2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Gen2Sim: Scaling up Robot Learning in Simulation with Generative Models

Pushkal Katara*, Zhou Xian*, Katerina Fragkiadaki

Carnegie Mellon University

https://gen2sim.github.io/

***Abstract*— Generalist robot manipulators need to learn a wide variety of manipulation skills across diverse environments. Current robot training pipelines rely on humans to provide kinesthetic demonstrations or to program simulation environments and to code up reward functions for reinforcement learning. Such human involvement is an important bottleneck towards scaling up robot learning across diverse tasks and environments. We propose *Gen*eration to *Sim*ulation (*Gen2Sim*), a method for scaling up robot skill learning in simulation by automating generation of 3D assets, task descriptions, task decompositions and reward functions using large pre-trained generative models of language and vision. We generate 3D assets for simulation by lifting open-world 2D object-centric images to 3D using image diffusion models and querying LLMs to determine plausible physics parameters. Given URDF files of generated and human-developed assets, we chain-of-thought prompt LLMs to map these to relevant task descriptions, temporal decompositions, and corresponding python reward functions for reinforcement learning. We show Gen2Sim succeeds in learning policies for diverse long horizon tasks, where reinforcement learning with non temporally decomposed reward functions fails. Gen2Sim provides a viable path for scaling up reinforcement learning for robot manipulators in simulation, both by diversifying and expanding task and environment development, and by facilitating the discovery of reinforcement-learned behaviors through temporal task decomposition in RL. Our work contributes hundreds of simulated assets, tasks and demonstrations, taking a step towards fully autonomous robotic manipulation skill acquisition in simulation.**

I. Introduction

Scaling up training data has been a driving force behind the recent revolutions in language modeling [1], image understanding [2], speech recognition [3], image generation [4], to name a few. This begs the question: can we scale up robot data to enable a similar revolution in robotic skill learning? One way to scale robot data is in the real world, by having multiple robots explore [5] or by having humans provide kinesthetic demonstrations [6], [7], [8]. This is a promising direction; however, safety concerns and wear and tear of the robots hinder robot exploration in the real-world, and collecting kinesthetic demonstrations scales poorly as it is time-consuming and labor-intensive [8]. Another way to scale robot data is in simulation, by developing simulated environments, defining tasks and their reward functions, and training robot policies with reinforcement learning, augmenting visuals and physics parameters to facilitate transfer of policies to the real world [9]. Such sim2real paradigm has seen recent successes in robot locomotion [10], [11], [12], [13], object re-orientation [14], [15], and drone flight [16]. These examples, though very important and exciting, are still fairly isolated.

A central bottleneck towards scaling up simulation environments and tasks is the laborious manual effort needed for developing the visuals and physics of assets, their spatial arrangement and configurations, the development of task definition and reward functions, or the collection of programmatic demonstrations. Tremendous resources have been invested in developing simulators for autonomous vehicles [17], warehouse robots, articulated objects [18], home environments [19], [20], [21], etc., many of which are proprietary and not open-sourced. Given these considerations, an important question naturally arises: How can we minimize manual effort in simulation development for diverse robotic skill learning?

In this paper, we explore automating the development of simulation environments, manipulation tasks and rewards for robot skill learning, by building upon latest advances in large pre-trained generative models of images and language. Our system strives to automate all stages of robot learning: from generating 3D assets, textures, and physics parameters, to generating task descriptions and reward functions, leading to automated skill learning in diverse scenarios, as shown in Figure 1. This generative pipeline was first proposed in a recent position paper [22], described as a promising pathway towards generating diverse data for generalist robot learning. In this paper, we present Gen2Sim, the first attempt and realization of such a generative paradigm. We automate 3D object asset generation by combining image diffusion models for 3D mesh and texture generation, and LLMs for querying physical parameters information. We showcase how LLMs and image generative models can diversify the appearances and behaviors of assets by producing plausible ranges of textures, sizes and physical parameters, achieving "intelligent" domain diversification. We automate task description, task decomposition and reward function generation by few-shot prompting of LLMs to generate language descriptions for semantically meaningful tasks, concerning affordances of existing and generated 3D assets, articulated or not,

* Equal contribution.

The authors are with School of Computer Science, Carnegie Mellon University <pkatara, xianz1, katef>@cs.cmu.edu

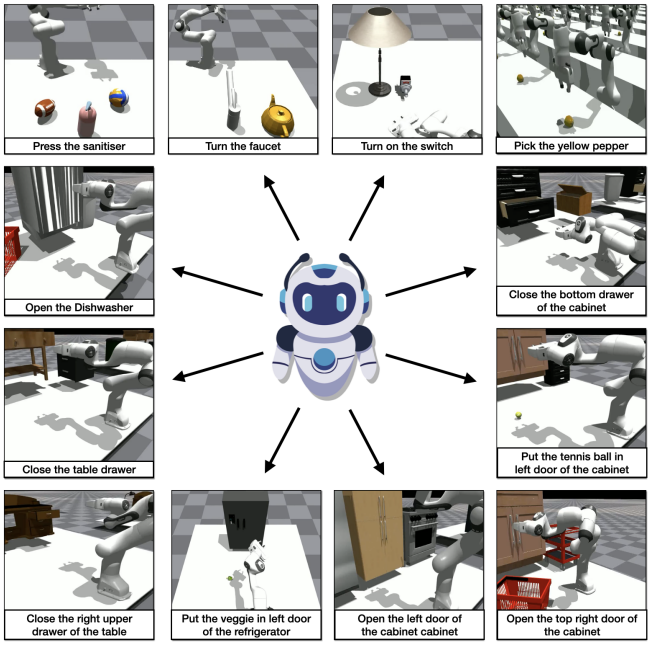

Fig. 1: **Gen2Sim** is an automated generative pipeline of assets, tasks, task decompositions, and rewards functions for autonomous robotic skill reinforcement learning in simulation. Here we show 12 generated tasks, concerning affordances of diverse types of object assets and their combinations.

alongside their reward functions. Gen2Sim is able to generate numerous object assets and task variations without any human involvement beyond few LLM prompt designs. We successfully train RL policies using our auto-generated tasks and reward functions. We also demonstrate the usefulness of our simulation-trained policies, by constructing digital-twin environments from given real scenes, allowing a robot to practice skills in the twin simulator and deploying it back to the real world to execute the task.

In summary, we make the following contributions:

- We show how pre-trained generative models of images and language can help automate 3D asset generation and diversification, task description generation, task decomposition and reward function generation that supports reinforcement learning of long horizon tasks in simulation with minimal human involvement.
- We deploy our method to generate hundreds of assets, and hundreds of manipulation tasks, their decompositions and their reward functions, for both human-developed and automatically generated object assets.

For code, videos and qualitative video results, please visit our project website: `https://gen2sim.github.io/`.

II. Related Work

Large Language Models for task and motion planning in robotics Large language models (LLMs) map instructions to language subgoals [23], [24], [25], [26] or action programs [27] with appropriate plan-like or program-like prompts. LLMs trained from Internet-scale text have shown impressive zero-shot reasoning capabilities for a variety of downstream language tasks [1] when prompted appropriately, without any weight fine-tuning [28], [29], [30], [31]. LLMs were used to generate task curricula and predict skills to execute in Minecraft worlds [32], [33], [34] Following the seminal work of Code as Policies, many works map language to programs over given skills [35] or hand-designed motion planners [36]. Our work instead maps task descriptions into task decompositions and reward functions, to guide reinforcement learning in simulation, to discover behaviours that achieve the generated tasks. Work of [37] also uses language for predicting reward functions for robot locomotion, but does not consider task generation and decomposition or interaction with objects. Our work is the first to use LLMs for task decomposition and reward generation, as well as asset generation.

Automating 3D asset creation with generative models The traditional process of creating 3D assets typically involves multiple labor-intensive stages, including geometry modeling, shape baking, UV mapping, material creation, texturing and physics parameter estimation, where different software tools and the expertise of skilled artists are often required. It is thus desirable to automate 3D asset generation to automatically generate high-quality assets that support realistic rendering under arbitrary views and have plausible physical behaviours during force application and contacts. The lack of available 3D data and the abundance of 2D image data have stimulated interest in learning 3D models from 2D image generators [38], [39]. The availability of strong 2D image generative models based on diffusion led to high-quality 3D models from text descriptions [40], [41], [42] or single 2D images using the diffusion model as a 2D prior [43], [44], [45]. In this work, instead of a text-conditioned model, we use a view and relative pose conditioned image generative model, which we found to provide better prior for score distillation. Some methods attempt to use videos of assets and differentiable simulations to estimate their physics parameters and/or adapt the simulation environment, in an attempt to close the simulation to reality gap [46], [47], [48]. Our effort is complementary to these works.

Procedural demonstration generation using symbolic planners Many recent works procedurally generate scenes and demonstration trajectories using planners that have access to privileged information to solve the task, and distill the demonstration solutions into learning-based policies that operate directly from pixel or point-cloud input [49], [50], [51]. Task and motion planners [52], [53], [54], [55] use predefined symbolic rules and known dynamics models, and infer discrete task plans given instruction with lookahead logic search [53], [56], [52], [53], [54], [55]. These methods predominantly rely on manually-specified symbolic transition rules, planning domains, and grounding, which limits their applicability. Indeed, works of [50], [51] demonstrate their results on relatively simple multi-object box stacking tasks. Scene procedural generation in the aforementioned works [50], [51], [57] entails randomizing locations and number of given 3D models under weak supervision from a human that defines the task and the possible location candi-

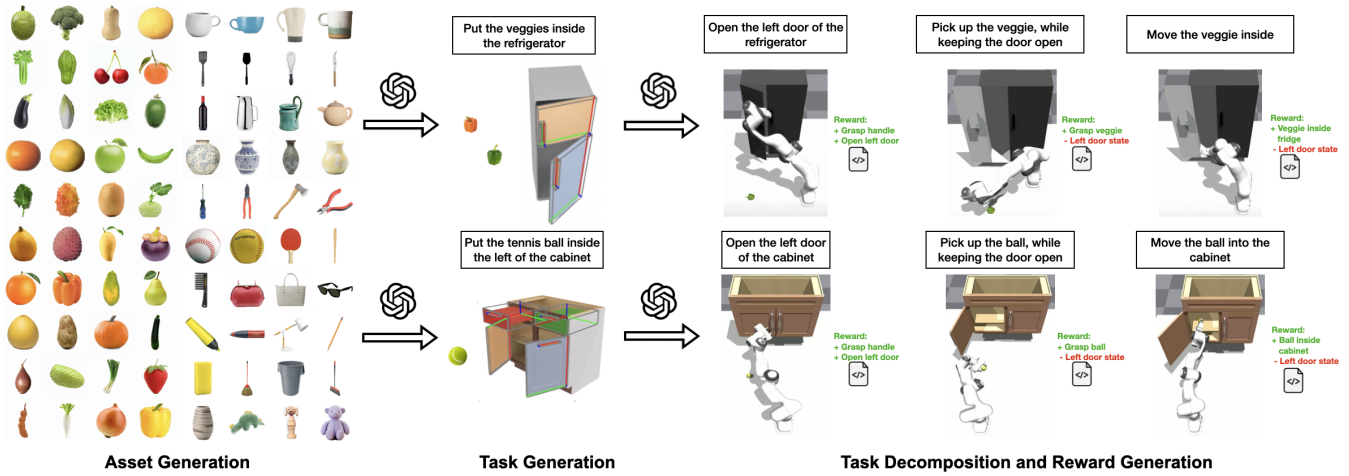

Fig. 2: **The Gen2Sim components.** Gen2Sim generates 3d assets by lifting object-centric 2D images to 3D. It then uses both generated assets and assets obtained from other publicly available datasets to populate scene environments. Afterwards, it queries LLMs to generate meaningful task descriptions for the assembled scenes, as well as decompose the generated task descriptions to sub-tasks and their reward functions.

dates. In contrast, we unleash the common sense knowledge and reasoning capabilities provided by LLMs and use them to suggest task descriptions, task decompositions, and reward functions. We then use reinforcement learning to discover solution trajectories instead of TAMP-based search.

Simulation environments for robotic skill learning In recent years, improving simulators for robot manipulation has attracted increasingly more attention. Many robotic manipulation environments and benchmarks [58], [59], [18] are built on top of either PyBullet [60] or MuJoCo [61] as their underlying physics engines, which mainly support rigid-body manipulation [62], [63], [64], [65], [66]. Recently, environments supporting soft-body manipulation ([67], [18], [68], [69], [70], [71]) provide capabilities for simulating deformable robots, objects and fluids. Our automated asset and task generation are not tied to any specific simulation platforms and can be used with any of them.

III. GEN2SIM

Gen2Sim generates 3D assets from object-centric images using image diffusion models and predicts physical parameters for them using LLMs (Section III-A). It then prompts LLMs to generate language task descriptions and corresponding reward functions for each generated or human-developed asset, suitable to their affordances (Section III-B). Finally, we train RL policies in the generated environments using the generated reward functions. We additionally show the applicability of the simulation-trained policy by constructing digital twin environment in simulation, and deploy the trained trajectory in the real world (Section III-C). See Figure 2 for our method overview.

A. 3D Asset Generation

Gen2Sim automates 3D asset generation by mapping 2D images of objects to textured 3D meshes with plausible physics parameters. The images can be 1) real images taken in the robot's environment, 2) real images provided by Google search under relevant category names, e.g., *"avocado"*, or 3) images generated by pre-trained text-conditioned diffusion models, such as stable diffusion [72], prompted appropriately to generate uncluttered images of the relevant objects, e.g., *"an image of an individual avocado"*. We query GPT-4 [73] for a list of object categories relevant for manipulation tasks to search online for or to generate, instead of manually designing it. Please, visit our project site for a detailed list of the objects we generated. Given a real or generated 2D image of an object, we lift it to a 3D model by minimizing re-reprojection error and maximizing likelihood of its image renderings using a diffusion model [40], [41]. We provide background on image diffusion models below, before we describe our 3D model fitting approach.

*1) **Image diffusion models**:* A diffusion model learns to model a probability distribution $p(x)$ by inverting a process that gradually adds noise to the image x. The diffusion process is associated with a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$, which defines how much noise is added at each time step. The noisy version of sample x at time t can then be written $x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, is a sample from a Gaussian distribution (with the same dimensionality as x), $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. One then learns a denoising neural network $\hat{\epsilon} = \epsilon_\phi(x_t; t)$ that takes as input the noisy image x_t and the noise level t and tries to predict the noise component ϵ. Diffusion models can be easily extended to draw samples from a distribution $p(x|\mathbf{c})$ conditioned on a prompt $\mathbf{c}$, where $\mathbf{c}$ can be a text description, a camera pose, and image semantic map, *etc* [4], [74], [75]. Conditioning on the prompt can be done by adding $\mathbf{c}$ as an additional input of the network ϵ_ϕ. For 3D lifting, we build on Zero-1-to-3 [76], a diffusion model for novel object view synthesis that conditions on an image view of an object and a relative camera rotation around the object to generate

plausible images for the target object viewpoint, $\mathbf{c} = [I_1, \pi]$. It is trained on a large collection $\mathcal{D}' = \{(x^i, \mathbf{c}^i)\}_{i=1}^N$ of images paired with views and relative camera orientations as conditioning prompt by minimizing the loss:

$$\mathcal{L}_{\text{diff}}(\phi; \mathcal{D}') = \frac{1}{|\mathcal{D}'|} \sum_{x^i, \mathbf{c}^i \in \mathcal{D}'} ||\epsilon_\phi(\sqrt{\bar{\alpha}_t} x^i + \sqrt{1-\bar{\alpha}_t}\epsilon, \mathbf{c}^i, t) - \epsilon||^2.$$

*2) **Image-to-3D Mesh using Score Distillation Sampling**:* Given an image and relative camera pose 2D diffusion model $p(I|[I_0, \pi])$, we extract from it a 3D rendition of the input image I_0, represented by a differential 3D representation using Score Distillation Sampling (SDS) [40], [77]. We do so by randomly sampling a camera pose π, rendering a corresponding view I_π, assessing the likelihood of the view based on a diffusion model $p(I_\pi|[I_0, \pi])$, and updating the differentiable 3D representation to increase the likelihood of the generated view based on the model. Specifically, the diffusion model is frozen and the 3D model is updated as:

$$\nabla(\theta)\mathcal{L}_{SDS}(\theta; \pi, \mathbf{c}, t) = \mathbb{E}_{t,\epsilon}[w(t)(\epsilon_\phi(a_t I + \sigma_t \epsilon; t, \mathbf{c}) - \epsilon) \cdot \nabla_\theta I],$$

where $I = R(\theta, \pi)$ is the image rendered from a given viewpoint π. The loss we use to backpropagate to the 3D model parameters θ includes an image re-projection loss for the camera viewpoint of the input image, and score distillation for the other views, using a pre-trained view and pose conditioned image diffusion model of [76] to measure 2D image likelihood. We use a two-stage fitting, wherein the first stage an instantNGP NeRF representation [78] is used, similar to RealFusion [43], and in the second stage a mesh-based representation is initialized from the NeRF and finetuned differentiably, similar to Fantasia3D [41]. More information of our score distillation sampling can be found in our website.

*3) **Texture generation**:* We augment the textures of our generated assets using the method of TEXTure [79] which iteratively edits a mesh's texture by rendering the mesh from different viewpoints and updating the rendered 2D images. While domain randomization [80] randomly re-textures simulated assets, TEXTure produces diverse yet plausible texture augmentations.

*4) **Generating plausible physical properties**:* The visual and collision parameters of an asset are generated from the Image-to-Mesh pipeline discussed above. To define 3D sizes and physics parameters for the generated 3D meshes, we query GPT-4 regarding the range of plausible width, height, and depth for each object, and the range of its mass given its category. We then scale the generated 3D mesh based on the generated size range. We feed the mass and 3D mesh information to MeshLab [81] to get the inertia matrix for the asset. Our prompts for querying GPT for mass and 3D object size can be found on our website. We wrap the generated mesh information, its semantic name, as well as the physical parameters into URDF files to be loaded into our simulator.

B. Task Generation, Temporal Decomposition and Reward Function Prediction

Given either generated assets or assets obtained from publically available datasets, we prompt LLMs to generate meaningful manipulation tasks considering their affordances, to decompose these tasks into subtasks when possible, and to generate reward functions for each subtask. We train reinforcement learning policies for each (sub)task using the generated reward functions, and then chain them together to solve long horizon tasks. Our LLM prompts contain the following sections:

1. Asset descriptions. We use combinations of assets we generate using the method of Section III-A, as well as articulated assets from PartNet Mobility [18] and GAPartNet dataset [82]. We populate our simulation environment with randomly sampled assets. Then, we extract information from the URDF files including link names, joint types and limits using automated scripts. For example, an asset `microwave` has parts [`door`, `handle`, and `body`], and joint [`door-joint`] of type `revolute` with a joint position range $[0, 1]$. We then describe the extracted configurations of the assets to the LLM, as shown below:

```
The environment contains the following assets:
1.  asset_name: "microwave"
    part_cofiguration:
        Part 1: "body"
        Part 2: "door"
            - link_name: "link_0"
            - joint_name: "joint_0"
            - joint_type: "revolute"
            - limit: [0, 1]
        Part 3: "handle"
            - link_name: "handle_0"
            - joint_name: "handlejoint_0"
            - joint_type: "fixed"
2.  asset_name: "cup"
    part_cofiguration:
        Part 1: cup
            - link_name: "base"
            - joint_name: "base_joint"
            - joint_type: "fixed"
```

2. Instructions. These include function APIs that can be used by the LLM to query the pose of the robot end-effector, as well as different assets in the given environment:

```
Available APIs from the simulator are:
# returns the pose of the link
get_pose_by_link_name(asset_name, link_name)
# returns the pose of the robot gripper
get_robot_gripper_pose(asset_name, link_name)
# returns the state of the joint
get_state_by_joint_name(asset_name, joint_name)
# returns the limit of the joint
get_limits_by_joint_name(asset_name, joint_name)

Note:
1. Only use the available APIs from the simulator.
2. Generate the reward function code snippets in Python.
```

3. Examples of task descriptions and decompositions. These are question-answer pairs that demonstrate task descriptions and their temporal decompositions.

```
List meaningful manipulation tasks that can be performed
in this environment. Give subtask decomposition and the
order of execution to solve the task. Also, provide the
reward function for each subtask.

The following tasks can be performed in this environment:
1. Open the Microwave Door
```

```
2. Close the Microwave Door
3. Pick Cup
4. Place Cup
5. Put the Cup in the Microwave
   This task needs to be decomposed into sub-tasks:
    - Open the Microwave
    - Pick Cup
    - Place the Cup in the Microwave
```

4. Examples of reward functions. These are task to reward function pairs that present demonstrations of how tasks can be translated to reward functions, as shown below:

```
Task: OpenMicrowaveDoor
Task Description: open the door of the microwave
'''
def compute_reward(env):
    # reward function
    door_handle_pose = env.get_pose_by_link_name("
     microwave", "handle_0")
    gripper_pose = env.get_robot_gripper_pose()
    distance_gripper_to_handle = torch.norm(
     door_handle_pose - gripper_pose, dim=-1)
    door_state = env.get_state_by_joint_name("microwave",
     "joint_0")
    cost = distance_gripper_to_handle - door_state
    reward = - cost

    # success condition
    target_door_state = env.get_limits_by_joint_name("
     microwave", "joint_0")["upper"]
    success = torch.abs(door_state - target_door_state) <
     0.1

    return reward, success
'''
```

For the example above, the reward function is comprised of 1) distance between the end-effector and the target part, and 2) distance between the current and the target pose of an articulated asset, link, or joint.

We provide our prompts on our website. We show in Section IV that our method can generalize across assets, suggest diverse and plausible tasks, decomposition and reward functions automatically, using a single in-context example in the prompt, without any additional human involvement.

C. Sequential Reinforcement Learning for Long Horizon Tasks

We train policies using Proximal Policy Optimization (PPO) [83] maximizing the generated reward functions for each subtask. We train RL for each generated subtask in temporal order. Once policy training for a subtask converges, we proceed to the next subtask, by sampling the initial state of the end-effector and the environment close to the terminal states of the previous subtask. This ensures policies can be temporally chained upon training. Our policies are trained per environment using privileged information of the simulation state to accelerate exploration. Such learned policies can be used as demonstration data and distilled into vision-language transformer policies, similar to [8], [84], [85]; we leave this for future work.

IV. Experiments

Our experiments aim to answer the following questions:

1. Can Gen2Sim generate plausible geometry, appearance, and physics for diverse types of objects and parts, without human expertise and with minimal human involvement?

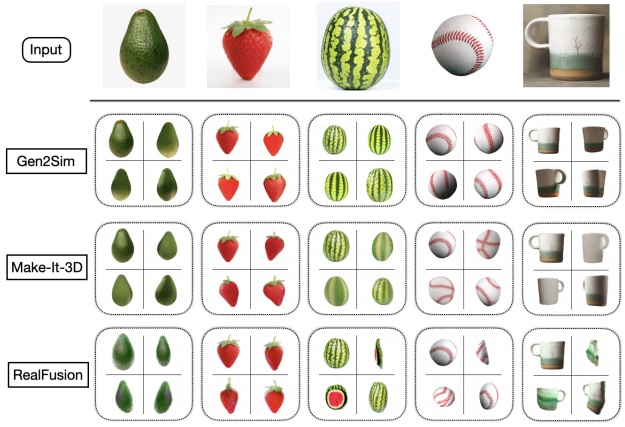

Fig. 3: **3D asset generation** from Gen2Sim, RealFusion [43] and Make-It-3D [44]. Gen2Sim uses a view and camera pose conditioned image generative model during score distillation, which helps generate more accurate 3D geometry in comparison to the baselines.

	Mass (gram)	Length (cm)	Width	Height
Papaya	500-1000	15-20	10-15	10-15
Cucumber	200-300	15-20	5-7	5-7
Watermelon	5000-7000	30-40	20-30	20-30
Raspberry	3-5	2-3	2-3	2-3
Coconut	600-800	10-15	8-12	8-12
Corn	50-100	10-15	8-12	8-12
Pumpkin	2000-5000	20-40	20-40	20-40
Avocado	150-250	10-12	6-8	4-5

TABLE I: **Size and physics parameter generated by LLMs** for a number of generated assets.

2. Can Gen2Sim generate task language goals and reward functions for novel object categories, novel assets with different part configurations, and a combination of multiple assets in an environment?

3. Can the generated environments and reward function lead to successful learning of RL policies?

A. Asset Generation

We compare our image-to-3D lifting with two baselines:

1. *RealFusion* [43], which uses textual inversion of [86] to learn a word embedding for the depicted object concept in an image, and uses text-conditioned diffusion with this text embedding during score distillation.

2. *Make-It-3D* [44], which uses the same NeRF and textured mesh two-stage fitting as Gen2Sim, but does not use a view and pose conditioned generative model, rather a text-based image diffusion model, similar to [40].

We show comparisons in Figure 3, with images rendered from 4 different views. Our model generates more plausible 3D model as our image diffusion prior comes from an image and pose-conditioned model in comparison to approaches like Fantasia3D or RealFusion which uses text conditioning. We show generated values for 3D sizes and mass for a number of example objects in Table I. We see that the common sense knowledge encoded in LLMs can produce reasonable physical parameters.

B. Automated Skill Learning

Gen2Sim generates diverse task descriptions, task decompositions and reward functions automatically for hundreds of assets, with different category labels and number of joints, **given only a single in-context prompt example** regarding the task decomposition and reward function of the task "putting a cup in a Microwave" . Then, the model can generalize to different scenes, asset articulated structures and task temporal lengths. We show some examples of such generated task descriptions in Figure 1 and more on our website. We show examples of task decompositions in Figure 2. We provide our prompts in our project website, alongside examples of the LLM's responses.

We learn policies that optimize LLM generated rewards with PPO, an off-the-shelf model-free RL algorithm [83]. We make use of GPU-parallel data sampling in IsaacGym [87] for reinforcement learning. Our robotic setup uses a Franka Panda arm with a mobile base. It is equipped with a parallel-jaw gripper. Our state representation for PPO includes the robot's joint position $q \in \mathbb{R}^{11}$, velocity $\dot{q} \in \mathbb{R}^{11}$ (7-DoF arm, x and y for the mobile base and 2 extra DoFs from the gripper jaws), orientation of the gripper $r \in SO(3)$, and poses and joint configurations of the assets present in the scene. We use position control and at each timestep t our policy produces target gripper pose and configurations which is converted to target robot configurations through inverse kinematics. A low-level PID torque controller provided by IsaacGym is used to produce low-level joint torque commands. We can successfully learn useful manipulation policies, and the polices are able to solve the tasks upon convergence. We show videos of such policies on our website.

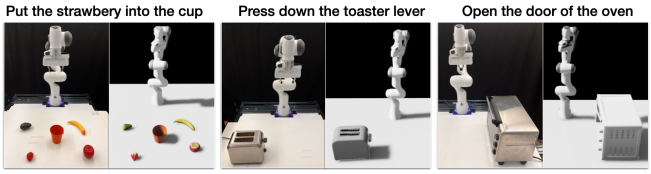

Fig. 4: **Twin environments** constructed and generated tasks for sim-to-real transfer. Left: real-world. Right: simulated.

C. Twin environment construction and sim-to-real world transfer

In order to validate the usefulness of the policies trained in simulation, we construct a twin simulated environment of our lab's real-robot setup (Figure 4). We detect, segment, and estimate the poses of the objects in the scene. For non-articulated assets, we use our model to lift the detected object image to corresponding 3D models; for articulated objects, we select the most similar asset from the [18], and populated the simulated environment. We train RL policies in simulation and transfer the joint space trajectory back to our real-world setup. Our method allows successful execution of the generated tasks. For more videos of the trained policies and their task executions in simulation, as well as the sim2real transfer, please refer to our website.

D. Limitations

Gen2Sim has currently the following two important points to address towards materializing into a platform for large-scale robot skill learning that are deployable in real-world:

1. Sim2real transfer of closed-loop policies: Our current real-world experiments transfer open loop trajectories optimized in the constructed twin environment. For closed-loop policies to transfer to the real world and consume realistic sensory input, we would need to generate large-scale augmentations for both visual appearances and dynamics for each task and sub-task, and then distil the state-based RL policies to a foundational vision-language policy network. This is a direct avenue for our future work.

2. Beyond rigid asset generation: The assets we can currently generate are rigid or mostly rigid objects, which do not deform significantly under external forces. For articulated assets, we are using existing manually designed and labelled datasets ([18], [82]). To generate articulated objects, deformable objects and liquids, accurate fine-grained video perception is required in combination with generative priors to model the temporal dynamics of their geometry and appearance. This is an exciting and challenging direction for future work.

V. Conclusion

We have presented Gen2Sim, a method for automating the development of simulation environments, tasks and reward functions with pre-trained generative models of vision and language. We presented methods that create and augment geometry, textures and physics of object assets from single images, parse URDF files of assets, generate task descriptions, decompositions and reward python functions, and train reinforcement learning policies to solve the generated long horizon tasks. Addressing the limitations including generating diverse assets with more complex physical properties, and transfering trained policies to real world are direct avenues for our future work. We believe generative models of images and language will play an important role in automating and supersizing robot training data in simulation, and in crossing the sim2real gap, necessary for delivering robot generalists in the real world. Gen2Sim takes one first step in that direction.

Acknowledgment

This work is supported by Sony AI, NSF award No 1849287, DARPA Machine Common Sense, an Amazon faculty award, and an NSF CAREER award.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[5] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," 2016.

[6] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-structured visual imitation," in *Conference on Robot Learning*. PMLR, 2020, pp. 979–989.

[7] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[9] S. Höfer, K. E. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. G. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. K. Liu, J. Peters, S. Song, P. Welinder, and M. White, "Sim2real in robotics and automation: Applications and challenges," *IEEE Trans Autom. Sci. Eng.*, vol. 18, no. 2, pp. 398–400, 2021. [Online]. Available: https://doi.org/10.1109/TASE.2021.3064065

[10] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak, "Coupling vision and proprioception for navigation of legged robots," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 273–17 283.

[11] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *CoRR*, vol. abs/2010.11251, 2020. [Online]. Available: https://arxiv.org/abs/2010.11251

[13] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," 2023.

[14] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*. PMLR, 2022, pp. 297–307.

[15] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," 2019.

[16] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Mueller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, 08 2023.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," 2017.

[18] 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," 2020.

[19] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," 2021.

[20] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," 2019.

[21] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. D. Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, M. Lingelbach, A. Curtis, K. Feigelis, D. M. Bear, D. Gutfreund, D. Cox, A. Torralba, J. J. DiCarlo, J. B. Tenenbaum, J. H. McDermott, and D. L. K. Yamins, "Threedworld: A platform for interactive multi-modal physical simulation," 2021.

[22] Z. Xian, T. Gervet, Z. Xu, Y.-L. Qiao, and T.-H. Wang, "Towards a foundation model for generalist robots: Diverse skill learning at scale via automated task and scene generation," *arXiv preprint arXiv:2305.10455*, 2023.

[23] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," *CoRR*, vol. abs/2012.07277, 2020. [Online]. Available: https://arxiv.org/abs/2012.07277

[24] D. Xu, R. Martín-Martín, D. Huang, Y. Zhu, S. Savarese, and L. Fei-Fei, "Regression planning networks," *CoRR*, vol. abs/1909.13072, 2019. [Online]. Available: http://arxiv.org/abs/1909.13072

[25] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *arXiv preprint arXiv:2201.07207*, 2022.

[26] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," 2022. [Online]. Available: https://arxiv.org/abs/2207.05608

[27] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," 2022. [Online]. Available: https://arxiv.org/abs/2209.07753

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *CoRR*, vol. abs/2201.11903, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903

[29] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *CoRR*, vol. abs/2107.13586, 2021. [Online]. Available: https://arxiv.org/abs/2107.13586

[30] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[31] S. C. Y. Chan, A. Santoro, A. K. Lampinen, J. X. Wang, A. Singh, P. H. Richemond, J. McClelland, and F. Hill, "Data distributional properties drive emergent in-context learning in transformers," 2022.

[32] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. Dai, "Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory," 2023.

[33] S. Lifshitz, K. Paster, H. Chan, J. Ba, and S. McIlraith, "Steve-1: A generative model for text-to-behavior in minecraft," 2023.

[34] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," 2023.

[35] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," *arXiv preprint arXiv:2307.14535*, 2023.

[36] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[37] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," *arXiv preprint arXiv:2306.08647*, 2023.

[38] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "Hologan: Unsupervised learning of 3d representations from natural images," 2019.

[39] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[40] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv*, 2022.

[41] R. Chen, Y. Chen, N. Jiao, and K. Jia, "Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation," *arXiv preprint arXiv:2303.13873*, 2023.

[42] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-

to-3d content creation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[43] L. Melas-Kyriazi, C. Rupprecht, I. Laina, and A. Vedaldi, "Realfusion: 360 reconstruction of any object from a single image," in *CVPR*, 2023. [Online]. Available: https://arxiv.org/abs/2302.10663

[44] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen, "Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior," 2023.

[45] B. Shen, X. Yan, C. R. Qi, M. Najibi, B. Deng, L. Guibas, Y. Zhou, and D. Anguelov, "Gina-3d: Learning to generate implicit neural assets in the wild," 2023.

[46] E. Heiden, C. E. Denniston, D. Millard, F. Ramos, and G. S. Sukhatme, "Probabilistic inference of simulation parameters via parallel differentiable simulation," 2022.

[47] E. Heiden, M. Macklin, Y. Narang, D. Fox, A. Garg, and F. Ramos, "Disect: A differentiable simulator for parameter inference and control in robotic cutting," 2022.

[48] K. Wang, W. R. J. I. au2, S. Lu, X. Huang, J. Booth, R. Kramer-Bottiglio, M. Aanjaneya, and K. Bekris, "Real2sim2real transfer for control of cable-driven robots via a differentiable physics engine," 2023.

[49] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," 2022.

[50] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," 2023.

[51] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," 2021.

[52] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *CoRR*, vol. abs/1911.04679, 2019. [Online]. Available: http://arxiv.org/abs/1911.04679

[53] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[54] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 1930–1936.

[55] D. Lyu, F. Yang, B. Liu, and S. Gustafson, "SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning," *CoRR*, vol. abs/1811.00090, 2018. [Online]. Available: http://arxiv.org/abs/1811.00090

[56] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Stripstream: Integrating symbolic planners and blackbox samplers," *CoRR*, vol. abs/1802.08705, 2018. [Online]. Available: http://arxiv.org/abs/1802.08705

[57] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox, "Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation," 2023.

[58] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[59] 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 preprint arXiv:1712.05474*, 2017.

[60] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016.

[61] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[62] H.-Y. F. Tung, Z. Xian, M. Prabhudesai, S. Lal, and K. Fragkiadaki, "3d-oes: Viewpoint-invariant object-factorized environment simulators," *arXiv preprint arXiv:2011.06464*, 2020.

[63] N. Gkanatsios, A. Jain, Z. Xian, Y. Zhang, C. Atkeson, and K. Fragkiadaki, "Energy-based models as zero-shot planners for compositional scene rearrangement," *arXiv preprint arXiv:2304.14391*, 2023.

[64] Z. Xian, S. Lal, H.-Y. Tung, E. A. Platanios, and K. Fragkiadaki, "Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks," *arXiv preprint arXiv:2103.09439*, 2021.

[65] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: Infinite resolution action detection transformer for robotic manipulation," *arXiv preprint arXiv:2306.17817*, 2023.

[66] Z. Xian, N. Gkanatsios, T. Gervet, T.-w. Ke, and K. Fragkiadaki, "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation," *Conference on Robot Learning*, 2023.

[67] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[68] C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano *et al.*, "Threedworld: A platform for interactive multi-modal physical simulation," *arXiv preprint arXiv:2007.04954*, 2020.

[69] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," *arXiv preprint arXiv:2011.07215*, 2020.

[70] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan, "Fluidlab: A differentiable environment for benchmarking complex fluid manipulation," in *International Conference on Learning Representations*, 2023.

[71] T.-H. Wang, A. E. Spielberg, P. Ma, Z. Xian, H. Zhang, J. B. Tenenbaum, and C. Gan, "Softzoo: A soft robot co-design benchmark for locomotion in diverse environments," in *International Conference on Learning Representations*, 2023.

[72] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[73] OpenAI, "Gpt-4 technical report," 2023.

[74] Y. Li, H. Liu, Q. Wu, F. Mu, J. Yang, J. Gao, C. Li, and Y. J. Lee, "Gligen: Open-set grounded text-to-image generation," *arXiv preprint arXiv:2301.07093*, 2023.

[75] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *arXiv preprint arXiv:2302.05543*, 2023.

[76] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," 2023.

[77] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich, "Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation," 2022.

[78] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, jul 2022. [Online]. Available: https://doi.org/10.1145%2F3528223.3530127

[79] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or, "Texture: Text-guided texturing of 3d shapes," 2023.

[80] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," 2017.

[81] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008.

[82] H. Geng, H. Xu, C. Zhao, C. Xu, L. Yi, S. Huang, and H. Wang, "Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts," 2023.

[83] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[84] S. Christen, W. Yang, C. Pérez-D'Arpino, O. Hilliges, D. Fox, and Y.-W. Chao, "Learning human-to-robot handovers from point clouds," 2023.

[85] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[86] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," 2022.

[87] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.