A Virtual Demonstrator Environment for Robot Imitation Learning

Di-Wei Huang*, Garrett Katz*, Joshua Langsfeld†, Rodolphe Gentili‡ and James Reggia*
*Department of Computer Science,†Department of Mechanical Engineering,‡Department of Kinesiology,
University of Maryland, College Park, MD 20742
Email: {dwh@cs., gkatz12@, jdlangs@, rodolphe@, reggia@cs.}umd.edu

***Abstract*—To support studies in robot imitation learning, this paper presents a software platform, SMILE (Simulator for Maryland Imitation Learning Environment), specifically targeting tasks in which exact human motions are not critical. We hypothesize that in this class of tasks, object behaviors are far more important than human behaviors, and thus one can significantly reduce complexity by not processing human motions at all. As such, SMILE simulates a virtual environment where a human demonstrator can manipulate objects using GUI controls without body parts being visible to a robot in the same environment. Imitation learning is therefore based on the behaviors of manipulated objects only. A simple Matlab interface for programming a simulated robot is also provided in SMILE, along with an XML interface for initializing objects in the virtual environment. SMILE lowers the barriers for studying robot imitation learning by (1) simplifying learning by making the human demonstrator be a virtual presence and (2) eliminating the immediate need to purchase special equipment for motion capturing.**

I. Introduction

Contemporary robots are built to exhibit a wide range of behaviors, but in most cases these behaviors must be programmed in manually. Such programming is time-consuming, difficult, expensive, and requires highly trained roboticists, yet the results usually do not generalize well to even moderately altered tasks, initial conditions, and/or changing environment. One potential solution is to replace manual programming with imitation learning [1], [2], or programming by demonstration, where a robot autonomously observes a human perform a task (i.e., demonstration) and then attempts to complete the same task by replicating the observed behaviors (i.e., imitation). This method potentially could allow almost anyone, including task experts, with little to no knowledge about robotics to program a robot in a straightforward fashion. In addition to its practical motivations in robotics, imitation learning has drawn increasing interest across several areas, including artificial intelligence and cognitive and neural sciences (the latter due to its relation to mirror neuron systems [3]).

Among many others, a key issue of imitation learning is how an interface between a human and a robot can be designed to facilitate effective transferring of task-related knowledge and skills from a human demonstrator to a robot. We will refer to this interface as the *demonstration interface*, whose format is of central importance because it directly determines what information a robot receives, what the robot can learn from it, and ultimately how well the robot can imitate. The demonstration interface affects the human side too, such as the time and the cost to set up necessary equipment and environment, as well as the friendliness of the interface for human demonstrators, especially for those who know little about robotics.

Existing demonstration interfaces can be classified into two classes. In the first class, the human demonstrator drives or manipulates the states of the robot directly, 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 [4], [5], or teleoperating, where the robot is driven by remotely located controls [6], [7]. The robot being driven can also exist in augmented realities [8] or virtual realities [9]. 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 [10], [11]. This method has been used in conjunction with kinesthetic teaching [12]. 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 motions are usually passed to the robot furnished with additional information, such as the states of task-related objects [13]–[15], from which the robot can learn the relations between hand motions and objects, and speech descriptions about the motions, from which the robot can learn the intentions of the motions from symbolic annotations [16].

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 equipments for capturing a human's movement and actions. 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. Further, coordinate transformations must be learned between the demonstrator's and learner's frames of references. 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, especially those involving a procedure (i.e., a series of actions conducted in a certain order or manner), 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 and fixing a computer. By making the demonstrator a virtual presence, one hugely simplifies the motion tracking and understanding during learning and eliminates the coordinate transformation problems involved. This approach effectively shifts the problem of imitation learning 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 [13], [14], they have either used an over-simplified environment (2D simulated world) or required special equipment.

As a first step towards studying the virtual demonstrator hypothesis, we present a software simulated environment, named SMILE (Simulator for Maryland Imitation Learning Environment), as a means to hide a demonstrator's body from a robot learner. The work reported here is mainly about the software platform SMILE and the virtual demonstrator hypothesis, as opposed to describing any specific cognitive robotics approaches to imitation learning. The latter will be the subject of a future paper. 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 support for the hypothesis. Complementary experiments guided by this current effort will also help assess this hypothesis.

II. Simulator for Maryland Imitation Learning Environment (SMILE)

SMILE is an integrated virtual environment for studying imitation learning based on the virtual demonstrator hypothesis [17]. In this environment, a task space can be set up with user-defined task-relevant objects. A demonstrator can then use mouse inputs to manipulate these objects, the process of which is recorded as an object-only demonstration "video" with accompanying description about the states of the objects. The video can be edited as it is created by undoing unwanted actions. The demonstrator's body is not represented in the demonstration, making the objects appear to be manipulated by "invisible hands" or move on their own. Finally, a robot agent learns from the demonstration and can test out the plans it forms with a simulated robot and objects in the same environment.

Our initial efforts have focused on creating a tabletop environment for bimanual object manipulation tasks, one that is being intensively studied such as in [18]. As a result, SMILE contains a 3D simulated world where a virtual demonstrator, a robot, a table, and a variety of task-related objects coexist. An example view of SMILE is given in Fig. 1, which contains a 3D environment and overlaid GUI controls (Fig. 1; top-right). The environment can be observed through the demonstrator's perspective (Fig. 1; main window) and/or through the robot's perspective (Fig. 1; bottom-right). The demonstrator's view is freely navigable by a user. Real-time rigid body physics simulation is included based on the Bullet physics library[1]. A user can demonstrate a task by a combination of clicking/dragging the objects and using the overlaid graphical user interface (GUI) system. No special hardware is required.

There are three major ways in which a user can interact with SMILE (Fig. 2): (a) to demonstrate a task through mouse inputs, (b) to program the robot's behaviors through Matlab scripts, and (c) to initialize task-related objects through XML. An application programming interface (API) for Matlab is provided for programming the behaviors of the simulated robot, which can act on objects in the environment. This API can potentially be used to implement and to experiment with imitation learning methods. The XML interface is provided to initialize the scenes, i.e., defining objects on the tabletop, for different tasks and experiments. The following subsections describe each of these three interfaces of SMILE.

[1] http://bulletphysics.org

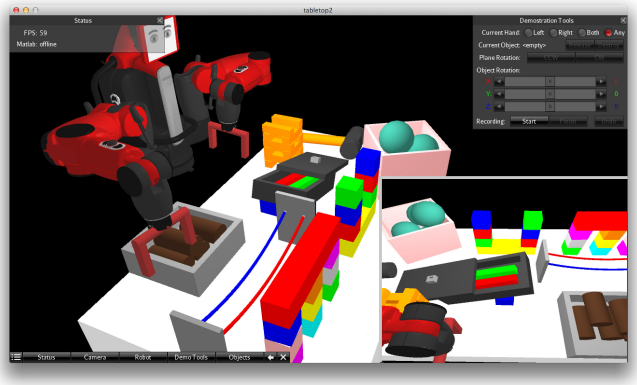

Fig. 1. An example view of SMILE. The main screen shows a simulated environment containing a tabletop and a variety of objects (block, boxes, lids, strings, etc.), as they are seen by the human demonstrator. The avatar for a two-armed programmable robot is also embedded in the environment. The bottom-right 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.

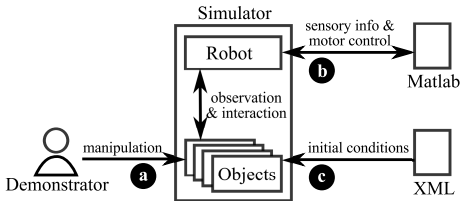

Fig. 2. The three primary interfaces of the simulator: (a) the demonstration interface, (b) the Matlab programming interface, and (c) the XML object generation interface.

A. Demonstration Interface

A human demonstrator can manipulate the objects through the GUI using mouse inputs. A demonstration containing multiple object manipulations can be recorded as a video (a sequence of images) that is then used subsequently for training a robot. The demonstration video can be edited by undoing unwanted actions. For the purpose of testing our virtual demonstrator hypothesis, the demonstrator is not visible in the simulated world, and therefore it is 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.

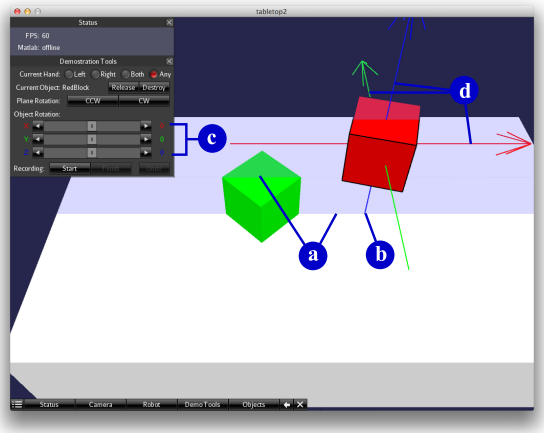

Fig. 3. The demonstration interface. The red block (right) is currently grasped by the demonstrator and raised above the tabletop. (a) A restricting plane perpendicular to the tabletop that guides movement of the grasped object in the 3D world. The plane can be rotated around the vertical axis to select 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).

The demonstrator is conceptually equipped with 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 using mouse inputs. 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 to a desired destination and rotated to a desired 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.

While recorded videos are suitable for visual learning, SMILE also provides an option for symbolic output. That is, in addition to the video frames, a symbolic description can be generated in text format describing object properties for each video frame, such as the shape, color, coordinates, etc. of each object. This information not only further simplifies image processing, but also enables researchers to opt to work at high-level reasoning about what happens in the scene.

SMILE's demonstration GUI can be readily extended to allow a demonstrator to specify which properties of a manipulated object are significant (not implemented yet). For example, when moving an object, the demonstrator can choose from a list of object properties. If "color" is selected, this information is passed to the robot learner to direct its internal attention mechanism towards the color of the object being moved.

B. Application Programming Interface for the Robot

The second interface of SMILE is a Matlab programming interface for controlling a simulated robot. The interface is designed such that a user's Matlab program acts like the "brain" of the robot. (In this subsection, a "user" refers to a developer who programs the robot's imitation learning behaviors.) That is, a user program receives sensory inputs in the form of vision (images) and proprioception (joint angles) information, and provides motor outputs in the form of motor commands (joint velocities). This framework is intended to facilitate work on bio-inspired robot controllers.

The robot currently built into this system is modeled after Baxter® (Fig. 4a), 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 (Fig. 4b), which can pan horizontally. The 3D model and joint configurations of the robot are extracted from data released by the manufacturer[2]. A gripper is attached at the end of 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.

To program the robot's behaviors, the user needs to provide two Matlab scripts: an initialization script and a callback script. The initialization script is invoked once by SMILE to initialize user-defined variables. After that, the callback script is invoked repeatedly at approximately 60Hz (depending on the computing capacity and complexity of the callback script). For each invocation, SMILE supplies the callback script with the robot's sensory information in a Matlab structure variable `sensor`, which contains, for example, time elapsed since last invocation, current joint angles, current gripper opening, visual images, and end effector positions. The callback script can then write to another predefined Matlab structure variable `motor` to issue motor commands, which contains joint velocities and gripper velocities. Additionally, both initialization and callback scripts can access a Matlab structure `aux`, which contains auxiliary information such as the angle range for each joint.

For debugging purposes, manual control of the robot is provided through GUI panels (Fig. 4c) where a user can rotate individual joints manually and exert other influences. Additionally, the `aux` variable allows the user to draw visual markers in the simulated environment. They are not visible to

[2]https://github.com/RethinkRobotics/

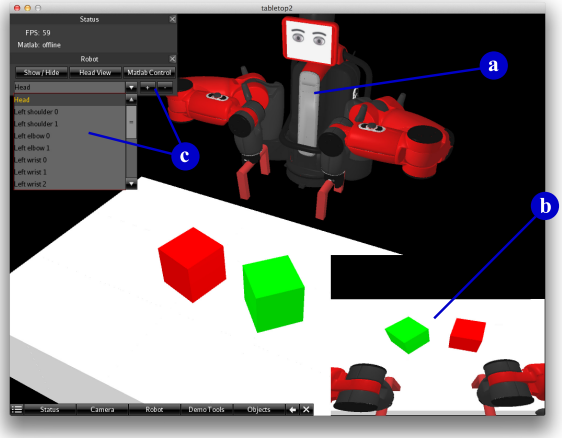

Fig. 4. The robot in the simulated world. (a) The 3D model of the robot. (b) The robot's view of the simulated world. The images are captured by its head-mounted virtual camera. (c) GUI controls for manually controlling the robot.

the robot, do not interact with other objects, and are useful in highlighting spatial locations in the environment.

C. Object Initialization Interface

The third interface is for initializing and creating objects in the simulated environment. This interface provides a simple way to specify scenarios for different tasks and experiments. Specifically, SMILE can load XML files written in conformance to a predefined XML schema and then generate objects in the simulated environment accordingly. The XML schema defines XML elements for generating simple objects including blocks, cylinders, spheres, and containers (Fig. 5a–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. 5d 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. 5e), which, when cut, will fall to the table (Fig. 5f). A container with a sliding lid can also be created (Fig. 5g). The lidded box can be positioned to become a container with a trap door (Fig. 5h).

III. Results: A Usage Example

As a first step in validating the virtual demonstrator hypothesis with SMILE, we show an example of using it during an imitation learning task 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, the process of which is recorded as a demonstration video. The goal is for the robot to start with the same set of blocks in different initial positions, 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 imitation learning, but simply provides a proof of concept:

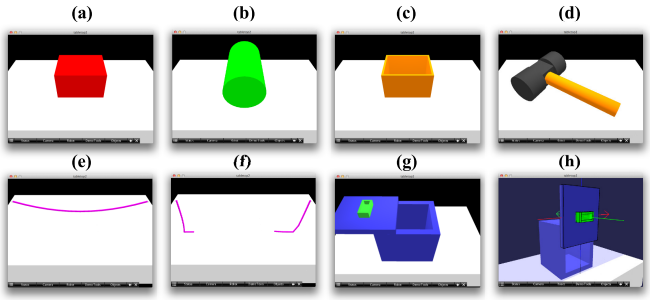

Fig. 5. Sample objects generated via XML files. (a)–(c) show simple objects: a block, a cylinder, and a box container. (d) shows a composite object, a hammer. (e), (f) show a string that when cut dynamically falls onto the table. (g), (h) show a box container with a sliding lid.

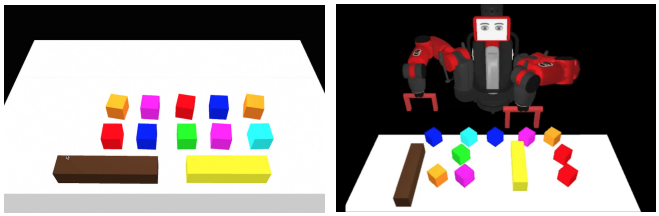

Fig. 6. Left: An initial scene as seen by the demonstrator. Right: A different initial scene for the robot to perform the demonstrated task.

that learning from a virtual demonstrator is viable, at least in certain situations.

First, an XML file is written to generate the initial scene (Fig. 6; left) for the demonstrator to work on. The scene contains ten unit-size blocks in six different colors, a $1 \times 1 \times 4$ block in yellow, and a $1 \times 1 \times 5$ block in brown. A human demonstrator then manipulates these blocks to build a structure using only mouse inputs and 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, and data structures are created internally to track object movements. The data structures are needed to restore environment states retroactively when the user elects to undo actions. Fig. 7 shows four screenshots sampled from a single demonstration. In each screenshot, the human demonstrator acts on the objects in the simulated environment through the main window, while the bottom-right corner shows what is being observed by the robot and recorded as the demonstration video. 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.

A second XML file is prepared to initialize a new scene for the robot to work on (Fig. 6; right). The initial locations of blocks are significantly different from what the human

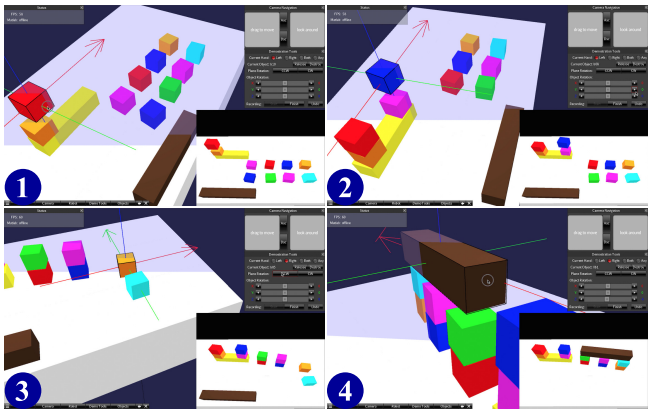

Fig. 7. Screenshots during a demonstration of stacking blocks into the letters "UM" using the GUI and mouse inputs. The screenshots are temporally ordered in (1)–(4). The main view of each image shows the demonstrator's view of the simulated environment. The lower-right corner of each image shows the robot's view of the demonstration.

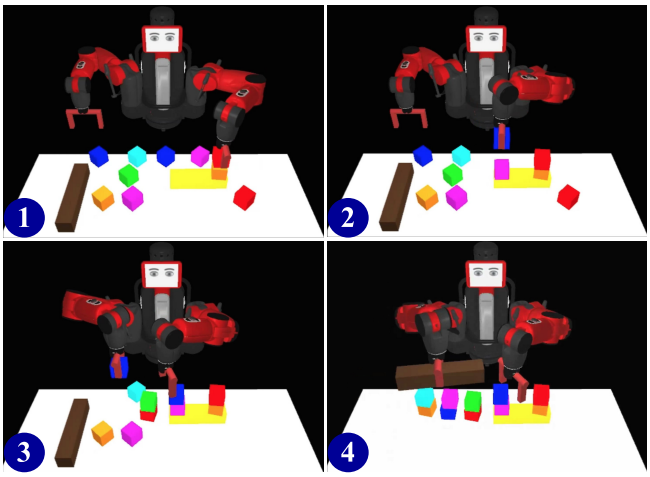

Fig. 8. The robot's successful imitation of stacking blocks into the letters "UM". Screenshots are taken in the order of (1)–(4).

demonstrator started with (Fig. 6; left). In this assessment of SMILE, one 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[3] is implemented in Matlab and communicates with SMILE through the robot's API. The Matlab program takes as input the demonstration video (along with symbolic information) and the robot's vision of the new initial scene (no symbolic information is used). The initial scene is parsed to determine the locations of each block. The Matlab program then generates a plan that best matches what is observed in the demonstration 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. 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 SMILE for robot simulation. The execution of the plan is shown in Fig. 8 (sampled screenshots). What the robot builds is structurally similar to what the human demonstrator builds, despite starting from a different initial conditions. The color arrangement of blocks are properly imitated, although the blocks sometimes are not perfectly aligned.

IV. Discussion

In summary, we have developed a software simulated environment, SMILE, for studying robot imitation learning based on the virtual demonstrator hypothesis, in which we postulate that imitation learning for many tasks can be simplified and potentially 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, SMILE provides an intuitive GUI-based demonstrator's interface that literally does not embody the demonstrator. Demonstrators are allowed to manipulate objects in the simulated environment without special equipment, and the recorded demonstration can be used for training a robot simulated in the same environment, where the robot can be programmed using Matlab scripts. Objects of various types can be created in the environment with ease using XML. We also showed a usage example where the robot, programmed in SMILE by simple Matlab scripts, can imitate a human stacking the letters "UM" using blocks.

The main purpose of SMILE is to provide an intuitive tool for demonstrating tasks in a simulated environment without the demonstrator's body being recorded. This not only allows us to further investigate the virtual demonstrator hypothesis, but also is a potential command interface for instructing a physical robot what to do once an imitating robot agent is in place. By using a simulated environment, the human demonstrator can avoid dangerous tasks and hazardous environments, and thus reduce critical emotional responses. This demonstration interface requires no special equipment and removes the need to capture and parse complex human motions, which greatly reduces the cost and complexity of a learning system. Compared with physical demonstration, where a human has to manually manipulate objects existing either in the physical world or in a virtual reality, we believe that the GUI introduced by SMILE can reduce human fatigue by using a keyboard and a mouse. Furthermore, the interface abstracts the sizes of objects, so it is now possible to demonstrate in SMILE 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 secondary purpose of SMILE is serving as a testbed for prototyping robot imitation learning. A simulated robot, situated in the same 3D world where a human demonstrates tasks, provides an inexpensive environment for experimenting with robot learning. Although there are well-known robot simulators such as Gazebo, OpenRAVE, Webots, etc., they generally do not support demonstrating and recording task procedures, and the learning curve is relatively steep. This can be a barrier for non-roboticists, such as researchers in general AI and neurocognitive sciences, to study the virtual

[3]We are describing only SMILE here; our robotic learning system will be the subject of a future paper.

demonstrator hypothesis. We chose to build our software platform to grant ourselves more freedom and to help identify research issues in this early stage of studying the hypothesis. Porting SMILE's core features to a well-known robot simulator will be one of our future goal.

While SMILE has its advantages, there are tradeoffs when using SMILE for robot imitation learning. First, a limitation of SMILE is that it is only applicable to a certain class of tasks where object behaviors alone matter. SMILE cannot be used in situations where the goal is to mimic human body motion, such as learning a sign language, although it is conceivable that it might be extended one day 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.

A second limitation arises as simulating physics in real time is generally a hard problem. Finding an exact solution to a physics system, especially ones that contain many interacting objects, is difficult and computationally expensive. Additionally, numerical accuracy is limited, potentially causing spatial fluctuations in the simulated world. These could possibly accumulate and be amplified over time and produce unrealistic or erroneous physical effects. In theory, one can easily increase numerical accuracy, but this significantly increases computation time, and thus may lower temporal resolution of a physics simulation with standard hardware to a point where erroneous physical effects can still take place. In order to maintain adequate temporal resolution, existing physics simulation software usually opts to limit numerical accuracy in favor of fast computation. Under these restrictions, SMILE is currently not well suited for tasks that require high-precision physics, such as simulating a small screw being put into a threaded hole and tightened using friction. To simulate high-precision physics, case-by-case simplifications can be added to the environment as a workaround. For example, the robot in SMILE does not grasp an object by friction, but there is a simple algorithm that makes a one-shot decision, based on contact points and normals between a gripper and the object, that whether a grasp of object is successful. If successful, the object is directly attached to the gripper unless released, instead of continuously calculating friction between the gripper and the object. Such a simplification can significantly reduce computational cost by sacrificing some fidelity to physics. However, there are situations where even simplifications are complicated, such as fluid and air dynamics.

Our immediate future work will be interfacing with a physical robot, such that the physical robot performs in the real world what a human demonstrates in the simulated world. This will allow a human to "program" a physical robot on a computer by virtual demonstration. We will evaluate the effectiveness of the virtual demonstrator hypothesis more rigorously in the near future, with user studies and performance comparisons between robot agents. Another focus will be on increasing task scene complexity by expanding types of objects supported by SMILE, as well as their simulated physical effects. We are particularly interested in adding tool objects that can act on other objects, such as scissors and screwdrivers.

ACKNOWLEDGMENT

Supported by ONR award N000141310597.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 1371–1394.

[3] E. Oztop, M. Kawato, and M. A. Arbib, "Mirror neurons: Functions, mechanisms and models," *Neurosci. Lett.*, vol. 540, pp. 43 – 55, 2013.

[4] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2012, pp. 391–398.

[5] A. G. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 370 – 384, 2006.

[6] J. Chen, E. Haas, and M. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 6, pp. 1231–1245, 2007.

[7] J. Sweeney and R. Grupen, "A model of shared grasp affordances from demonstration," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2007, pp. 27–35.

[8] J. Chong, S. Ong, A. Nee, and K. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 689–701, 2009.

[9] R. Martin, P. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the web: a case study on remote programming," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1506–1520, 2005.

[10] P. Azad, T. Asfour, and R. Dillmann, "Robust real-time stereo-based markerless human motion capture," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2008, pp. 700–707.

[11] R. Dillmann, T. Asfour, M. Do, R. Jäkel, A. Kasper, P. Azad, A. Ude, S. Schmidt-Rohr, and M. Lösch, "Advances in robot programming by demonstration," *KI - Künstliche Intelligenz*, vol. 24, no. 4, pp. 295–303, 2010.

[12] S. Calinon and A. G. Billard, "What is the teacher's role in robot programming by demonstration?: Toward benchmarks for improved learning," *Interaction Studies*, vol. 8, no. 3, pp. 441–464, 2007.

[13] J. Aleotti and S. Caselli, "Physics-based virtual reality for task learning and intelligent disassembly planning," *Virtual Reality*, vol. 15, no. 1, pp. 41–54, 2011.

[14] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders, "Achieving corresponding effects on multiple robotic platforms: Imitating in context using different effect metrics," in *Int. Symp. on Imitation in Animals and Artifacts*, 2005.

[15] A. Chella, H. Dindo, and I. Infantino, "A cognitive framework for imitation learning," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 403–408, 5 2006.

[16] P.-F. Dominey, M. Alvarez, B. Gao, M. Jeambrun, A. Cheylus, A. Weitzenfeld, A. Martinez, and A. Medrano, "Robot command, interrogation and teaching via social interaction," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2005, pp. 475–480.

[17] D.-W. Huang, G. E. Katz, R. J. Gentili, and J. A. Reggia, "The maryland virtual demonstrator environment for robot imitation learning," University of Maryland, Tech. Rep. CS-TR-5039, 2014.

[18] A. Feniello, H. Dang, and S. Birchfield, "Program synthesis by examples for object repositioning tasks," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.