes. Simulated physics can also be different from real physics in complex scenes. It is up to individual robot learning methods to incorporate these factors to build a more generalized controller that reacts to environmental variations.

3 Results: An Example

As a first step in validating the virtual demonstrator hypothesis and to provide an example of our object-centric paradigm, we studied a case of programming by demonstration that involves a simple block stacking task. Initially, there are several blocks in three different sizes lying on the tabletop. The demonstrator manipulates these blocks to build a structure in the simulated environment, and the process is simultaneously recorded as a demonstration. The goal is for a physical robot to start with the same set of blocks placed at different initial positions in the real world, and eventually build the same structure in the same sequence as seen in the demonstration. This example by no means exhausts all possible issues one may encounter in studying robot programming by demonstration, but simply provides a proof of concept: that learning from a virtual demonstrator is viable, at least in certain situations.

First, an XML file was written to generate the initial scene (Fig. 4; left) for the demonstrator to work on. The scene contains ten cubic blocks in red or blue, a $1 \times 1 \times 3$ block in blue, and two $1 \times 1 \times 2$ blocks in red. A human demonstrator then manipulates these blocks to build a structure using only mouse inputs and

[7] `https://robotraconteur.com`

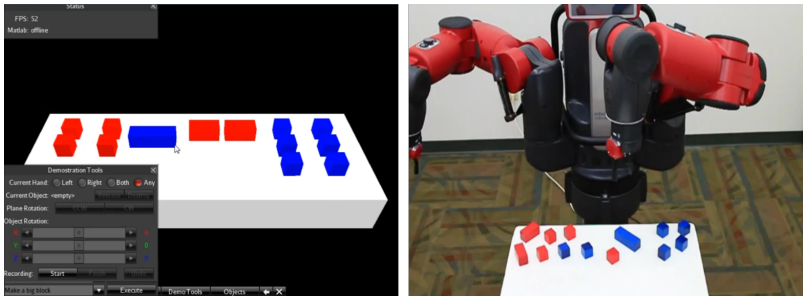

Fig. 4. Left: An initial scene as seen by the demonstrator. Right: A different initial scene for the robot to perform the demonstrated task.

GUI controls. In this specific example the demonstrator builds what resembles the letters "UM" (for our institution). A demonstration video is recorded during the demonstration. The demonstrator may elect to undo mistaken actions, in which case the video is updated accordingly. Figure 5 shows four screenshots sampled from a single demonstration. In each screenshot, the human demonstrator acts on the objects in the simulated environment. The demonstrator is invisible to the robot, and thus the blocks appear to be "flying" around from the robot's perspective. Along with the video, symbolic descriptions are generated to indicate which demonstrator's hand is being used and the locations of objects, etc. This symbolic information is optional: a purely vision-based learning system would be able to estimate the locations of objects without referring to symbolic information, but here we intentionally keep our Matlab program simple by using symbolic information to skip some of the complex aspects of image processing.

The physical robot is presented with a new initial scene containing the same blocks from the demonstration (Fig. 4; right).The initial locations of blocks are significantly different from what the human demonstrator started with (Fig. 4; left). In this assessment, the robotic system is learning the intentions/goals of the demonstrator from a single demonstration, and not memorizing the demonstrator's arm trajectories (the demonstrator is invisible as far as the robotic learner is concerned). Thus the robot must immediately generalize to handling a new initial state rather than the initial state used by the demonstrator.

The robot imitation learner[8] is implemented in Matlab and uses the same API to communicate with the physical robot and simulated robot. The Matlab program takes as input the demonstration video (along with symbolic information) and the image capture of the new initial scene (no symbolic information is used). The raw image is "parsed" to determine the locations and orientations of each block, and encode this information in a symbolic form. The Matlab program then generates a plan that best matches what is observed in the demonstration

[8] We are mainly describing the paradigm here; our robotic learning system will be the subject of a future paper.

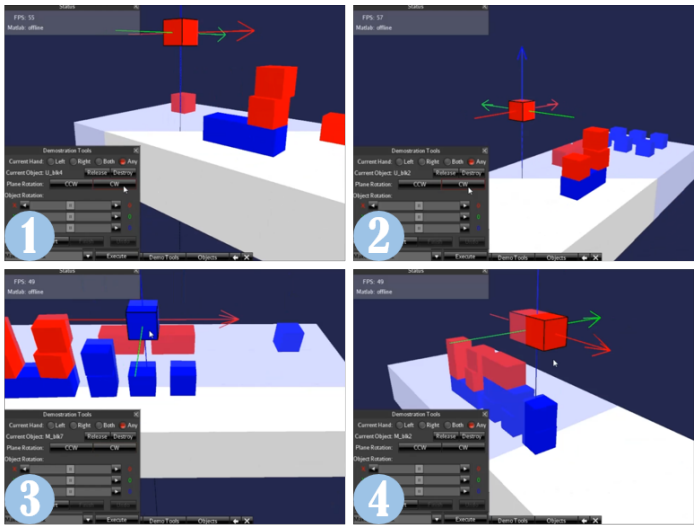

Fig. 5. Screenshots during a demonstration of stacking blocks into the letters "UM" in the virtual environment. The screenshots are temporally ordered in (1)–(4). In (2), the "U" is almost done. In (4), the final piece is going into position.

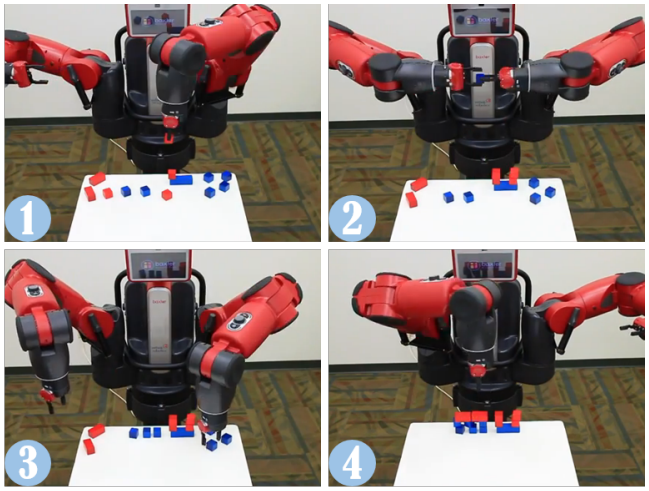

Fig. 6. The robot's successful imitation of stacking blocks into the letters "UM". Pictures are taken in the order of (1)–(4). In (2), arm coordination is being executed as Baxtor transfers a block from one hand to the other. In (4), the final "UM" is present (pictures were taken from the opposite side of "UM").

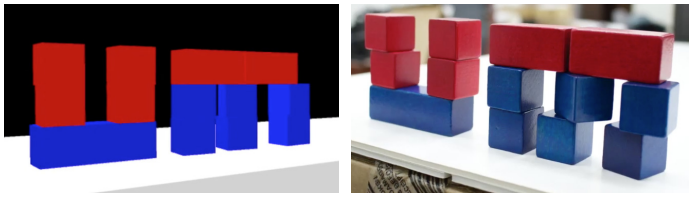

Fig. 7. The goal state demonstrated in the virtual environment (left) and the one executed by the robot (right).

video, including the order in which to stack blocks, the colors and sizes of blocks to be used, the relative positions of blocks in the target structure, and which arm to use. If no arm is specified in the demonstration, the program chooses which arm to use in accordance with the new initial scene. The plan is eventually translated into a sequence of waypoints for the robot arms to reach, which are then converted to arm joint velocities and passed to the robot for execution. The execution of the plan is shown in Fig. 6 (sampled screenshots). The resulting structure the robot built and the procedure in which the robot built it are qualitatively similar to what the human demonstrated (see Fig. 7), despite that they were started from different initial conditions. The color arrangement of blocks are properly imitated, although the blocks sometimes are not perfectly aligned.

4 Discussion

In summary, we presented an object-centric paradigm for robot programming by demonstration, in which we postulate that in many cases learning from demonstration can be both simplified and made more effective by having the demonstrator be invisible. As such, efforts during learning can be shifted from the difficult problem of human motion perception and interpretation to more task-relevant aspects by focusing only on the behaviors of the objects being manipulated. To this end, a software-based simulated environment was built for creating object-only demonstrations as well as training and testing robot learners. The resulting robot agents can then be used to control a real robot. We were able to quickly develop a block stacking robot imitator without recording or parsing human motions, and without analyzing human-robot body correspondences and coordinate transformations. The "UM" task showed that a software-generated object-centric demonstration can eventually be converted to physical robot actions.

Our method promotes learning based on the effects of actions, i.e., the movements of objects, rather than the actions themselves. In many tasks whose goal is to create a certain state external to the performer's body, capturing and understanding body actions can be difficult and irrelevant. On the other hand, learning from the effects of actions can potentially result in more goal-driven

action plans. In fact, since a robot's body configurations are usually very different from those of humans, direct learning from a human's actions is not always possible and may not result in the same effects. In our current result, although the human demonstrator used the same hand throughout the task, the robot decided to transfer blocks from one hand to the other (Fig. 6(2)) due to limitations on which blocks each hand could reach. In this case, imitation strategies that directly mimic human motions may fail. Moreover, highly sophisticated robots may benefit from not mimicking human motions. For example, robots with more arms than humans can potentially complete a task in a way more efficient than a human's approach.

The approach we took heavily relies on software simulations. Compared with physical demonstrations, this simulation-based approach provides a natural way to completely hide the demonstrator's body from the robot observer. The locations and orientations of each object in the simulated world can be obtained easily, instead of having to attach physical markers to each object and/or the demonstrator's body. A simulated environment also reduces human fatigue and risks, as well as managing mental workload and emotional responses. For example, a human can avoid physically demonstrating dangerous tasks in hazardous environments such as battlefield or underwater. After a robot is trained, it can potentially work with humans as a team while taking over the most dangerous parts of a task, which avoids poor performance caused by human fatigue and critical emotional responses. Furthermore, the simulation abstracts the sizes of objects, so it is now possible to demonstrate what is normally too large or too small to demonstrate in the physical world, such as building a bridge or operating at a nanotechnology level.

The main tradeoff for using our method is that not all tasks can be demonstrated in the software simulation. It is only applicable to a certain class of tasks where object behaviors alone matter. Our method cannot be used in situations where the goal is to mimic human body motions, such as learning a sign language, although it is conceivable that it might be extended to such a challenging task. It does not faithfully capture the trajectory and velocity at which a human would move an object by hand, due to its using a GUI and mouse inputs which inevitably restrict all possible trajectories and velocities to a small subset. It is also not suitable for tasks where the amounts of forces applied to objects are important, although such information is difficult to present even in a physical demonstration environment. Moreover, simulating high-precision physics in real time is generally a hard problem, and thus some simplifications and compromises must be made to reduce computational costs. This requires extra efforts to be put in software implementation and inevitably incurs some unrealistic physical effects in the simulated environment. Currently, our software is limited in simulating only rigid body physics.

Our future work will focus on developing a neurocognitive system that learns from object-centric demonstrations in a complex task scene where it is critical for the robot agent to reason about what happens in a demonstration. We will expand the types of supported task objects including tools.

Acknowledgment. Supported by ONR award N000141310597.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469 – 483 (2009)
2. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 1371–1394. Springer (2008)
3. Akgun, B., Cakmak, M., Yoo, J.W., Thomaz, A.L.: Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In: ACM/IEEE Int. Conf. on Human-Robot Interaction. pp. 391–398 (2012)
4. Billard, A.G., Calinon, S., Guenter, F.: Discriminative and adaptive imitation in uni-manual and bi-manual tasks. Robotics and Autonomous Systems 54(5), 370 – 384 (2006)
5. Chen, J., Haas, E., Barnes, M.: Human performance issues and user interface design for teleoperated robots. IEEE Trans. Syst., Man, Cybern. C 37(6), 1231–1245 (2007)
6. Sweeney, J., Grupen, R.: A model of shared grasp affordances from demonstration. In: IEEE-RAS Int. Conf. on Humanoid Robots. pp. 27–35 (2007)
7. Chong, J., Ong, S., Nee, A., Youcef-Youmi, K.: Robot programming using augmented reality: An interactive method for planning collision-free paths. Robotics and Computer-Integrated Manufacturing 25(3), 689–701 (2009)
8. Martin, R., Sanz, P., Nebot, P., Wirz, R.: A multimodal interface to control a robot arm via the web: a case study on remote programming. IEEE Trans. Ind. Electron. 52(6), 1506–1520 (2005)
9. Azad, P., Asfour, T., Dillmann, R.: Robust real-time stereo-based markerless human motion capture. In: IEEE-RAS Int. Conf. on Humanoid Robots. pp. 700–707 (2008)
10. Dillmann, R., Asfour, T., Do, M., Jäkel, R., Kasper, A., Azad, P., Ude, A., Schmidt-Rohr, S., Lösch, M.: Advances in robot programming by demonstration. KI - Künstliche Intelligenz 24(4), 295–303 (2010)
11. Calinon, S., Billard, A.G.: What is the teacher's role in robot programming by demonstration?: Toward benchmarks for improved learning. Interaction Studies 8(3), 441–464 (2007)
12. Aleotti, J., Caselli, S.: Physics-based virtual reality for task learning and intelligent disassembly planning. Virtual Reality 15(1), 41–54 (2011)
13. Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K., Saunders, J.: Achieving corresponding effects on multiple robotic platforms: Imitating in context using different effect metrics. In: Int. Symp. on Imitation in Animals and Artifacts (2005)
14. Chella, A., Dindo, H., Infantino, I.: A cognitive framework for imitation learning. Robotics and Autonomous Systems 54(5), 403–408 (5 2006)
15. Dominey, P.F., Alvarez, M., Gao, B., Jeambrun, M., Cheylus, A., Weitzenfeld, A., Martinez, A., Medrano, A.: Robot command, interrogation and teaching via social interaction. In: IEEE-RAS Int. Conf. on Humanoid Robots. pp. 475–480 (2005)
16. Feniello, A., Dang, H., Birchfield, S.: Program synthesis by examples for object repositioning tasks. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2014)