Habitat 2.0: Training Home Assistants to Rearrange their Habitat

Andrew Szot[2], [*]**Alex Clegg**[1], **Eric Undersander**[1], **Erik Wijmans**[1,2], **Yili Zhao**[1], **John Turner**[1], **Noah Maestre**[1], **Mustafa Mukadam**[1], **Devendra Chaplot**[1], **Oleksandr Maksymets**[1], **Aaron Gokaslan**[1], **Vladimir Vondrus, Sameer Dharur**[2], **Franziska Meier**[1], **Wojciech Galuba**[1], **Angel Chang**[4], **Zsolt Kira**[2], **Vladlen Koltun**[3], **Jitendra Malik**[1,5], **Manolis Savva**[4], **Dhruv Batra**[1,2]
[1]Facebook AI Research, [2]Georgia Tech, [3]Intel Research, [4]Simon Fraser University [5]UC Berkeley

Abstract

We introduce Habitat 2.0 (H2.0), a simulation platform for training virtual robots in *interactive* 3D environments and complex physics-enabled scenarios. We make comprehensive contributions to all levels of the embodied AI stack – data, simulation, and benchmark tasks. Specifically, we present: (i) ReplicaCAD: an artist-authored, annotated, reconfigurable 3D dataset of apartments (matching real spaces) with articulated objects (*e.g.* cabinets and drawers that can open/close); (ii) H2.0: a high-performance physics-enabled 3D simulator with **speeds exceeding 25,000 simulation steps per second (850$\times$ real-time)** on an 8-GPU node, representing 100$\times$ speed-ups over prior work; and, (iii) Home Assistant Benchmark (HAB): a suite of common tasks for assistive robots (tidy the house, stock groceries, set the table) that test a range of mobile manipulation capabilities. These large-scale engineering contributions allow us to systematically compare deep reinforcement learning (RL) at scale and classical sense-plan-act (SPA) pipelines in long-horizon structured tasks, with an emphasis on generalization to new objects, receptacles, and layouts. We find that (1) flat RL policies struggle on HAB compared to hierarchical ones; (2) a hierarchy with independent skills suffers from 'hand-off problems', and (3) SPA pipelines are more brittle than RL policies.

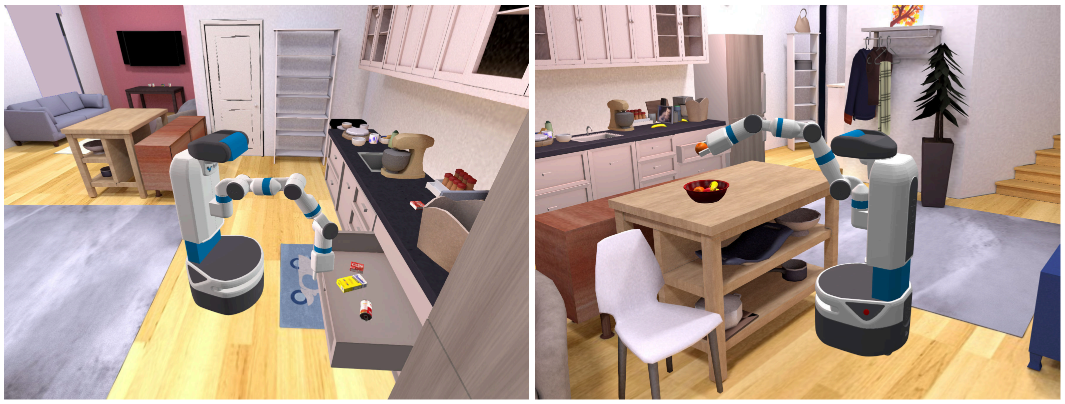


Figure 1: A mobile manipulator (Fetch robot) simulated in Habitat 2.0 performing rearrangement tasks in a ReplicaCAD apartment – (left) opening a drawer before picking up an item from it, and (right) placing an object into the bowl after navigating to the table. Best viewed in motion at https://aihabitat.org/docs/habitat2.

1 Introduction

Consider a home assistant robot illustrated in Fig. 1 – a mobile manipulator (Fetch [1]) performing tasks like stocking groceries into the fridge, clearing the table and putting dishes into the dishwasher,

[*]Correspondence to: aszot3@gatech.edu

fetching objects on command and putting them back, *etc*. Developing such embodied intelligent systems is a goal of deep scientific and societal value. So how should we accomplish this goal?

Training and testing such robots in hardware directly is slow, expensive, and difficult to reproduce. We aim to advance the entire 'research stack' for developing such embodied agents in simulation – (1) data: curating house-scale interactive 3D assets (*e.g*. kitchens with cabinets, drawers, fridges that can open/close) that support studying generalization to unseen objects, receptacles, and home layouts, (2) simulation: developing the next generation of high-performance photo-realistic 3D simulators that support rich interactive environments, (3) tasks: setting up challenging representative benchmarks to enable reproducible comparisons and systematic tracking of progress over the years. To support this long-term research agenda, we present:

- **ReplicaCAD**: an artist-authored fully-interactive recreation of 'FRL-apartment' spaces from the Replica dataset [2] consisting of 111 unique layouts of a single apartment background with 92 authored objects including dynamic parameters, semantic class and surface annotations, and efficient collision proxies, representing 900+ person-hours of professional 3D artist effort. ReplicaCAD (illustrated in figures and videos) was created with the consent of and compensation to artists, and will be shared under a Creative Commons license for non-commercial use with attribution (CC-BY-NC).

- **Habitat 2.0 (H2.0)**: a high-performance physics-enabled 3D simulator, representing approximately 2 years of development effort and the next generation of the Habitat project [3] (Habitat 1. 0). H2.0 supports piecewise-rigid objects (*e.g*. door, cabinets, and drawers that can rotate about an axis or slide), articulated robots (*e.g*. mobile manipulators like Fetch [1], fixed-base arms like Franka [4], quadrupeds like AlienGo [5]), and rigid-body mechanics (kinematics and dynamics). The design philosophy of H2.0 is to prioritize performance (or speed) over the breadth of simulation capabilities. H2.0 by design and choice does not support non-rigid dynamics (deformables, fluids, films, cloths, ropes), physical state transformations (cutting, drilling, welding, melting), audio or tactile sensing – many of which are capabilities provided by other simulators [6–8]. The benefit of this focus is that we were able to design and optimize H2.0 to be *exceedingly* fast – simulating a Fetch robot interacting in ReplicaCAD scenes at 1200 steps per second (SPS), where each 'step' involves rendering 1 RGBD observation (128×128 pixels) and simulating rigid-body dynamics for $1/30$ sec. Thus, 30 SPS would be considered 'real time' and 1200 SPS is 40× real-time. H2.0 also scales well – achieving 8,200 SPS (273× real-time) multi-process on a single GPU and over 25,000 SPS (850× real-time) on a single node with 8 GPUs. For reference, existing simulators typically achieve 10-400 SPS (see Tab. 1). These 100× simulation-speedups correspond to cutting experimentation time from 6 months to under 2 days, unlocking experiments that were hitherto infeasible, allowing us to answer questions that were hitherto unanswerable. As we will show, they also directly translate to training-time speed-up and accuracy improvements from training agents (for object rearrangement tasks) on more experience.

- **Home Assistant Benchmark (HAB):** a suite of common tasks for assistive robots (`TidyHouse`, `PrepareGroceries`, `SetTable`) that are specific instantiations of the generalized rearrangement problem [9]. Specifically, a mobile manipulator (Fetch) is asked to rearrange a list of objects from initial to desired positions – picking/placing objects from receptacles (counter, sink, sofa, table), opening/closing containers (drawers, fridges) as necessary. We use the GeometricGoal specification prescribed by Batra *et al*. [9] – *i.e*., initial and desired 3D (center-of-mass) position of each target object i to be rearranged $\left(s_i^0, s_i^*\right)_{i=1}^N$. The choice of GeometricGoal is deliberate – we aim to create the PointNav [10] equivalent for mobile manipulators. As witnessed in the navigation literature, such a task becomes the testbed for exploring ideas [11–19] and a starting point for more semantic tasks [20–22]. The robot operates entirely from onboard sensing – head- and arm-mounted RGB-D cameras, proprioceptive joint-position sensors (for the arm), and egomotion sensors (for the mobile base) – and may not access any privileged state information (no prebuilt maps, no 3D models of rooms or objects, no physically-implausible sensors providing knowledge of mass, friction, articulation of containers, *etc*.). Notice that an object's center-of-mass provides no information about its size or orientation. The target object may be located inside a container (drawer, fridge), on top of supporting surfaces (shelf, table, sofa) of varying heights and sizes, and surrounded by clutter; all of which must be sensed and maneuvered. Receptacles like drawers and fridges start closed, meaning that the agent must open and close articulated objects to succeed. An episode is considered successful if all target objects are placed within 15cm of their desired positions (without considering orientation). The robot uses continuous end-effector control for the arm and velocity control for the base. We deliberately focus on gross motor control (the base and arm) and not fine motor control (the gripper), following

	Rendering		Physics		Scene	Speed
	Library	Supports	Library	Supports	Complexity	(steps/sec)
Habitat [3]	Magnum	3D scans	none	continuous navigation (navmesh)	building-scale	3,000
AI2-THOR [6]	Unity	Unity	Unity	rigid dynamics, animated interactions	room-scale	30 - 60
ManipulaTHOR [33]	Unity	Unity	Unity	AI2-THOR + manipulation	room-scale	30 - 40
ThreeDWorld [7]	Unity	Unity	Unity (PhysX) + FLEX	rigid + particle dynamics	room/house-scale	5 - 168
SAPIEN [34]	OpenGL/OptiX	configurable	PhysX	rigid/articulated dynamics	object-level	200 - 400†
RLBench [35]	CoppeliaSim (OpenGL)	Gouraud shading	CoppeliaSim (Bullet/ODE)	rigid/articulated dynamics	table-top	1 - 60†
iGibson [36]	PyRender	PBR shading	PyBullet	rigid/articulated dynamics	house-scale	100
Habitat 2.0 (H2.0)	Magnum	3D scans + PBR shading	Bullet	rigid/articulated dynamics + navmesh	house-scale	1,200

Table 1: High-level comparison of different simulators. Note: Speeds were taken directly from respective publications or obtained via direct personal correspondence with the authors when not publicly available (indicated by †). Benchmarking was conducted by different teams on different hardware with different underlying 3D assets simulating different capabilities. Thus, these should be considered qualitative comparisons representing what a user expects to experience on a single instance of the simulator (no parallelization).

the 'abstracted grasping' recommendations from [9]. Specifically, once the end-effector reaches 15cm (or closer) to an object, a discrete grasp action becomes available that, if executed, snaps the object into its parallel-jaw gripper [2]. We conduct a systematic study of two distinct techniques – monolithic 'sensors-to-actions' policies trained with reinforcement learning (RL) at scale, and classical sense-plan-act pipelines (SPA) [26] – with a particular emphasis on systematic generalization to new objects, receptacles, apartment layouts (not just robot starting pose). Our findings include:

1. **Flat vs hierarchical:** Monolithic RL policies successfully learn diverse *individual* skills (pick/place, navigate, open/close drawer). However, crafting a combined reward function and learning scheme that elicits chaining of such skills for the long-horizon HAB tasks remained out of our reach. We saw significantly stronger results with a hierarchical approach that assumes knowledge of a perfect task planner (via STRIPS [27]) to break it down into a sequence of skills.
2. **Hierarchy cuts both ways:** However, a hierarchy with independent skills suffers from 'hand-off problems' where a succeeding skill isn't set up for success by the preceding one – *e.g.*, navigating to a bad location for subsequent manipulation, only partially opening a drawer to grab an object inside, or knocking an object out of reach that is later needed.
3. **Brittleness of SensePlanAct:** For simple skills, SPA performs just as well as monolithic RL. However, it is significantly more brittle since it needs to map all obstacles in the workspace for planning. More complex settings involving clutter, challenging receptacles, and imperfect navigation can poorly frame the target object and obstacles in the robot's camera, leading to incorrect plans.

We hope our work will serve as a benchmark for many years to come. H2.0 is free, open-sourced under the MIT license, and under active development. [3] We believe it will reduce the community's reliance on commercial lock-ins [28, 29] and non-photorealistic simulation engines [30–32].

2 Related Work

What *is* a simulator? Abstractly speaking, a simulator has two components: (1) a *physics engine* that evolves the world state s over time $s_t \to s_{t+1}$, and (2) a *renderer* that generates sensor observations o from states: $s_t \to o_t$. The boundary between the two is often blurred as a matter of convenience. Many physics engines implement minimal renderers to visualize results, and some rendering engines include integrations with a physics engine. PyBullet [37], MuJoCo [28], DART [38], ODE [39], PhysX/FleX [40, 41], and Chrono [42] are primarily physics engines with some level of rendering, while Magnum [43], ORRB [44], and PyRender [45] are primarily renderers. Game engines like Unity [46] and Unreal [47] provide tightly coupled integration of physics and rendering. Some simulators [3, 48, 49] involve largely static environments – the agent can move but not change the state of the environment (*e.g.* open cabinets). Thus, they are heavily invested in rendering with fairly lightweight physics (*e.g.* collision checking with the agent approximated as a cylinder).

How are interactive simulators built today? Either by relying on game engines [6, 50, 51] or via a 'homebrew' integration of existing rendering and physics libraries [7, 34, 36, 52]. Both options have problems. Game engines tend to be optimized for human needs (high image-resolution, ∼60 FPS, persistent display) not for AI's needs [53] (10k+ FPS, low-res, 'headless' deployment on a

[2]To be clear, H2.0 fully supports the rigid-body mechanics of grasping; the abstract grasping is a *task-level* simplification that can be trivially undone. Grasping, in-hand manipulation, and goal-directed releasing of a grasp are all challenging open research problems [23–25] that we believe must further mature in the fixed-based close-range setting before being integrated into a long-horizon home-scale rearrangement problem.

[3]Start using Habitat 2.0 with the tutorial at https://aihabitat.org/docs/habitat2

cluster). Reliance on them leads to limited control over the performance characteristics. On the other hand, they represent decades of knowledge and engineering effort whose value cannot be discounted. This is perhaps why 'homebrew' efforts involve a high-level (typically Python-based) integration of existing libraries. Unfortunately but understandably, this results in simulation speeds of 10-100s of SPS, which is *orders of magnitude* sub-optimal. H2.0 involved a deep low-level (C++) integration of rendering (via Magnum [43]) and physics (via Bullet [37]), enabling precise control of scheduling and task-aware optimizations, resulting in substantial performance improvements.

Object rearrangement. Task- and motion-planning [54] and mobile manipulation have a long history in AI and robotics, whose full survey is beyond the scope of this document. Batra et al. [9] provide a good summary of the historical background of rearrangement, a review of recent efforts, a general framework, and a set of recommendations that we adopt here. Broadly speaking, our work is distinguished from prior literature by a combination of the emphasis on visual perception, lack of access to state, systematic generalization, and the experimental setup of visually-complex and ecologically-realistic home-scale environments. We now situate w.r.t. a few recent efforts. [55] study replanning in the presence of partial observability but do not consider mobile manipulation. [52] tackle 'interactive navigation', where the robot can bump into and push objects during navigation, but does not have an arm. Some works [56–58] abstract away gross motor control entirely by using symbolic interaction capabilities (*e.g*. a 'pick up X' action) or a 'magic pointer' [9]. We use abstracted grasping but not abstract manipulation. [19] develop hierarchical methods for mobile manipulation, combining RL policies for goal-generation and motion-planning for executing them. We use the opposite combination of planning and learning – using task-planning to generate goals and RL for skills. [33] is perhaps the most similar to our work. Their task involves moving a single object from one location to another, excluding interactions with container objects (opening a drawer or fridge to place an object inside). We will see that rearrangement of multiple objects while handling containment is a much more challenging task. Interestingly, our experiments show evidence for the opposite conclusion reached therein – monolithic end-to-end trained RL methods are outperformed by a modular approach that is trained stage-wise to handle long-horizon rearrangement tasks.

3 Replica to ReplicaCAD: Creating Interactive Digital Twins of Real Spaces

We begin by describing our dataset that provides a rich set of indoor layouts for studying rearrangement tasks. Our starting point was Replica [2], a dataset of *highly* photo-realistic 3D reconstructions at room and building scale. Unfortunately, static 3D scans are unsuitable for studying rearrangement tasks because objects in a static scan cannot be moved or manipulated.

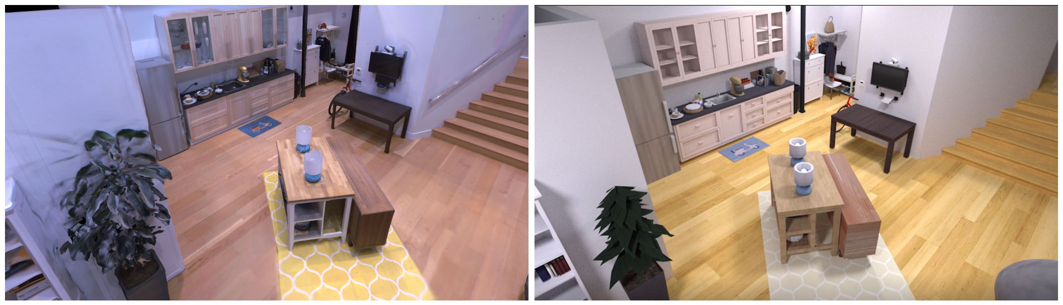


Figure 2: Left: The original Replica scene. Right: the artist recreated scene ReplicaCAD. All objects (furniture, mugs) including articulated ones (drawers, fridge) in ReplicaCAD are fully physically simulated and interactive.

Asset Creation. ReplicaCAD is an artist-created, fully-interactive recreation of 'FRL-apartment' spaces from the Replica dataset [2]. First, a team of 3D artists authored individual 3D models (geometry, textures, and material specifications) to faithfully recreate nearly all objects (furniture, kitchen utensils, books, *etc*.; 92 in total) in all 6 rooms from the FRL-apartment spaces as well as an accompanying static backdrop (floor and walls). Fig. 2 compares a layout of ReplicaCAD with the original Replica scan. Next, each object was prepared for rigid-body simulation by authoring physical parameters (mass, friction, restitution), collision proxy shapes, and semantic annotations. Several objects (*e.g*. refrigerator, kitchen counter) were made 'articulated' through sub-part segmentation (annotating fridge door, counter cabinet) and authoring of URDF files describing joint configurations (*e.g*. fridge door swings around a hinge) and dynamic properties (*e.g*. joint type and limits). For each large furniture object (*e.g.* table), we annotated surface regions (*e.g.* table tops) and containment volumes (*e.g.* drawer space) to enable programmatic placement of small objects on top of or within.

Human Layout Generation. Next, a 3D artist authored an additional 5 semantically plausible 'macro variations' of the scenes – producing new scene layouts consisting only of larger furniture from the same 3D object assets. Each of these macro variations was further perturbed through 20 'micro variations' that re-positioned objects – *e.g.* swapping the locations of similarly sized tables or a sofa and two chairs. This resulted in a total of 105 scene layouts that exhibit major and minor semantically-meaningful variations in furniture placement and scene layout, enabling controlled testing of generalization. Illustrations of these variations can be found in Appendix A.

Procedural Clutter Generation. To maximize the value of the human-authored assets we also develop a pipeline that allows us to generate new clutter procedurally. Specifically, we dynamically populate the annotated supporting surfaces (*e.g.* table-top, shelves in a cabinet) and containment volumes (*e.g.* fridge interior, drawer spaces) with object instances from appropriate categories (*e.g.*, plates, food items). These inserted objects can come from ReplicaCAD or the YCB dataset [59]. We compute physically-stable insertions of clutter offline (*i.e.* letting an inserted bowl 'settle' on a shelf) and then load these stable arrangements into the scene dynamically at run-time.

ReplicaCAD is fully integrated with the H2.0 and a supporting configuration file structure enables simple import, instancing, and programmatic alternation of any of these interactive scenes. Overall, ReplicaCAD represents 900+ person-hours of professional 3D artist effort so far (with augmentations in progress). It was created with the consent of and compensation to artists, and will be shared under a Creative Commons license for non-commercial use with attribution (CC-BY-NC). Further ReplicaCAD details and statistics are in Appendix A.

4 Habitat 2.0 (H2.0): a Lazy Simulator

H2.0's design philosophy is that speed is more important than the breadth of capabilities. H2.0 achieves fast rigid-body simulation in large photo-realistic 3D scenes by being lazy and only simulating what is absolutely needed. We instantiate this principle via 3 key ideas – localized physics and rendering (Sec. 4.1), interleaved physics and rendering (Sec. 4.2), and simplify-and-reuse (Sec. B.1). We also describe motion planning integration in Appendix B.2.

4.1 Localized Physics and Rendering

Realistic indoor 3D scenes can span houses with multiple rooms (kitchen, living room), hundreds of objects (sofa, table, mug) and 'containers' (fridge, drawer, cabinet), and thousands of parts (fridge shelf, cabinet door). Simulating physics for every part at all times is slow and unnecessary. We leverage Bullet's built-in island sleep system to minimize simulation overhead for idle objects. In addition, we make several optimizations: (1) We employ a navigation mesh to move the robot base kinematically (which has been shown to transfer well to real the world [60]) rather than simulating wheel-ground contact. (2) For multi-body articulated furniture, we remove static parts (e.g. the walls and floor of a cabinet) from the Bullet multi-body and instead load these as separate static rigid objects. This improves the sleeping behavior of the entire simulation, for example, an idle object resting on the floor of the cabinet can sleep even while the cabinet door is moving. (3) We use the sleeping state of objects to optimize rendering by caching and re-using scene graph transformation matrices and frustum-culling results.

4.2 Interleaved rendering and physics

Most physics engines (*e.g.* Bullet) run on the CPU, while rendering (*e.g.* via Magnum) typically occurs on the GPU. After our initial optimizations, we found each to take nearly equal compute-time. This represents a *glaring* inefficiency – as illustrated in Fig. 3, at any given time either the CPU is sitting idle waiting for the GPU or vice-versa. Thus, interleaving them leads to significant gains. However, this is complicated by a sequential dependency – state transitions depend on robot actions $\mathcal{T} : (s_t, a_t) \to s_{t+1}$, robot actions depend on the sensor observations: $\pi : o_t \to a_t$, and observations depend on the state $\mathcal{O} : s_t \to o_t$. Thus, it ostensibly appears that physics and rendering outputs (s_{t+1}, o_t) cannot be computed in parallel from s_t because computation of a_t cannot begin till o_t is available. We break this sequential dependency by changing the agent policy to be $\pi(a_t \mid o_{t-1})$ instead of $\pi(a_t \mid o_t)$. Thus, our agent predicts the current action a_t not from the current observations o_t but from an observation from 1 timestep ago o_{t-1}, essentially 'living in the past and acting in the future'.

This simple change means that we can generate s_{t+1} on the CPU at the same time as o_t is being generated on the GPU.

This strategy not only increases simulation throughput, but also offers two other fortuitous benefits – increased biological plausibility and improved sim2real transfer potential. The former is due to closer analogy to all sensors (biological or artificial) having a sensing latency (*e.g.*, the human visual system has approximately 150ms latency [61]). The latter is due to a line of prior work [62–64] showing that introducing this latency in simulators improves the transfer of learned agents to reality.

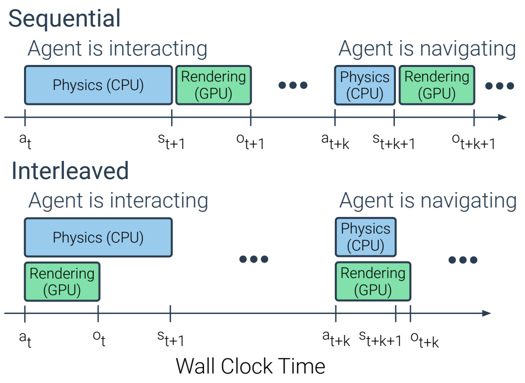


Figure 3: Interleaved physics and rendering. Top shows the normal sequential method of performing physics $(s_t, a_t) \to s_{t+1}$ then rendering $s_{t+1} \to o_{t+1}$. Bottom shows H2.0's interleaved physics and rendering.

4.3 Benchmarking

We benchmark using a Fetch robot, equipped with (up to) two RGB-D cameras (128×128 pixels) in ReplicaCAD scenes under three scenarios: (1) Idle-1×RGB: with the robot initialized in the center of the living room somewhat far from furniture or any other object and taking random actions and equipped with a single RGB camera, (2) Idle-2×RGB-D, Idle-RGB with two RGB-D cameras, (3) Interact: with the robot initialized fairly close to the fridge and taking actions from a pre-computed trajectory that results in representative interaction with objects and equipped with two RGB-D cameras. Each simulation step consists of 1 rendering pass and 4 physics-steps, each simulating 1/120 sec for a total of 1/30 sec. New joint position goals are set every 1/30 sec and a joint controller computes the joint torques to achieve the joint goals for the current joint state every 1/120 sec. This is a fairly standard experimental configuration in robotics (with 30 FPS cameras and 120 Hz control). In this setting, a simulator operating at 30 steps per (wallclock) second (SPS) corresponds to 'real time'.

Benchmarking was done on machines with dual Intel Xeon Gold 6226R CPUs – 32 cores/64 threads (32C/64T) total – and 8 NVIDIA GeForce 2080 Ti GPUs. For single-GPU benchmarking processes are confined to 8C/16T of one CPU, simulating an 8C/16T single GPU workstation. For single-GPU multi-process benchmarking, 16 processes were used. For multi-GPU benchmarking, 64 processes were used with 8 processes assigned to each GPU. We used python-3.8 and gcc-9.3 for compiling H2.0. We report average SPS over 10 runs and a 95% confidence-interval computed via standard error of the mean. Note that 8 processes do not fully utilize a 2080 Ti and thus multi-process multi-GPU performance may be better on machines with more CPU cores.

	1 Process			1 GPU			8 GPUs		
	Idle 1×RGB	Idle 2×RGB-D	Interact 2×RGB-D	Idle 1×RGB	Idle 2×RGB-D	Interact 2×RGB-D	Idle 1×RGB	Idle 2×RGB-D	Interact 2×RGB-D
H2.0 (Full)	1191 ±36	669 ±13	510 ±6	8186 ±47	1926 ±19	1660 ±6	25734 ±301	9542 ±71	7699 ±177
- render opts.	781 ±9	364 ±2	282 ±2	6709 ±89	1076 ±6	1035 ±3	18844 ±285	6397 ±43	5517 ±31
- physics opts.	271 ±3	252 ±3	358 ±6	2290 ±5	1270 ±30	1606 ±6	7942 ±50	5535 ±41	6119 ±51
- all opts.	242 ±2	177 ±3	224 ±3	2223 ±3	814 ±2	941 ±2	7192 ±55	3965 ±30	4829 ±50

Table 2: Benchmarking H2.0 performance: simulation steps per second (higher better) over 10 runs and a 95% confidence-interval In Idle, the agent is executing random actions but not interacting with the scene, while Interact uses a precomputed trajectory and thus results in representative interaction with objects. To put these numbers into context, see Tab. 1. Reproduce these numbers at https://aihabitat.org/docs/habitat2.

Table 2 reports benchmarking numbers for H2.0. We make a few observations. The ablations for H2.0 (denoted by '*- render opts*', '*-physics opts*', and '*-all opts.*') show that principles followed in our system design lead to significant performance improvements.

Our 'Idle-1×RGB' setting is similar to the benchmarking setup of iGibson [36], which reports 100 SPS. In contrast, H2.0 single-process *with all optimizations turned off* is 240% faster (242 vs 100 SPS). H2.0 single-process with optimizations on is ∼1200% faster than iGibson (1191 vs 100 SPS). The comparison to iGibson is particularly illustrative since it uses the 'same' physics engine (PyBullet) as H2.0 (Bullet). We can clearly see the benefit of working with the low-level C++ Bullet rather than PyBullet and the deep integration between rendering and physics. However, we note the comparison between the two benchmarks is not exact since the robot type, number of objects, and object assets are different. A direct comparison against other simulators is not feasible due to different capabilities, assets, hardware, and experimental settings. But a qualitative order-of-magnitude survey is illustrative – AI2-THOR [6] achieves 60/30 SPS in idle/interact, SAPIEN [34] achieves 200/400 SPS (personal communication), TDW [7] achieves 5 SPS in interact, and RLBench [35] achieves between 1 and 60 SPS depending on the sensor suite (personal communication). Finally, H2.0 scales well – achieving 8,186 SPS (272$\times$ real-time) multi-process on a single GPU and 25,734 SPS (850$\times$ real-time) on a single node with 8 GPUs. These 100$\times$ simulation-speedups correspond to cutting experimentation time from 6-month cycle to under 2 days. iGibson also supports multi-process parallization for speeding simulation speeds beyond the reported single process numbers in the current version of the paper [36], and we recommend following updates of their work for more details.

5 The Pick Task: a Base Case of Rearrangement

We first carry out systematic analyses on a relatively simple robotic manipulation task: picking up one object from a cluttered 'receptacle'. This forms a 'base case' and an instructive starting point that we eventually expand to the more challenging Home Assistant Benchmark (HAB) (Sec. 6).

Task Definition: `Pick` (s^0). Fig. 4 illustrates an episode in the pick task. Our agent (a Fetch robot [1]) is spawned close to a receptacle (a table) that holds multiple objects (*e.g.* cracker box, bowl). The task for the robot is to pick up a target object with center-of-mass coordinates $s^0 \in R^3$ (provided in robot's coordinate system) as efficiently as possible without excessive collisions. We study systematic generalization to new clutter layout on the receptacle, to new objects, and to new receptacles. **Agent embodiment and sensing.** Fetch [1] is a wheeled base with a 7-DoF arm manipulator and a parallel-jaw gripper, equipped with two RGBD cameras (90° FoV, 128×128 pixels) mounted on its 'head' and arm. It can sense its proprioceptive-state – arm joint angles (7-dim), end-effector position (3-dim), and base-egomotion (6-dim, also known as GPS+Compass in the navigation literature [3]). Note: the episodes in `Pick` are constructed such that the robot does not need to move its base. Thus, the egomotion sensor does not play a role in `Pick` but will be important in HAB tasks (Section 6).

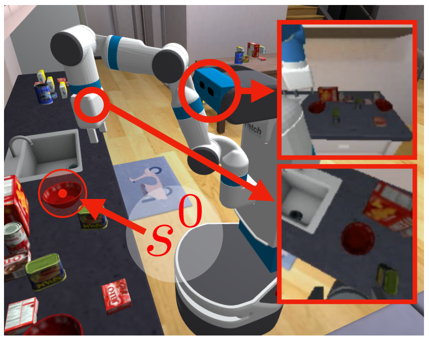


Figure 4: Fetch with head and arm cameras picking up a bowl from the counter.

Action space: gross motor control. The agent performs end-effector control at 30Hz. At every step, it outputs the desired *change* in end-effector position $(\delta x, \delta y, \delta z)$; the desired end-effector position is fed into an inverse kinematics solver from PyBullet [37] to derive desired states for all joints, which are used to set the joint motor targets, achieved using PD control. The maximum end-effector displacement per step is 1.5cm, and the maximum impulse of the joint motors is 10Ns with a position gain of Kp=0.3. In `Pick`, the base is fixed but in HAB, the agent also emits linear and angular velocities for the base.

Abstracted grasping. The agent controls the gripper by emitting a scalar. If this scalar is positive and the gripper is not currently holding an object and the end-effector is within $15cm$ of an object, then the object closest to the end-effector is snapped into the parallel-jaw gripper. The grasping is perfect and objects do not slide out. If the scalar is negative and the gripper is currently holding an object, then the object currently held in the gripper is released and simulated as falling. In all other cases, nothing happens. For analysis of other action spaces see Appendix F.6.

Evaluation. An object is considered successfully picked if the arm returns to a known 'resting position' with the target object grasped. The agent fails if the accumulated contact force experienced by the arm/body exceeds a threshold of 5k Newtons. If the agent picks up the wrong object, the episode terminates. Once the object is grasped, the drop action is masked out meaning the agent will never release the object. The episode horizon is 200 steps.

Methods. We compare two methods representing two distinctive approaches to this problem: 1. **MonolithicRL**: a 'sensors-to-actions' policy trained end-to-end with reinforcement learning (RL). The visual input is encoded using a CNN, concatenated with embeddings of proprioceptive-sensing and goal coordinates, and fed to a recurrent actor-critic network, trained with DD-PPO [11] for 100

Million steps of experience (see Appendix C for details). This baseline translates our community's most-successful paradigm yet from navigation to manipulation.

2. **SensePlanAct** (**SPA**) pipeline: Sensing consists of constructing an accumulative 3D point-cloud of the scene from depth sensors, which is then used for collision queries. Motion planning is done using Bidirectional RRT [65] in the arm joint configuration space (see Appendix D). The controller was described in 'Action Space' above and is consistent with **MonolithicRL**. We also create **SensePlanAct-Priviledged** (**SPA-Priv**), that uses *privileged* information – perfect knowledge of scene geometry (from the simulator) and a perfect controller (arm is kinematically set to desired joint poses). The purpose of this baseline is to provide an upper-bound on the performance of **SPA**.

Systematic Generalization. With H2.0 we can compare how learning based systems generalize compared to **SPA** architectures. Tab. 3 shows the results of a systematic generalization study of 4 unseen objects, 3 unseen receptacles, and 20 unseen apartment layouts (from 1 unseen 'macro variation' in ReplicaCAD). In training the agent sees 9 objects from the YCB dataset kitchen and food categories (chef can, cracker box, sugar box, tomato soup can, tuna fish cap, pudding box, gelatin box, potted meat can, and bowl). During evaluation it is tested on 4 unseen objects (apple, orange, mug, sponge). Likewise, the agent is trained on the counter, sink, light table, cabinet, fridge, dark table, and sofa receptacles (view in Fig. 11) but evaluated on the unseen receptacles of tv stand, shelves, and chair (view in Fig. 12).

Method	Seen	Unseen		
		Layouts	Objects	Receptacles
MonolithicRL	91.7 ± 1.1	86.3 ± 1.4	74.7 ± 1.8	52.7 ± 2.0
SPA	70.2 ± 1.9	72.7 ± 1.8	72.7 ± 1.8	60.3 ± 2.0
SPA-Priv	77.0 ± 1.7	80.0 ± 1.6	79.2 ± 1.7	60.7 ± 2.0

Table 3: `Pick` generalization analysis: success rates with mean and standard error on 600 episodes (and across 3 seeds for **MonolithicRL**).

MonolithicRL generalizes fairly well from seen to unseen layouts (91.7 $\rightarrow$ 86.3%), significantly outperforming **SPA** (72.7%) and even **SPA-Priv** (80.0%). However, generalization to new objects is challenging (91.7 $\rightarrow$ 74.7%) as a result of the new visual feature distribution and new object obstacles. Generalization to new receptacles is poor (91.7 $\rightarrow$ 52.7%). However, the performance drop of **SPA** (and qualitative results) suggest that the unseen receptacles (shelf, armchair, tv stand) may be objectively more difficult to pick up objects from since the shelf and armchair are tight constrained areas whereas the majority of the training receptacles, such as counters and tables, have no such constraints (see Fig. 12). We believe the performance of **MonolithicRL** will improve as more receptacles 3D assets become available since the training distribution was only 4 receptacles. We cannot make any such claims for **SPA**.

In the supplementary we also analyze different sensor input modalities (Appendix F.1), the surprising success of "blind" policies (Appendix F.2), the effect of different camera placements (Appendix F.3), different action spaces (Appendix F.6), the effect of the time delay on performance (Appendix F.5), and qualitative evidence of self-tracking (Appendix F.4).

6 Home Assistant Benchmark (HAB)

We now describe our benchmark of common household assistive robotic tasks. We stress that these tasks *illustrate* the capabilities of H2.0 but do not *delineate* them – a lot more is possible but not feasible to pack into a single coherent document with clear scientific takeaways.

Task Definition. We study three (families of) long-range tasks that correspond to common activities:

1. `TidyHouse`: Move 5 objects from random (unimpeded) locations back to where they belong (see Fig. 19a). This task requires no opening or closing and no objects are contained.

- Start: 5 target objects objects spawned in 6 possible receptacles (excluding fridge and drawer).
- Goal: Each target object is assigned a goal in a different receptacle than the starting receptacle.
- Task length: 5000 steps.

2. `PrepareGroceries`: Remove 2 objects from the fridge to the counters and place one object back in the fridge (see Fig. 19b). This task requires no opening or closing and no objects are contained.

- Start: 2 target objects in the fridge and one on the left counter. The fridge is fully opened.
- Goal: The goal for the target objects in the fridge are on the right counter and light table. The goal for the other target object is in the fridge.
- Task length: 4000 steps

3. `Set Table`: Get a bowl from a drawer, a fruit from fridge, place the fruit in the bowl on the table (see Fig. 19c).

- Start: A target bowl object is in one of the drawers and a target fruit object in the middle fridge shelf. Both the fridge and drawer start closed.
- Goal: The goal for the bowl is on the light table, the goal for the fruit is on top of the bowl. Both the fridge and drawer must be closed.
- Task length: 4500 steps.

The list is in increasing order of complexity – from no interaction with containers (`Tidy House`), to picking and placing from the fridge container (`Prepare Groceries`), to opening and closing containers (`Set Table`). Note that these descriptions are provided purely for human understanding; the robot operates entirely from a GeometricGoal specification [9] – given by the initial and desired 3D (center-of-mass) position of each target object i to be moved $\left(s_i^0, s_i^*\right)_{i=1}^{N}$. Thus, `Pick` (s_i^0) is a special case where $N = 1$ and s_i^* is a constant (arm resting) location. For each task episode, we sample a ReplicaCAD layout with YCB [59] objects randomly placed on feasible placement regions (see procedural clutter generation in Section 3). Each task has 5 clutter objects per receptacle. Unless specified, objects are sampled from the 'food' and 'kitchen' YCB item categories in the YCB dataset.

The agent is evaluated on unseen layouts and configurations of objects, and so cannot simply memorize. We characterize task difficulty by the required number of rigid-body transitions (*e.g.*, picking up a bowl, opening a drawer). The task evaluation, agent embodiment, sensing, and action space remain unchanged from Section 5, with the addition of base control via velocity commands. Details on episode statistics, as well as the evaluation protocols are in Appendix G.

Methods. We extend the methods from Sec. 5 to better handle the above long-horizon tasks with a high-level STRIPS planner using a parameterized set of skills: `Pick`, `Place`, `Open fridge door`, `Close fridge door`, `Open drawer`, `Close drawer`, and `Navigate`. The full details of the planner implementation and how methods are extended are in Appendix H. Here, we provide a brief overview.

1. **MonolithicRL**: Essentially unchanged from Sec. 5, with the exception of accepting a list of start and goal coordinates $\left(s_i^0, s_i^*\right)_{i=1}^{N}$, as opposed to just s_1^0.

2. **TaskPlanning+SkillsRL** (**TP+SRL**): a hierarchical approach that assumes knowledge of a perfect task planner (implemented with STRIPS [27]) and the initial object containment needed by the task planner to break down a task into a sequence of parameterized skills: `Navigate`, `Pick`, `Place`, `Open fridge door`, `Close fridge door`, `Open drawer`, `Close drawer`. Each skill is functionally identical to **MonolithicRL** in Sec. 5 – taking as input a single 3D position, either s_i^0 or s_i^*. For instance, in the `Set Table` task, let (a^0, a^*) and (b^0, b^*) denote the start and goal positions of the apple and bowl, respectively. The task planner converts this task into:

$$\overbrace{\texttt{Navigate}(b^0), \texttt{Open drawer}(b^0)}^{\text{Open Drawer}}, \overbrace{\texttt{Pick}(b^0), \texttt{Navigate}(b^*), \texttt{Place}(b^*)}^{\text{Transport Bowl}}, \overbrace{\texttt{Navigate}(b^0), \texttt{Close drawer}(b^0)}^{\text{Close Drawer}},$$

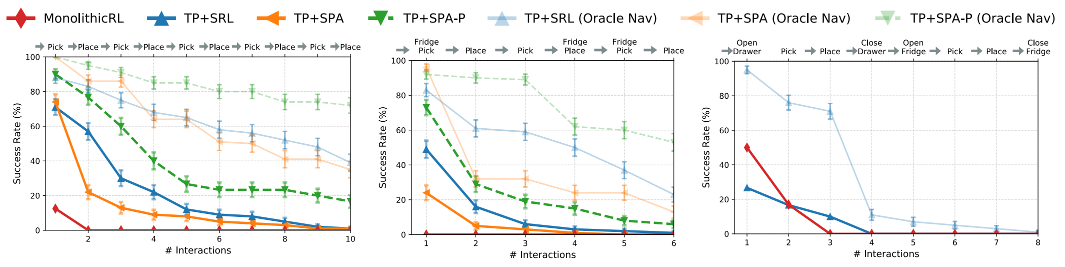
$$\underbrace{\texttt{Navigate}(a^0), \texttt{Open fridge door}(a^0)}_{\text{Open Fridge}}, \underbrace{\texttt{Navigate}(a^*), \texttt{Place}(a^*)}_{\text{Transport Apple}}, \underbrace{\texttt{Navigate}(a^0), \texttt{Close fridge door}(a^0)}_{\text{Close Fridge}}.$$

Simply listing out this sequence highlights the challenging nature of these tasks.

3. **TaskPlanning+SensePlanAct** (**TP+SPA**): Same task planner as above, with each skill implemented via **SPA** from Sec. 5 except for `Navigate` where the same learned navigation policy from **TP+SPA** is used. **TP+SPA-Priv** is analogously defined. Crafting an **SPA** pipeline for opening/closing unknown articulated containers is an open unsolved problem in robotics – involving detecting and tracking articulation [66, 67] without models, constrained full-body planning [68–70] without hand engineering constraints, and designing controllers to handle continuous contact [71, 72] – making it out of scope for this work. Thus, we do not report **TP+SPA** on `Set Table`.

Results and Findings. Figure 5 shows progressive success rates for different methods on all tasks. Due to the difficulty of the full task, for analysis, the X-axis lists the sequence of agent-environment interactions (pick, place, open, close) required to accomplish the task, same as that used by the task-planner.[4] The number of interactions is a proxy for task difficulty and the plot is analogous to precision-recall curves (with the ideal curve being a straight line at 100%). Furthermore, since

[4]This sequence from the task plan is useful for experimental analysis and debugging, but does not represent the only way to solve the task and should be disposed in future once methods improve on the full task.



(a) `TidyHouse` (b) `PrepareGroceries` (c) `SetTable`

Figure 5: Success rates for Home Assistant Benchmark tasks. Due to the difficulty of full HAB tasks, we analyze performance as completing a part of the overall task. For the TP methods that use an explicit navigation skill, we indicate with an arrow in the interaction names where navigation occurs and include versions for learned and oracle navigation. Results are on unseen layouts with mean and standard error computed for 100 episodes.

navigation is often executed between successive skills, we include versions of the task planning methods with an oracle navigation skill. We make the following observations (See Appendix I for skill learning curves and **SPA** failure statistics):

1. **MonolithicRL** performs abysmally. We were able to train *individual* skills with RL to reasonable degrees of success (see Appendix I.2). However, crafting a *combined* reward function and learning scheme that elicits chaining of such skills for a long-horizon task, without any architectural inductive bias about the task structure, remained out of our reach despite prolonged effort.

2. Learning a navigation policy to chain together skills is challenging as illustrated by the performance drop between learned and oracle navigation. In navigation for the sake of navigation (PointNav [10]), the agent is provided coordinates of the reachable goal location. In navigation for manipulation (`Navigate`), the agent is provided coordinates of a target object's center-of-mass but needs to navigate to an unspecified non-unique *suitable* location from where the object is manipulable.

3. Compounding errors hurt performance of task planning methods. Even with the relatively easier skills in `TidyHouse` in Figure 5a all methods with oracle navigation gradually decrease in performance as the number of required interactions increases.

4. Sense-plan-act variants scale poorly to increasing task complexity. In the easiest setting, `TidyHouse` with oracle navigation (Figure 5a), **TP+SPA** performs better than **TP+SRL**. However, this trend is reversed with learned navigation since **TP+SPA** methods, which rely on egocentric perception for planning, are not necessarily correctly positioned to sense the workspace. In the more complex task of `PrepareGroceries` (Figure 5b), **TP+SRL** outperforms **TP+SPA** both with and without oracle navigation due to the perception challenge of the tight and cluttered fridge. **TP+SPA** fails to find a goal configuration 3x more often and fails to find a plan in the allowed time 3x more often in `PrepareGroceries` than `TidyHouse`.

7 Societal Impacts, Limitations, and Conclusion

ReplicaCAD was modeled upon apartments in one country (USA). Different cultures and regions may have different layouts of furniture, types of furniture, and types of objects not represented in ReplicaCAD; and this lack of representation can have negative social implications for the assistants developed. While H2.0 is a fast simulator, we find that the performance of the overall simulation+training loop is bottlenecked by factors like synchronization of parallel environments and reloading of assets upon episode reset. An exciting and complementary future direction is holistically reorganizing the rendering+physics+RL interplay as studied by [73–78]. As illustrated in Figure 3, there is idle GPU time when rendering is faster than physics, because inference waits for both o_t and s_{t+1} to be ready despite not needing s_{t+1}. This is done because existing RL training systems expect the reward r_t to be returned when the agent takes an action a_t, but r_t is typically a function of s_t, a_t, and s_{t+1}. Reorganizing the rendering+physics+RL interplay is an exciting problem for future work.

We presented the ReplicaCAD dataset, the Habitat 2.0 platform and a home assistant benchmark. H2.0 is a fully interactive, high-performance 3D simulator that enables efficient experimentation involving embodied AI agents rearranging richly interactive 3D environments. Coupled with the ReplicaCAD data these improvements allow us to investigate the performance of RL policies against classical MP approaches for the suite of challenging rearrangement tasks we defined. We hope that the Habitat 2.0 platform will catalyze work on embodied AI for interactive environments.

8 Acknowledgements

Funding in direct support of this work: The Georgia Tech portion of this work supported by state funds from Georgia Tech (AS, ZK), NSF (DB), AFRL (DB), DARPA (DB), ONR YIPs (DB), ARO PECASE (DB), Amazon (DB). The SFU portion of this work is supported by a CIFAR AI Chair (AC), a Canada Research Chair (MS), and NSERC Discovery Grants (AC,MS). This work was also supported by FAIR (AC, EU, YZ, JT, NM, MM, DC, OM, AG, FM, WG) and Intel (VK).

Additional revenues related to this work: NSF (ZK), DARPA (ZK), ONR (ZK), NGA (ZK), Samsung (ZK), Airbus (ZK), consulting for Marble Inc (ZK), paid talk by Data Science Connect (ZK), sponsored research funding by FAIR (AC,MS).

References

[1] Fetch robotics. Fetch. http://fetchrobotics.com/, 2020.

[2] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[3] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019.

[4] Franka. Franka emika specification. https://www.franka.de, 2020.

[5] Unitree robotics. Aliengo. https://www.unitree.com, 2020.

[6] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-Thor: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[7] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, et al. ThreeDWorld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

[8] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[9] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied AI. *arXiv preprint arXiv:2011.01975*, 2020.

[10] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[11] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations (ICLR)*, 2020.

[12] Erik Wijmans, Irfan Essa, and Dhruv Batra. How to train pointgoal navigation agents on a (sample and compute) budget. *arXiv preprint arXiv:2012.06117*, 2020.

[13] Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation. *arXiv preprint arXiv:2007.04561*, 2020.

[14] Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. *arXiv preprint arXiv:2105.01060*, 2021.

[15] Peter Karkus, Shaojun Cai, and David Hsu. Differentiable slam-net: Learning particle slam for visual navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[16] Claudia Pérez-D'Arpino, Can Liu, Patrick Goebel, Roberto Martín-Martín, and Silvio Savarese. Robot navigation in constrained pedestrian environments using reinforcement learning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[17] Santhosh K. Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*, 2020.

[18] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Conference on Robot Learning (CoRL)*, 2019.

[19] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[20] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[21] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*, 2020.

[22] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[23] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6238. IEEE, 2020.

[24] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[25] Kaiyu Hang, Miao Li, Johannes A Stork, Yasemin Bekiroglu, Florian T Pokorny, Aude Billard, and Danica Kragic. Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation. *IEEE Transactions on robotics*, 32(4):960–972, 2016.

[26] Robin R Murphy. *Introduction to AI robotics*. MIT press, 2019.

[27] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[28] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[29] Nvidia. Isaac Sim. `https://developer.nvidia.com/isaac-sim`, 2020.

[30] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[31] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[32] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[33] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A framework for visual object manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[34] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[35] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[36] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martın-Martın, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. iGibson, a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint*, 2020.

[37] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[38] Jeongseok Lee, Michael X Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S Srinivasa, Mike Stilman, and C Karen Liu. Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500, 2018.

[39] R Smith. ODE: Open Dynamics Engine. `http://www.ode.org/`, 01 2009.

[40] Nvidia. PhysX. `https://developer.nvidia.com/gameworks-physx-overview`.

[41] Nvidia. FleX. `https://developer.nvidia.com/flex`, 2020.

[42] Hammad Mazhar, Toby Heyn, Arman Pazouki, Dan Melanz, Andrew Seidl, Aaron Bartholomew, Alessandro Tasora, and Dan Negrut. CHRONO: A parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. *Mechanical Sciences*, 4:49–64, 02 2013. doi: 10.5194/ms-4-49-2013. URL `https://projectchrono.org/`.

[43] Vladimír Vondruš and contributors. Magnum. `https://magnum.graphics`, 2020.

[44] Lilian Weng Maciek Chociej, Peter Welinder. Orrb: Openai remote rendering backend. In *eprint arXiv*, 2019. URL `https://arxiv.org/abs/1906.11633`.

[45] Matthew Matl. Pyrender. `https://github.com/mmatl/pyrender`, 2020.

[46] Unity Technologies. Unity. `https://unity.com/`.

[47] Epic Games. Unreal Engine. `https://www.unrealengine.com/`.

[48] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

[49] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[50] Claudia Yan, Dipendra Misra, Andrew Bennnett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*, 2018.

[51] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[52] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[53] HeeSun Choi, Cindy Crump, Christian Duriez, Asher Elmquist, Gregory Hager, David Han, Frank Hearl, Jessica Hodgins, Abhinandan Jain, Frederick Leve, Chen Li, Franziska Meier, Dan Negrut, Ludovic Righetti, Alberto Rodriguez, Jie Tan, and Jeff Trinkle. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1), 2021. ISSN 0027-8424. doi: 10.1073/pnas.1907856118. URL https://www.pnas.org/content/118/1/e1907856118.

[54] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *arXiv preprint arXiv:2010.01083*, 2020.

[55] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Online replanning in belief space for partially observable task and motion problems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[56] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*, 2018.

[57] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[58] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[59] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[60] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

[61] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520–522, 1996.

[62] Sandeep Singh Sandha, Luis Garcia, Bharathan Balaji, Fatima M Anwar, and Mani Srivastava. Sim2real transfer for deep reinforcement learning with stochastic state transition delays. *CoRL 2020*, 2020.

[63] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint*, 2020.

[64] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *RSS 14*, 2018.

[65] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[66] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking. In *Robotics: Science and Systems*, volume 2. Berkeley, CA, 2014.

[67] Richard Sahala Hartanto, Ryoichi Ishikawa, Menandro Roxas, and Takeshi Oishi. Hand-motion-guided articulation and segmentation estimation. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 807–813. IEEE, 2020.

[68] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

[69] Felix Burget, Armin Hornung, and Maren Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *2013 IEEE International Conference on Robotics and Automation*, pages 1656–1662. IEEE, 2013.

[70] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1:159–185, 2018.

[71] Wim Meeussen, Melonee Wise, Stuart Glaser, Sachin Chitta, Conor McGann, Patrick Mihelich, Eitan Marder-Eppstein, Marius Muja, Victor Eruhimov, Tully Foote, et al. Autonomous door opening and plugging in with a personal robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 729–736. IEEE, 2010.

[72] Advait Jain and Charles C Kemp. Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control. In *2010 IEEE International Conference on Robotics and Automation*, pages 1807–1814. IEEE, 2010.

[73] Steven Dalton, Iuri Frosio, and Michael Garland. Accelerating reinforcement learning through gpu atari emulation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[74] Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

[75] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

[76] Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. *arXiv preprint arXiv:1910.06591*, 2019.

[77] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3D control from pixels at 100000 FPS with asynchronous reinforcement learning. In *International Conference on Machine Learning*, pages 7652–7662. PMLR, 2020.

[78] Brennan Shacklett, Erik Wijmans, Aleksei Petrenko, Manolis Savva, Dhruv Batra, Vladlen Koltun, and Kayvon Fatahalian. Large batch simulation for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=cP5IcoAkfKa.

[79] Binomial LLC. Basis universal. https://github.com/BinomialLLC/basis_universal, 2020.

[80] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[81] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.

[84] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[85] Yoshiaki Kuwata, Gaston A Fiore, Justin Teo, Emilio Frazzoli, and Jonathan P How. Motion planning for urban driving using rrt. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1681–1686. IEEE, 2008.

[86] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009.

[87] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[88] Carlos Hernandez, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, et al. Team delft's robot winner of the amazon picking challenge 2016. In *Robot World Cup*, pages 613–624. Springer, 2016.

[89] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37 (11):1319–1340, 2018.

[90] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

[91] Brian Hou, Sanjiban Choudhury, Gilwoo Lee, Aditya Mandalika, and Siddhartha S Srinivasa. Posterior sampling for anytime motion planning on graphs with expensive-to-evaluate edges. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4266–4272. IEEE, 2020.

[92] Fahad Islam, Chris Paxton, Clemens Eppner, Bryan Peele, Maxim Likhachev, and Dieter Fox. Alternative paths planner (app) for provably fixed-time manipulation planning in semi-structured environments. *arXiv preprint arXiv:2012.14970*, 2020.

[93] Michael Pantic, Lionel Ott, Cesar Cadena, Roland Siegwart, and Juan Nieto. Mesh manifold based riemannian motion planning for omnidirectional micro aerial vehicles. *arXiv preprint arXiv:2102.10313*, 2021.

[94] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[95] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 3067–3074. IEEE, 2015.

[96] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

[97] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[98] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120. IEEE, 2018.

[99] Mohak Bhardwaj, Byron Boots, and Mustafa Mukadam. Differentiable gaussian process motion planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10598–10604. IEEE, 2020.

[100] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *2009 IEEE international conference on robotics and automation*, pages 625–632. IEEE, 2009.

[101] Naoki Yokoyama, Sehoon Ha, and Dhruv Batra. Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation. *arXiv preprint arXiv:2103.08022*, 2021.

[102] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations (ICLR)*, 2021.

[103] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[104] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rkl3m1BFDB.

[105] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See second paragraph of Sec. 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the first paragraph of Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and setup instructions can be found at https://github.com/facebookresearch/habitat-lab.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All methods and training details are described in detail in Sec. H.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Results in Sec. 5 are over three random seeds and 600 episodes each. Results in Sec. F.1 and Sec. F.3 are over 10 random seeds and 500 episodes each. Finally, results in Sec. 6 are over 100 episodes.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Compute resources for the benchmark are described in Section 4, for RL training in Appendix C.2, and for motion planning in Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Cited [59] for use of the YCB objects.
   (b) Did you mention the license of the assets? [Yes] ReplicaCAD is released under the Creative Commons license and H2.0 is open-sourced under the MIT license
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The ReplicaCAD dataset will be made publicly available for free prior to publication under the Creative Commons license. Download instructions can be found at https://github.com/facebookresearch/habitat-lab.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The released ReplicaCAD dataset includes furniture layouts and common kitchen items. There is no identifiable or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]