RFUniverse: A Physics-based Action-centric Interactive Environment for Everyday Household Tasks

Haoyuan Fu\textsuperscript{1}, Wenqiang Xu\textsuperscript{1}, Han Xue\textsuperscript{1}, Huinan Yang\textsuperscript{2}, Ruolin Ye\textsuperscript{1}, Yongxi Huang\textsuperscript{1}, Zhendong Xue\textsuperscript{1}, Yanfeng Wang\textsuperscript{1} and Cewu Lu\textsuperscript{1}

Abstract—Household environments are important testbeds for embodied AI research. Many simulation environments have been proposed to develop learning models for solving everyday household tasks. However, though interactions are paid attention to in most environments, the actions operating on the objects are not well supported concerning action types, object types, and interaction physics. To bridge the gap at the action level, we propose a novel physics-based action-centric environment, RFUniverse, for robot learning of everyday household tasks. RFUniverse supports interactions among 87 atomic actions and 8 basic object types in a visually and physically plausible way. To demonstrate the usability of the simulation environment, we perform learning algorithms on various types of tasks, namely fruit-picking, cloth-folding and sponge-wiping for manipulation, stair-chasing for locomotion, room-cleaning for multi-agent collaboration, milk-pouring for task and motion planning, and bimanual-lifting for behavior cloning from VR interface. Client-side Python APIs, learning codes, models, and the database will be released. Demo video for atomic actions can be found in supplementary materials: \url{https://sites.google.com/view/rfuniverse}

I. INTRODUCTION

It seems household tasks are endless in daily life, but they can be regarded as combinations of finite atomic actions interacting with different kinds of objects. In this way, action is viewed as a tuple $A = \langle \text{action}, \text{object}, \text{constraints} \rangle$, and a task is thus an action sequence $T = \{A_1, ..., A_n\}$. As learning from real-world domestic environments is cost-prohibitive, many researchers proposed simulation environments for household tasks \cite{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}. Despite that all of them provide "interactive" functionalities, the supports for the agent actions are limited in three aspects, namely action types, object types, and interaction physics.

In this work, we first determine what action and object types are important for daily household tasks. We summarize from a previous proposed human activity knowledge base in household environment \cite{4}. As a result, we extract 14 navigation-related (e.g. turn_left, look_at), 3 locomotion-related (i.e. walk, run, climb) and 70 manipulation-related atomic actions (i.e. 33 single-hand, 23 bimanual, 14 tool-based), which concerns 5 basic object types (i.e. rigid, articulated, flexible, transparent, and tearable). Besides, 3 additional object types namely fluid, gas, fire can be used to represent the visual object states (e.g. steams from boiled water in Fig. 2). Based on the actions, we present a novel Unity-based simulation environments named **RFUniverse**, which organizes different physics engines to enable the physics-based interaction of all the atomic actions upon different object types.

From the action side, previous environments generally built around simple atomic actions like navigation \cite{2, 6}, grasping, transporting, and placing (or dropping) \cite{1, 7, 8, 10}. However, for some environments \cite{1, 14}, even...
these simple actions are usually implemented in an abstract, not physics-based way. For example, an abstract grasping is performed by attaching the target object like suctioning when approaching it within a certain distance. As for complex actions like cutting are usually implemented in the “pre-defined” manner [9], [12]. For example, iGibson 2 [9] prepares sliced meshes in advance, so that the object will be cut off no matter where the knife is contact. VRKitchen [12] defines the part to be removed by heuristic rules. Nevertheless, the lack of supporting physics-based actions will limit the task ranges, and constrain the motion policy learned for complex actions.

From the object side, as mentioned earlier, there are at least 8 basic types of objects that are relevant to the household tasks. By “basic”, we mean these properties can be combined, like a plastic bag which is flexible, tearable and transparent. Considering interaction with the basic object types, the previous simulators usually support the rigid or articulated [1], [9] objects as the actionable objects. Though some environments can simulate dynamics of flexible objects [15], but the agent cannot operate on the flexible objects due to the limitation of deployed physics engines. The tearable and transparent properties are individually studied by different researchers [16], [17], but they have never been implemented to interact with the virtual embodied agent in a physically plausible way.

Aside from the key features regarding physics-based interaction, RFUniverse also provides full functionalities to support simulation and learning of robots doing household tasks: Python APIs, photo-realistic rendering, multi-modal sensing, synthetic data generation for perception models, and gym-like wrapper for reinforcement learning models. A powerful ROS-free motion planner, RFMove [18] is natively integrated to plan for full-body movements. Moreover, we provide a VR interface to extend the interactive ability from the real to the simulated world.

We evaluate the usability of RFUniverse with various kinds of tasks related to household, namely fruit-picking for rigid object manipulation, cloth-folding for 2D flexible object manipulation, sponge-wiping for 3D flexible object manipulation, stair-chasing for locomotion, room-cleaning for multi-agent collaboration, milk-pouring for task and motion planning, bimanual-lifting via VR interface for behavior cloning.

We summarize our contributions as follows:

- We construct a simulation environment RFUniverse, which can support physics-based interactions for all the extracted action and object types. It also provides a client-server communication framework based on gRPC, which can enable full functionality control of Unity with Python language.
- We benchmark 7 kinds of learning tasks on RFUniverse, which cover different aspects of the household tasks.

II. RELATED WORKS

Physics-based Object Simulation Physics-based action requires the reaction of manipulated objects should also be physically plausible. We roughly categorize objects concerned in household tasks as rigid, articulated, flexible, tearable, transparent, fluid, gas, and fire.

For the latter three object types, as discussed earlier, only the visual effects are concerned. They can be simulated by particle system implemented by Flex [19].

For rigid and articulated objects, they can generally be properly simulated by common physics engines like Dart [20], ODE [21], Bullet [22], MuJoCo [23] and PhysX [24]. MuJoCo is arguably the best for articulation [25], but it is close-sourced. Though it provides an integration to Unity, we find the default PhysX can also do well.

Simulation on flexible objects is more tricky, as their deformation mechanisms are very different according to the materials [26]. Thus though many common physics engines [23], [22] claim to support 1D (rope), 2D (cloth), and 3D (soft body) flexible object deformation, they either adopt simple models [27] to simulate or provide limited controlling parameters. Sofa framework [28] can simulate the deformation with accurate modeling for different materials, but the Unity integration plugin is premium. Aside from simulating the flexible object with one physics engine, we can also deal with them separately. Some physics engines have specialties in simulating cloth [29], [30] and some can simulate rope and soft body [31], [19].

The difficulty in simulating the tearable object is the fracture modeling [16]. Since real-time fracture modeling and rendering is challenging, we model it with simple topology modification as described in [28].

Recently, MPM-based approaches [32], [33], [16] shed a light on simulating different object types within the same framework. Though MPM-based methods can provide visually plausible effects for the interactions, the force applied on the object is hard to be obtained accurately, which may limit the applications on force-concerned tasks.

Transparent objects are objects with unique optical properties, which can be handled by scripting shaders in different kinds of rendering pipelines in Unity.

Simulated Environments for Household Tasks In comparison with typical reinforcement learning environments [34], [35], [36], [37], the household-oriented simulated environments feature navigation in visually plausible scenes and interaction with realistic object and scene geometry.

An important line of household environments is based on AI2THOR [1]. It is developed upon Unity and provides Python APIs to control the asset loading and agent behaviors. In AI2THOR, the objects can only be grasped abstractly. Such property is inherited to the following RoboTHOR [11], ManipulaTHOR [10] or the environments [38], [5] proposed for the language-vision researches, like language-guided navigation and instruction following.

For those non-AI2THOR-based environments, Virtual-Home [4] and ThreeDWorld [15] also adopt Unity as the rendering system. The former takes a study on human activities in domestic environments, resulting in a knowledge base. The key difference between their knowledge base and ours is whether the task describing ”what I would do in
TABLE I

| Environment    | Obj. Types | # Actions | Physics Engine | Colli. Inter. | Integrated Planner | VR |
|----------------|------------|-----------|----------------|---------------|---------------------|----|
| HoME [3]       | R          | 4 + 3     | Bullet         | ✓             | /                   | /  |
| AI2THOR series [1], [11], [10] | R,A,Fu     | 10 + 10   | Bullet         | /             | /                   | /  |
| Habitat [6]    | R          | 3 + 0     | Bullet         | ✓             | /                   | /  |
| SAPIEN [13]    | R          | /         | Bullet         | ✓             | /                   | ✓  |
| VirtualHome [4] | R,A,Fu     | 9 + 7     | PhysX          | /             | /                   | /  |
| VRKitchen [12] | R,A,Fu     | 0 + 18    | PhysX          | /             | /                   | ✓  |
| ThreeDWorld [15], [14] | R,A,F,Fu | 4 + 2     | PhysX, Flex    | /             | ✓                   | ✓  |
| Gibson series [2], [7], [8], [9] | R,A,F,Fu | 4 + 4     | Bullet         | ✓*            | /                   | ✓  |
| RFUniverse     | R,A,F,Fu,Te,Ti,G,Ft | 17 + 70   | PhysX, Flex, Cloth Dynamics, Obi, [MuJoCo, Bullet, SOFA] | ✓             | ✓                   | ✓  |

Comparison with different simulation environments. As some environments are continuously developed, we summarize them into a series and the features are reported based on the latest version.

Obj. Types: R - Rigid, A - Articulated, F - Flexible, Fu - Fluid, Te - Tearble, Tr - Transparent, G - Gas, Fr - Fire.

# Actions: n1 + n2, where n1 means atomic actions for navigation/locomotion, and n2 means actions for manipulation.

Physics Engine: engines in [ ] means these engines are optional with APIs but not used in the benchmark experiments.

Colli. Inter.: Whether the agent-object interaction are collision-based. Some environment can support the collision-based interaction, but they choose not to. ✓* means not all actions are collision-based, for example cutting in iGibson2[9].

On the other hand, instead of adopting mature rendering and physics engine management system like Unity, some environments decide to write their own. Habitat [6] and Gibson [2] support only navigation-related tasks. Later, the Gibson series [7], [8], [9] have been evolved from a pure navigation environment to support simple manipulation and object states. The latest iGibson2.0 supports cutting aside from the typical grasping, transporting and dropping, but in a pre-defined way.

The functionality comparison between RFUniverse and previous environments can be referred to Table I.

III. RFUniverse Simulation Environment

A. Python Interface & Communication Framework

RFUniverse adopts a client/server framework to communicate the client-side Python-based program and the Unity-side C#-based server, as shown in Fig. 1. The communication is based on gRPC, which can handle different programming languages, OS platforms, and networks. We adopt Python as the client language because of its simplicity and powerful ecosystem, especially with learning frameworks [39], [40]. The user can control or obtain any information through the Python interface. For example, to enable physics-based action for the benchmark tasks, we can complete all the procedures from the Python interface:

- Establish a connection to the Unity server;
- Setup a scene layout based on the sampling heuristics or configuration files;
- Configure the rendering options, so that can adjust the computational resources for rendering or generate synthetic datasets;
- Configure the physics options for physics engines;
- (optional) Collecting Data through VR Interface;
- Read the states, observations from the given camera(s) in the simulation;
- Control the agent with agent id, so that enables multi-agent feature.

Fig. 1. The framework of RFUniverse.

B. Task Layout Configuration

Scene Assets We adopt the scene assets from the Gibson dataset, as it is scanned from the real world. We make some fixes upon these scenes like installing the windows, doors, and mirrors. But we leave the scanning error unfixed as we think it is part of the scanning process.

Object Assets To accompany the tasks, the objects are selected from 3D warehouse[1]. For transparent objects, we attach the glass-like material with a customized shader to the original object model. The transparent effects can be presented in different rendering pipelines (i.e. SRP, URP, HDRP) in Unity. For each object, we can query the object states (e.g. object pose, applied force) and attributes (e.g. mass, inertial, index of refraction) during interaction through Python API.

1https://3dwarehouse.sketchup.com/
**Agent Assets** Different household tasks may require different workspace and dexterity for the agent. While currently, to the best of our knowledge, there exists no single robot agent that can handle all kinds of household tasks. Thus, we support various robot agents. In the benchmark experiments, for manipulation tasks which can be accomplished by a single arm with a fixed base, the agent is Franka Emika Panda; for the multi-agent task, we select Philips FC 8800 Robotic Floor Cleaner for room-cleaning; for the locomotion task which requires leg like stair-chasing, we adopt a human boy avatar; for the task and motion planning, and behavior cloning of bimanual grasping, we propose a dual-arm mobile manipulator, Tobor. The design and the hardware specification of the Tobor robot can be referred to in supplementary materials. Besides, other common robots like UR5, Fetch, shadow hands can be trivially supported.

**Layout Generation** The layout can be generated in two ways, through an interactive GUI or sampling heuristics.

- Through an interactive GUI like blender, Unity, or any other proper 3D software, we can output the name and position of scene/object to a configuration file. Then we can reconstruct the scene configuration to the Unity server through Python API.
- On the other hand, we can also specify different sampling heuristics on objects selection, initial positions, fixed or falling from Python API.

**C. Rendering System**

In this work, we are particularly interested in simulating convincing physics-based visual effects which can reflect the object states. As mentioned earlier, 4 kinds of objects are related to visual effects, namely the fluid, gas, fire, and transparent objects. Thanks to the highly customizable rendering pipeline of Unity, we can simulate convincing visual effects as shown in Fig. 2. The trigger conditions for these visual effects are labeled in the object assets, for example, when the water is contained in a pot (container), and the pot is being heated by fire, the boiling and steam effects will be simulated.

Besides, the lighting conditions are also important, especially when they can change the appearance of the RGB snapshots, which is critical to the synthetic dataset generation and domain randomization. These two functionalities are also provided in RFUniverse, but they are not used in the benchmark tasks, thus the descriptions are left in the supplementary materials.

**D. Physics System**

Our key motivation in this work is to support atomic actions in a physics-based manner. To achieve this, the underlying physics engine should support plausible interaction effects for different object types. However, existing physics engines are built for different purposes, not a single one aims for full physics simulation. A workaround is to simulate different interaction behaviors with different physics engines. Thanks to the strong ecosystem of Unity, which can enable us to connect with different backend physics engines for different object types. In practice, we choose PhysX [24] for rigid and articulated objects, Obi [31] for rope and soft body, Cloth Dynamics [30] for cloth simulation, and Flex [19] for fluid, gas, fire effect simulation. The tearing effect can be achieved in real-time and the separated meshes can be calculated on the fly with Mesh Slicer [41]. Besides, we also provide wrappers for MuJoCo 1.55 [23], Bullet [22] and SOFA [28].

**E. Gym-like Wrapper for Reinforcement Learning**

As for reinforcement learning algorithms, we provide a standard gym-like wrapper for different kinds of environments, we will later show how to run the manipulation, locomotion with standard reinforcement learning library, stable-baselines3 [42] in Sec. IV.

**F. Task and Motion Planner**

We integrate a TAMP framework based on PDDLStream [43]. If not specified, the action stream samplers are implemented by RFMove [18], a motion planning framework without the requirement of ROS [44] installation or models learned from reinforcement learning. For navigation, Unity’s NavMesh framework is adopted.

**G. VR Interface**

The VR interface provides the opportunity for the user to directly interact with the simulated environment so that we can collect demonstrations in the virtual environment. Different VR devices can be connected to RFUniverse through SteamVR, we take the HTC Vive headset with Noitom Hi5 glove as an example, as shown in Fig. 3.

**IV. EXPERIMENTS**

In this section, we will perform 5 benchmark tasks, including 3 manipulation tasks (i.e fruit-picking for rigid, cloth-folding for 2D flexible, and sponge-wiping for 3D
flexible object manipulation), 1 locomotion task (i.e. stair-chasing), and 1 multi-agent task (i.e. room-cleaning). The other 2 tasks for TAMP (i.e. milk-pouring) and behaviour cloning from VR demonstration (i.e. bimanual-lifting) are discussed in supplementary materials due to page limit.

A. Machine Specification

During training, all experiments are benchmarked on Ubuntu 20.04 platform with two Intel(R) Xeon(R) Platinum 8276 CPU and 1 NVIDIA Geforce RTX 3080 graphic card. Due to different calculating burden in various tasks, we dynamically adjust the action time step. We set the action time step for rigid object interaction $t_a^s = \frac{5}{36} s$ and for deformable object interaction $t_a^d = \frac{1}{3} s$. We raise time scale to accelerate training process and reach physical time step $t_a^e = \frac{1}{30} s$ and $t_a^d = \frac{1}{30} s$.

B. Reinforcement Learning of Manipulation Tasks

1) Task Description: The fruit-picking is a standard rigid object manipulation task. The agent is trained to pick up rigid fruits on the table and place them into a basket. During training, we randomly place one fruit on the table and select a position for the basket in each episode. The task is judged to be finished when the given fruit is placed into the basket.

For cloth-folding, we place a random-state cloth into the workspace of the agent. The task is to fold the cloth towards a given configuration. The agent must grasp a corner of the cloth and place it somewhere proper so that the cloth can be folded. To better benchmark this task, in each episode, we randomly select two corners and treat the task as a success if two chosen corners are placed within 5cm distance threshold.

For sponge-wiping, we place a sponge on a table and train the agent using this sponge to wipe the table with grasp push. In each episode, the sponge will be placed at a random location, and a target position for the sponge is also selected. The final success decision is made when the sponge is moved to the target position within a 5cm tolerance. We track the position of the sponge and determine a failure if it is moved above the table.

2) Implementation: In all manipulation tasks mentioned above, we adopt Hindsight Experience Replay (HER) [45] algorithm to accelerate sampling efficiency with off-policy Truncated Quantile Critics (TQC) [46] algorithm as the base model. We set 50 time steps for each episode. In all three environments, we wrap the task into the goal-based condition and select sparse reward function, which is aimed to train the agent to finish the task as soon as possible. The observation includes the Panda gripper position, velocity, and open width, together with the object’s current position, orientation, velocity, angular velocity, and the target position. We train this agent for $1000k$ time steps ($20k$ episodes). Figure 4 (a)(b)(c) depict the relationship between manipulation task success rate and time steps during training.

C. Locomotion Task

1) Task Description: To demonstrate the use of RFUniverse for locomotion tasks, we train an articulated human-like agent to chase the toy on the stairs. The task is denoted as Stair-chasing. The agent is copied from ML-Agents [40] Walker environment which has 16 DoFs and walks based on the friction between his feet and ground. In each episode, the agent is initialized at a fixed position, but facing a random orientation, with a target toy randomly fallen on the stairs. The agent needs to balance its body under physics to avoid any part (except feet) colliding with ground or stairs since this collision will return a reward of $−1$ and end this episode. If the agent touches the target toy, it will get a reward of 1. To better organize the agent’s movement, we set a target velocity vector $v^*$ where $||v^*||$ is fixed and the agent will be trained to achieve this velocity. In each time step, if the agent keeps its balance without falling, it will get an extra reward $r$ combining:

- The distance between actual velocity vector $v$ and $v^*$.
- The difference between heading orientation unit vector $o$ and target orientation unit vector $o^*$.

The reward function is

$$r = \left[1 - \left(\text{clamp}(||v^* - v||, 0, ||v^*||) \right)^2 \right] \times o^* \cdot o + 1$$

2) Implementation: We use Soft Actor-Critic (SAC) [47] algorithm to train the agent for $1.5e7$ time steps with a maximum episode length setting to 5000 in this task. It should be noticed that we only provide observation of the agent’s position, velocity, and orientation relative to the target toy as well as the joint parameters of the agent. Based on the Markov property of RL, we make the world’s states blind to agents, so that agents will not only learn how to walk but also be trained with self-adaption from hitting stairs. The relationship between training episodic reward and time steps is shown in Figure 4[d].

D. Multi-agent Collaboration Task

1) Task Description: To verify the multi-agent support of RFUniverse, we setup a benchmark room-cleaning which simulates the working process of sweeping robot in household. Three identical sweeping robots are placed into a room, which is divided into $8 \times 8$ grids. If a sweeping robot’s center of mass passes a grid, the area inside that grid is regarded as being cleaned with its color turning from grey to original color of floor. The agents’ task is to clean as many grids as possible in given 100 time steps, but they should avoid colliding with each other. Since this is a multi-agent task, we take each agent’s position and velocity as shared observation.
In each time step, each agent will receive a 2-d vector as representation of force in $X$-axis and $Y$-axis. As for the reward, if there’s collision among agents, an episode will end at once and return a reward of $-50$. If there’s no collision, reward will be the sum of agents’ average velocity and the number of grids been cleaned.

2) Implementation: We use SAC algorithm to train for $1e6$ time steps with maximum episodic length as $100$ time steps. The relationship between episodic reward and time steps is shown in figure 4(e).

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we present a physics-based action-centric simulation environment, RFUniverse, for household tasks. RFUniverse supports 87 atomic actions and 8 object types which are summarized from [4]. It has been evaluated for different kinds of manipulation, locomotion and navigation tasks. In the future, with RFUniverse as the foundational platform, we are interested in integrating multi-modal perception into the action models to further extend the adaptive ability of robots.

REFERENCES

[1] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, “AI2-THOR: An Interactive 3D Environment for Visual AI,” arXiv, 2017.

[2] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: Real-world perception for embodied agents,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 9068–9079.

[3] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville, “HoMe: a Household Multimodal Environment,” in NIPS 2017’s Visually-Grounded Interaction and Language Workshop, Long Beach, United States, Dec, 2017. [Online]. Available: https://hal.inria.fr/hal-01653037

[4] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, “Virtualhome: Simulating household activities via programs,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8494–8502.

[5] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, “ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020, [Online]. Available: https://arxiv.org/abs/1912.01734

[6] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Kolmun, J. Malik, D. Parikh, and D. Batra, “Habitat: A Platform for Embodied AI Research,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[7] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Tohee, R. Martin-Martín, and S. Savarese, “Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 713–720, 2020.

[8] B. Shen, F. Xia, C. Li, R. Martin-Martín, L. Fan, G. Wang, S. Buch, C. D’Arpino, S. Srivastava, L. P. Tchapmi, K. Vainio, L. Fei-Fei, and S. Savarese, “igibson, a simulation environment for interactive tasks in large realistic scenes,” arXiv preprint, 2020.

[9] C. Li, F. Xia, R. Martin-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, C. K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese, “igibson 2.0: Object-centric simulation for robot learning of everyday household tasks,” 2021.

[10] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kemhavi, and R. Mottaghi, “ManipulaTHOR: A Framework for Visual Object Manipulation,” in CVPR, 2021.

[11] M. Deitke, W. Han, A. Herrasti, A. Kemhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, “RoboTHOR: An Open Simulation-to-Real Embodied AI Platform,” in CVPR, 2020.

[12] X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S. Zhu, “Vrkitchen: an interactive 3d virtual environment for task-oriented learning,” arXiv, vol. abs/1903.05757, 2019.

[13] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, “SAPIEN: A simulated part-based interactive environment,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.
C. Gan, S. Zhou, J. Schwartz, S. Alter, A. Bhandwaldar, D. Gutfreund, D. L. K. Yamins, J. J. DiCarlo, J. McDermott, A. Torralba, and J. B. Tenenbaum, “The three-world transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied agents,” 2021.

C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. D. Freitas, J. Kiblits, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, D. Mrowca, M. Lingelbach, A. Curtis, K. Feigelis, D. M. Bear, D. Gutfreund, D. Cox, J. J. DiCarlo, J. McDermott, J. B. Tenenbaum, and D. L. K. Yamins, “Threedworld: A platform for interactive multimodal physical simulation,” 2020.

J. Wolper, Y. Chen, M. Li, Y. Fang, Z. Qu, J. Lu, M. Cheng, and C. Jiang, “Anisompnm: Animating anisotropic damage mechanics,” ACM Trans. Graph., vol. 39, no. 4, 2020.

S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song, “Clear grasp: 3d shape estimation of transparent objects for manipulation,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3634–3642.

W. X. Yongxi Huang, Danqing Li, “Rfmove.” [Online]. Available: https://github.com/mvig/robotflow/rfmove

J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, “Gpu-accelerated robotic simulation for distributed reinforcement learning.” 2018.

J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. Srinivasa, M. Stilman, and K. L. Liu, “Dart Dynamic animation and robotics toolkit,” Journal of Open Source Software, vol. 3, no. 22, p. 500, 2018. [Online]. Available: https://doi.org/10.21105/joss.00500

R. Smith, “Open dynamics engine,” 2008, http://www.ode.org/. [Online]. Available: http://www.ode.org/

E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.

E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012, pp. 5026–5033. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6386109

NVIDIAGamesWorks, “Nvidiaframeworks/physics: Nvidia physics sdk.” [Online]. Available: https://github.com/NVIDIAGamesWorks/Physics

T. Erez, Y. Tassa, and E. Todorov, “Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physics,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4397–4404.

E. Sifakis and J. Barbic, “Fem simulation of 3d deformable solids: A practitioner’s guide to theory, discretization and model reduction,” in ACM SIGGRAPH 2012 Courses, ser. SIGGRAPH ‘12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2343483.2343501

D. Baraff and A. Witkin, “Large steps in cloth simulation,” in Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 43–54. [Online]. Available: https://doi.org/10.1145/280814.280821

F. Faure, C. Duriez, D. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bouquet, I. Peterlik, and S. Cotin, “SOFA: A Multi-Model Framework for Interactive Physical Simulation,” in Soft Tissue Biomechanical Modeling for Computer Assisted Surgery, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, June 2012, vol. 11, pp. 283–321. [Online]. Available: https://hal.inria.fr/hal-00681539

NVIDIAGamesWorks, “Nvidiagameworks/nvcloth.” [Online]. Available: https://github.com/NVIDIAGamesWorks/NvCloth

UnityStore. [Online]. Available: https://assetstore.unity.com/packages/tools/physics/cloth-dynamics-19448

V. M. Studio. [Online]. Available: http://obi.virtualmethodstudio.com/

Y. Fang, Z. Qu, M. Li, X. Zhang, Y. Zhu, M. Anjaneya, and C. Jiang, “Iq-nmpm: An interface quadrature material point method for non-sticky two-way coupled nonlinear solids and fluids,” ACM Transactions on Graphics, 2020.

J. Wolper, Y. Fang, M. Li, J. Lu, M. Gao, and C. Jiang, “Cd-nmpm: Continuum damage material point method for dynamic fracture animation,” ACM Trans. Graph., vol. 38, no. 4, July 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322949

L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, “Surreal: Open-source reinforcement learning framework and robot manipulation benchmark,” in Conference on Robot Learning, 2018.

Y. Fang, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” in Conference on Robot Learning (CoRL), 2019. [Online]. Available: https://arxiv.org/abs/1910.10897

S. James, Z. Ma, D. Rovic, Arrojo, and A. J. Davison, “Ribench: The robot learning benchmark & learning environment,” IEEE Robotics and Automation Letters, 2020.

D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Farhadi, “Iqa: Visual question answering in interactive environments,” in Computer Vision and Pattern Recognition (CVPR), 2018. IEEE Conference on, 2018.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, et al., “Unity: A general platform for intelligent agents,” arXiv preprint arXiv:1809.06227, 2018. [Online]. Available: https://github.com/Unity-Technologies/ml-agents

A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, “Controlling overestimation bias with truncated mixture of continuous distributional qualtile critics,” in International Conference on Machine Learning, PMLR, 2020, pp. 5556–5566.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in International conference on machine learning. PMLR, 2018, pp. 1861–1870.