A Virtual Reality Platform for Dynamic Human-Scene Interaction

Jenny Lin[1]*, Xingwen Guo[12]*, Jingyu Shao[1]*, Chenfanfu Jiang[3], Yixin Zhu[1], Song-Chun Zhu[1]*
[1] UCLA Center for Vision, Cognition, Learning and Autonomy
[2] University of Hong Kong Electrical and Electronic Engineering Department
[3] UCLA Computer Graphics & Vision Laboratory

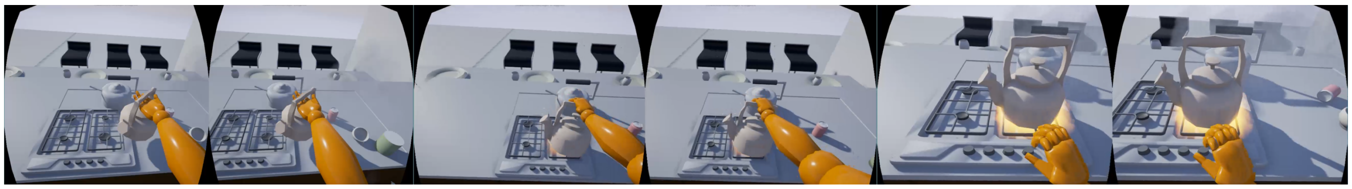


Figure 1: *Using Virtual Reality with Kinect body tracking & Leap Motion hand tracking, a real world agent moves a kettle in a virtual scene by reaching out and grasping it. This singular action is happening within the context of a larger overall task: in this case, boiling some water.*

Abstract

Both synthetic static and simulated dynamic 3D scene data is highly useful in the fields of computer vision and robot task planning. Yet their virtual nature makes it difficult for real agents to interact with such data in an intuitive way. Thus currently available datasets are either static or greatly simplified in terms of interactions and dynamics. In this paper, we propose a system in which Virtual Reality and human / finger pose tracking is integrated to allow agents to interact with virtual environments in real time. Segmented object and scene data is used to construct a scene within Unreal Engine 4, a physics-based game engine. We then use the Oculus Rift headset with a Kinect sensor, Leap Motion controller and a dance pad to navigate and manipulate objects inside synthetic scenes in real time. We demonstrate how our system can be used to construct a multi-jointed agent representation as well as fine-grained finger pose. In the end, we propose how our system can be used for robot task planning and image semantic segmentation.

Keywords: virtual reality, benchmark suite, 3D scene dataset

Concepts: •**Computing methodologies** → **Image manipulation;** *Computational photography;*

1 Introduction

Consider a scenario where we are trying to train a robot to perform a task in an indoor scene: say, find a can of coke, and pour some into a cup. From the AI perspective, we could take a spatial-temporal reasoning approach, using our knowledge about the 3D layout of the scene as well as the coke bottle's properties to predict a best approach [Durrant-Whyte 1988]. From a visual learning perspective, we could teach the robot to use RGB or RGB-D data to navigate the scene and infer appropriate grasp methods [Kragic et al. 2002]. In the robotics community, we would have it learn about the task directly, either using reinforcement learning approaches by allowing the robot to attempt the task repeatedly [Kaelbling et al. 1996; Levine et al. 2016], or using the scheme of Learning from Demonstration (LfD) [Argall et al. 2009] by collecting information about a human performing the same task with synthetic grasping [Shimoga 1996]. No matter what our methodology might be, we must have an appropriate dataset to train on, with the appropriate information labeled. Generating this data is already nontrivial, but it only grows more difficult as our task becomes more generalized: it might be reasonable dealing with a can of coke in a single kitchen, but what if we wish to expand to other beverages in other kitchens? What if we need the robot to learn other tasks as well? Labeling aside, even visiting that many scenes would be difficult.

*J. Lin, X. Guo and J. Shao contributed equally to this work. E-mail: jhl999888@g.ucla.edu, guoxingwen@ucla.edu, shaojy15@ucla.edu, cffjiang@cs.ucla.edu, yixin.zhu@ucla.edu, sczhu@stat.ucla.edu

SA '16, December 05-08 2016, Macao
ISBN: 978-1-4503-4548-4/16/12
DOI: http://dx.doi.org/10.1145/2992138.2992144

One solution would be to use virtual environments instead of real world scenes. 3D virtual environment data is publicly available both as fully scanned scenes [Zhu et al. 2016; Xiao et al. 2013], synthetic scenes [Handa et al. 2016; Handa et al. 2015], and scenes that could potentially composed from CAD models [Chang et al. 2015]. Indeed, programs exist which allow robots to simulate interactions with virtual environment [Miller and Allen 2004]. However, these virtual environments usually lack a detailed virtual human with fine-detailed interactions and complex indoor layout. Thus we propose the use of Virtual Reality (VR).

Recent commercial advancements has made VR available to the average consumer, making the time right for its usage in research. By combining VR with pose tracking technology such as the Kinect [Shotton et al. 2013] and Leap Motion [Weichert et al. 2013], we can incorporate the agent into the scene as a multi-jointed avatar capable of interacting with objects in real-time. Furthermore, using VR provides not only data about the various objects making up the scene, but detailed data about the agent as well, including the trajectories, human poses at each frame, fine-grained articulated hand poses.

We propose a system in which dynamic 3D scene models are imported into a physics-based engine and then combined with a virtual reality setup so that an agent can interact with each individual objects, including subparts (e.g. closet doors, cabinet drawers) of certain objects. This allows for the generation of fine-detailed dynamic manipulation data.

Related Work. Large-scale datasets have motivated the development of artificial intelligence and computer vision research by providing training/testing data as well as benchmark suits and protocols. Existing large datasets in computer vision field are mainly focused on non-interactive objects [Deng et al. 2009; Chang et al. 2015; Lin et al. 2014] or static scenes [Xiao et al. 2013; Silberman et al. 2012; Armeni et al. 2016], while robotics datasets are built up for small scale, project-oriented task planning and learning manipulation in limited scenarios. A closely related topic in robotics is social simulation among agents [Nehaniv and Dautenhahn 2007; Shu et al. 2016].

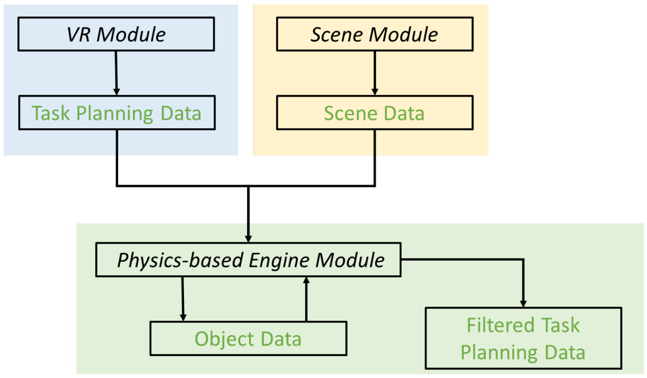


Figure 2: *System Modules*

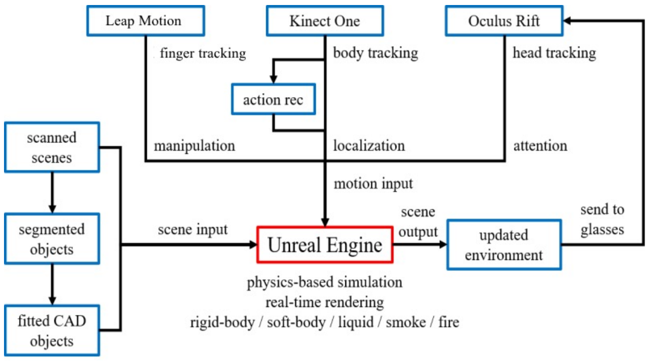


Figure 3: *System Design*

2 System Design

Our system [Fig. 2] is composed of three sub-modules: i) a VR Module, which collects data about the real world agent (e.g. head pose, body pose, hand pose), and displays the virtual environment to the agent, ii) a Scene Module, which provides 3D data about the virtual scene, and iii) a Physics-based Engine Module, which consolidates all the data, allowing the real world agent to interact with the virtual environment [Fig. 3].

2.1 Scene Setup

Scenes are constructed in the 3D modeling software SketchUp using assets downloaded from 3D Warehouse. Additional segmentation and scaling of the model is applied to obtain movable sub-parts and maintain consistent scale. For non-manipulable stationary objects in the scene (e.g., large furniture), we use SketchUp to compose them as a singular mesh. The composition is performed in the way such that the mesh is entirely contained within octant IV in SketchUp and faces the negative x-axis. This means a projection of the scene into the xy plane would place the origin at the scene's bottom left corner, while the entirety of the scene remains within the positive z-axis. Similarly, in the case of movable objects to manipulate, the object is placed in the scene such that it is contained entirely in the the z-axis and a projection onto the xy plane places the origin in its center.

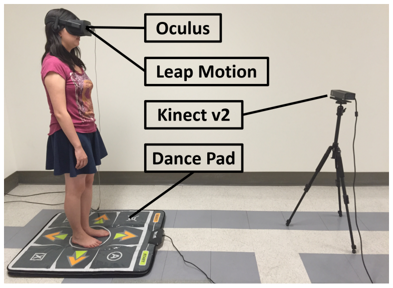


Figure 4: *An agent is wearing an **Oculus Rift** Headset with a **Leap Motion** controller mounted on its front, while standing on a **dance pad** and facing towards a **Kinect** sensor.*

2.2 VR Setup

The agent wears an Oculus Rift, which serves to both display the virtual scene to the agent as well as to collect the agent's head location for use by the engine. Mounted on the Oculus Rift is the Leap Motion Controller, which allows for fine-grained articulated finger tracking. Since the Kinect does not perform well if the agent's back is facing the sensor, the agent is asked to orient themselves such that their body is always facing towards the Kinect sensor to ensure optimal body tracking [Fig. 4]. Due to the limited range of the Kinect, the agent cannot physically navigate through a large virtual environment, hence an additional control scheme is required to navigate the virtual scene. Since the agent's hands must be free to interact with the scene, we have elected to use an eight-directional dance pad. This allows the agent to translate with two degrees of freedom (forward-back, left-right) and rotate without constraint around the vertical axis all while remaining within a constrained area and orientation. The dance pad serves the additional function of providing a defined location relative to the Kinect and Oculus' camera where the agent must stand.

2.3 Engine Integration

We chose to use Unreal Engine 4 (UE4) for our physics-based engine. UE4 is capable of simulating rigid body dynamics, soft body dynamics, and fluid dynamics. It is designed for demanding applications such as high quality real-time rendering. Its built-in support for VR development makes it easy to work with and also helps with rendering complex scenes at consistently high frame rates.

Scene Data Import. The scene data is exported from SketchUp as either FBX file or DAE file with the units set to centimeter, which is the standard unit within Unreal Engine 4. The auto generated collision boundaries are replaced with more accurate ones by using ”Auto Convex Collision” and setting parameter “Accuracy” to 1.0 and “Max Hull Verts” to 32. In the case where this step fails, we instead build a collision boundary from system defined collision components.

Avatar Setup. The VR data is polled from within UE4 to set up the pose and orientation of the virtual representation of the real world agent, hereon referred to as the avatar. The avatar consists of three components: i) a humanoid mesh that can be deformed accordingly based on the underlying skeleton; ii) a root transform, used to define the overall position of the avatar inside the virtual scene; and iii) a camera transform, which represents the viewpoint of the agent within the scene. The hierarchy of these components is setup such that the root is the parent of the avatar and the avatar is the parent of the camera, with any changes in the parent's orientation also affecting the child's. The camera transform is attached to the head, where a person's eyes would be.

System Initialization. Upon starting the simulation, the avatar is spawned at a predefined start point. Once tracking is established and the agent is in position, the system is calibrated to determine the transformation between the coordinate systems of the sensors, the avatar and the virtual scene. This allows for the translation of joints to be properly set.

Sensor Data Fusion. With each time step, input from the dance pad is applied as translation and vertical rotation to the root transform. Data from the Kinect is then used to set the avatar's pose. We start by defining the hips joint of the skeleton as the root and setting its transform relative to the avatar's root transform. Joint orientation data from the Kinect is then applied to the avatar's skeleton using forward kinematics. The precise position of the head and hands, as taken from the Oculus Rift and Leap Motion controller, are set in skeleton, with inverse kinematics being used to adjust the rest of the skeleton. The orientations of the individual finger joints are then taken from the Leap Motion and used to set the skeleton's fingers using forward kinematics. Through the above processes, an accurate body pose is fused from different sensors while prioritizing areas particularly crucial to object manipulation and user comfort.

3 Applications

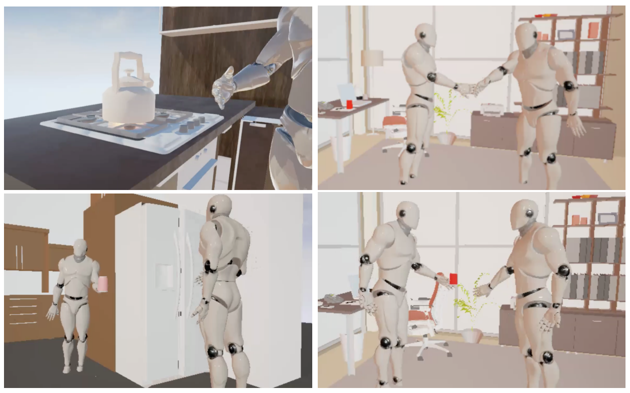


Figure 5: *Agents performing tasks. Top left: boil water; Bottom left: close fridge; Top right: shake hand; Bottom right: hand coke.*

We consider how our dataset can be used for robot task planning and large-scale image semantic segmentation. Some examples of the dataset are shown in Fig. 5.

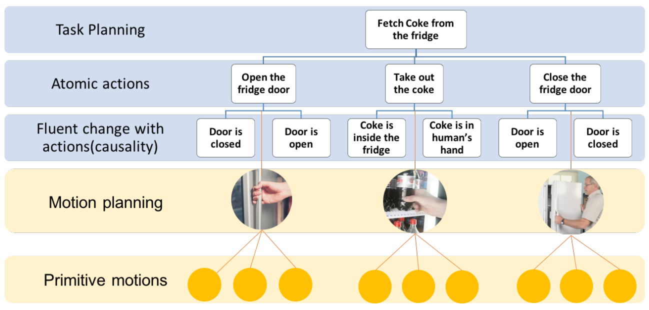
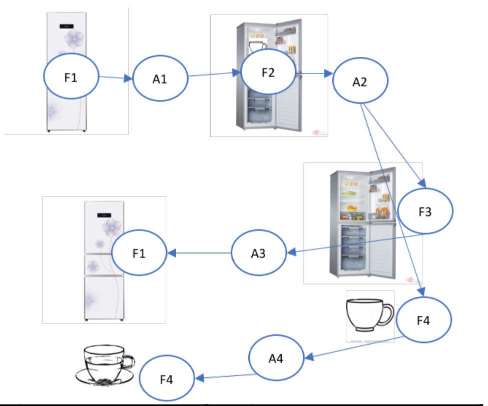


Figure 6: *Task illustration of opening a fridge. Blue: task planning recorded in VR. Yellow: Motion planning recorded by leap motion.*

Task planning requires that the agent understands how to decompose the high-level tasks into sub-tasks in a recursive fashion until reaching the level of atomic actions which are executed in an certain order (Fig. 6). Widely used schemes include Hierarchical Task Network (HTN) [Erol et al. 1994] and Markov Decision Process (MDP) [Kaelbling et al. 1998].



F1	Is_FridgeDoorOpen	A1	Open Fridge door
F2	Is_CupOutside	A2	Take out the cup
F3	Is_FridgeDoorClosed	A3	Close Fridge door
F4	Is_CupEmpty	A4	Pour water into the cup

Figure 7: *Object state change when opening a fridge. Top: object state changing with atomic actions; Bottom: notations.*

To define these atomic actions, we can take advantage of the concept of causality, or the relationships between actions and object states over action time [Fire and Zhu 2013; Li et al. 2016]. Fig. 7 illustrates the causality between atomic action and object state. Object proximity is used to define action time, as an action may only begin when the agent is close enough, and must end once the agent is too far away. Thus we can define an atomic action as a state transition that occurs while the agent is within some reasonable distance of the object. For motion planning, we need to understand what motions the agent is making in order to complete each atomic action. This is relatively straightforward to determine from our dataset, as we are already capturing joint-level orientation over time from the Leap Motion and Kinect and converting it into a single coordinate system for the sake of controlling the avatar. We can then use the time frames from the task planning data to map the appropriate motions to the appropriate task action.

Semantic Segmentation. Each object in the 3D scene is either manually segmented and labeled after reconstruction of the real scenes, or automatically labeled for synthetic scenes. By projecting the 3D scenes into 2D images together with their ground truth labels of objects, each pixel of the 2D images will come with a label, which forms the ground truth for image semantic segmentation. Compared to current computer vision datasets for semantic segmentation, which are manually segmented and labeled, our method would provide better precision on object boundaries which could potentially boost the performance using current deep learning methods. Furthermore, the 3D scenes can be projected onto arbitrarily many viewpoints at different points during agent-scene interaction, resulting in virtually unlimited training data for semantic segmentation.

4 Future Directions

In simulating interactions between the agent and surrounding objects, we consider two methods: physical simulation and hand pose detection. Physical simulation simulates forces between the agent and the object in order to affect some change in the object. This

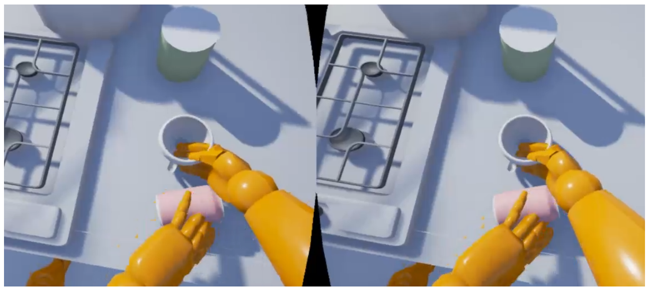


Figure 8: *Failure case in which the agent's hand penetrates the object's mesh.*

is closer to reality, but has difficulty simulating delicate grasping actions, and can result in hand-mesh penetration, which the physics system cannot reconcile, as shown in Fig. 8. Hand pose detection requires a predefined primitive motion be performed in the correct region relative to the object. We record these primitive actions in advance, and their data is registered with a margin of error. If the agent is detected using the appropriate motion in the correct region, then the object state will be updated accordingly. Hand pose detection is more reliable for actions that are difficult to physically simulate. However, as a result we lose all object-hand interactions, not just those which are physically impossible. The final system will likely use some mixture of these methods.

Our system can be further improved by making it more intuitive for the agent. While our current scene navigation system works, one that would allow the agent to walk around freely would be more natural for the agent. Methods exist which map virtual and physical spaces[Sun et al. 2016], however we would need to deal of the further constraint of maintaining Kinect tracking no matter how the agent moves. Another point where interactivity can be improved is the lack of force feedback for the human agent. Areas of further investigation would be some kind of haptic force feedback, or an augmented reality solution, where key objects have a physical representation [Azmandian et al. 2016] It would also be interesting extending the system to accommodate two or more agents, in which case investigating both verbal and nonverbal means of communication in VR is necessary [Li et al. 2015]

5 Conclusion

In this paper, we propose a system in which dynamic 3D scene models are imported into a physics-based engine and then combined with a virtual reality setup, thus allowing a human agent to interact intuitively with the virtual scene. In the process of generating these dynamic scenes, we investigate both constructing scenes from CAD models and segmenting whole scenes constructed via SLAM and other computer vision models.

Aside from allowing a human to interact with a physical scene, our system carries the added benefit of more detailed data on agent trajectory and hand pose. In using the Kinect for body tracking, we represent the agent not as a single point, but a series of joints in space. We also use the Leap Motion to acquire in detail human finger pose and generation. This allows us to generate fine-detailed dynamic manipulation data.

The applications of a robust virtual reality testing environment are also many, and can extend into remote human-robot interaction experiments as well as remote evaluation for robotics competitions.

References

ARGALL, B. D., CHERNOVA, S., VELOSO, M., AND BROWNING, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems 57*, 5, 469–483.

ARMENI, I., SENER, O., ZAMIR, A. R., JIANG, H., BRILAKIS, I., FISCHER, M., AND SAVARESE, S. 2016. 3d semantic parsing of large-scale indoor spaces. CVPR.

AZMANDIAN, M., HANCOCK, M., BENKO, H., OFEK, E., AND WILSON, A. D. 2016. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 1968–1979.

CHANG, A. X., FUNKHOUSER, T., GUIBAS, L., HANRAHAN, P., HUANG, Q., LI, Z., SAVARESE, S., SAVVA, M., SONG, S., SU, H., ET AL. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

CHOI, S., ZHOU, Q.-Y., AND KOLTUN, V. 2015. Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 5556–5565.

DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, IEEE, 248–255.

DURRANT-WHYTE, H. F. 1988. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation 4*, 1, 23–31.

EROL, K., HENDLER, J., AND NAU, D. S. 1994. Htn planning: Complexity and expressivity. In *AAAI*, vol. 94, 1123–1128.

FIRE, A. S., AND ZHU, S.-C. 2013. Learning perceptual causality from video. In *AAAI Workshop: Learning Rich Representations from Low-Level Sensors*.

HANDA, A., PATRAUCEAN, V., BADRINARAYANAN, V., STENT, S., AND CIPOLLA, R. 2015. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*.

HANDA, A., PĂTRĂUCEAN, V., BADRINARAYANAN, V., STENT, S., AND CIPOLLA, R. 2016. Understanding real world indoor scenes with synthetic data.

KAELBLING, L. P., LITTMAN, M. L., AND MOORE, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research 4*, 237–285.

KAELBLING, L. P., LITTMAN, M. L., AND CASSANDRA, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*, 1, 99–134.

KRAGIC, D., CHRISTENSEN, H. I., ET AL. 2002. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv 15*.

LEVINE, S., PASTOR, P., KRIZHEVSKY, A., AND QUILLEN, D. 2016. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*.

LI, H., TRUTOIU, L., OLSZEWSKI, K., WEI, L., TRUTNA, T., HSIEH, P.-L., NICHOLLS, A., AND MA, C. 2015. Facial performance sensing head-mounted display. *ACM Transactions on Graphics (TOG) 34*, 4, 47.

LI, B., WU, T., XIONG, C., AND ZHU, S.-C. 2016. Recognizing car fluents from video. *arXiv preprint arXiv:1603.08067*.

LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*, Springer, 740–755.

MILLER, A. T., AND ALLEN, P. K. 2004. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine 11*, 4, 110–122.

NEHANIV, C. L., AND DAUTENHAHN, K. 2007. *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions*. Cambridge University Press.

SHIMOGA, K. B. 1996. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research 15*, 3, 230–266.

SHOTTON, J., SHARP, T., KIPMAN, A., FITZGIBBON, A., FINOCCHIO, M., BLAKE, A., COOK, M., AND MOORE, R. 2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM 56*, 1, 116–124.

SHU, T., RYOO, M. S., AND ZHU, S.-C. 2016. Learning social affordance for human-robot interaction. In *IJCAI*.

SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. 2012. Indoor segmentation and support inference from rgbd images. In *ECCV*, Springer, 746–760.

SUN, Q., WEI, L.-Y., AND KAUFMAN, A. 2016. Mapping virtual and physical reality. *ACM Transactions on Graphics (TOG) 35*, 4, 64.

WEICHERT, F., BACHMANN, D., RUDAK, B., AND FISSELER, D. 2013. Analysis of the accuracy and robustness of the leap motion controller. *Sensors 13*, 5, 6380–6393.

XIAO, J., OWENS, A., AND TORRALBA, A. 2013. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 1625–1632.

ZHU, Y., JIANG, C., ZHAO, Y., TERZOPOULOS, D., AND ZHU, S.-C. 2016. Inferring forces and learning human utilities from videos. In *CVPR*, 3823–3833.