Appendix for BEHAVIOR: Benchmark for Everyday Household Activities in Virtual, Interactive, and Ecological Environments

A.1 Visualizing 100 BEHAVIOR Activities

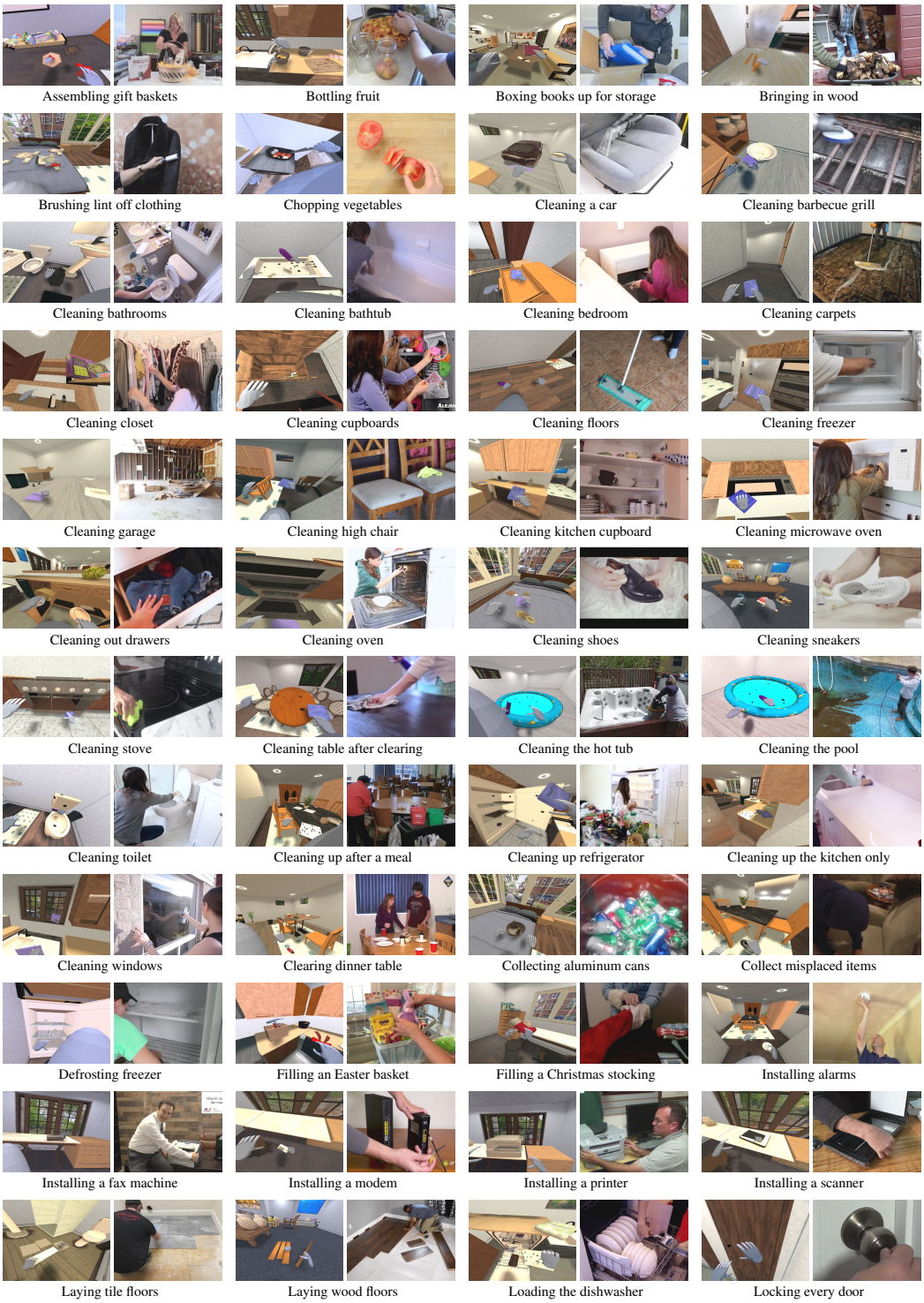


Figure A.1: **BEHAVIOR 100 activities:** Each pair of images depict a frame of the execution of the activity in BEHAVIOR from the agent's perspective in virtual reality (*left*) and the same activity in real-life from a YouTube video (*right*). All activities are selected from the American Time Use Survey [34], and correspond to simulatable household chores relevant in human's everyday life. The set of activities cover common areas like cleaning, maintenance, preparation for social activities, or household management.

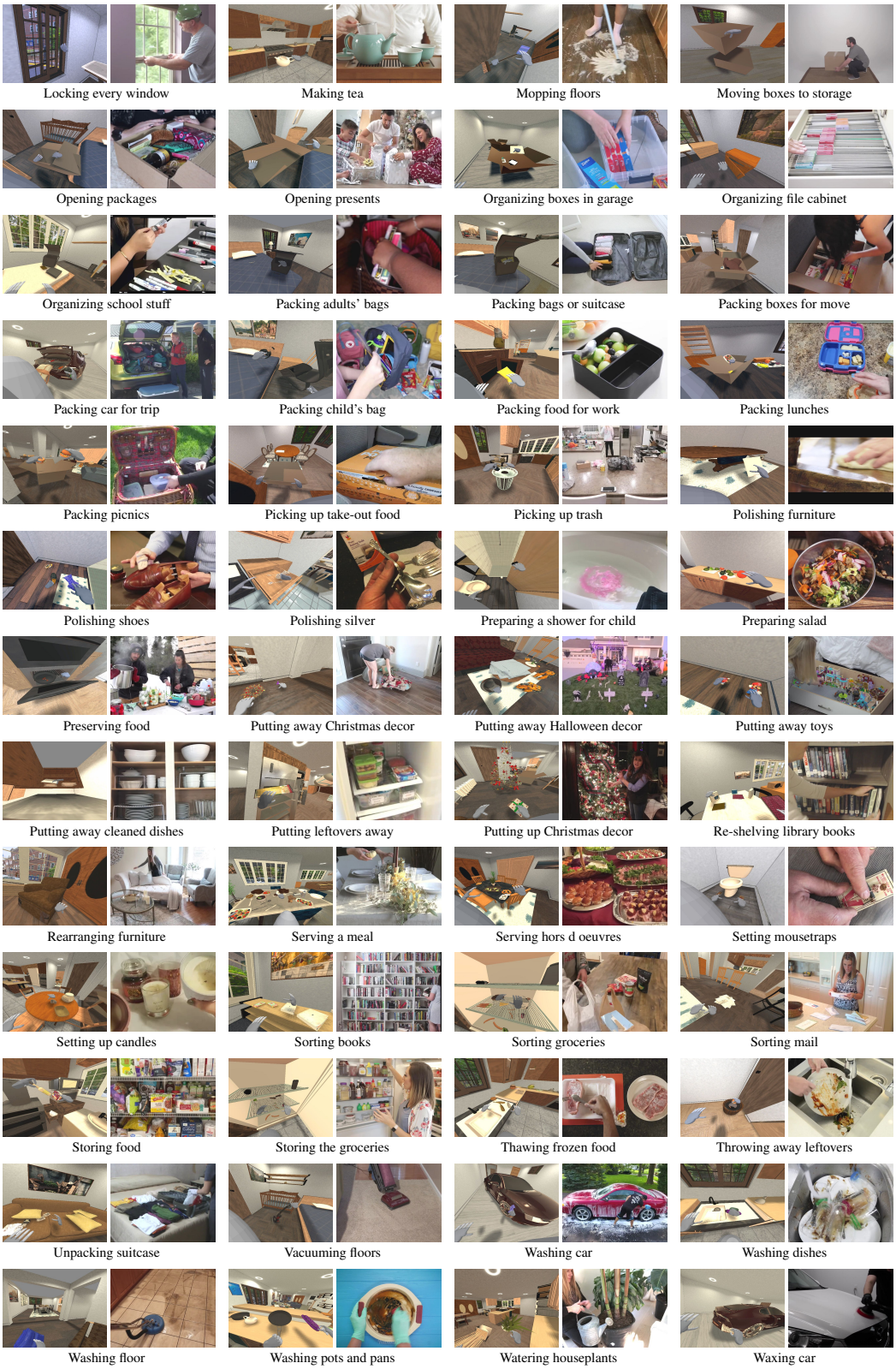


Figure A.1: **BEHAVIOR 100 activities** (cont.)

A.2 Additional Comparison between BEHAVIOR and other Embodied AI Benchmarks

			BEHAVIOR	AI2THOR Visual Room Rearrangement	TDW Transport Challenge	Rearrangement T5 (Habitat)	ManipulaTHOR ArmPointNav	Interactive Gibson Benchmark	VirtualHome	ALFRED	OCRTOC	RLBench	Metaworld	IKEA Furniture Assembly	Robosuite	SoftGym	DeepMind Control Suite	OpenAIGym	Habitat 1.0	Gibson
Realism	Realistic activities	Activities match human time-use	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
	Realistic physics	Kinematics, dynamics	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
		Continuous temperature	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
		Flexible materials	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✘	✘
	Realistic embodied AI agents	Realistic action execution	✔	✘	✔	✘	✔	✔	✘	✘	✘	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔
		Realistic observations	✔	✔	✔	✔	✘	✔	✘	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
	Realistic scenes	Visually realistic	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✘	✘	✘	✘	✔	✔
		Scenes reconstructed from real homes	✔	✘	✘	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔
	Realistic object models	Visually realistic	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	✘	✘	✔	N/A
		Weight, CoM, texture, cook temp	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	N/A
Diversity	Diverse activities	# Activities	100	1	1	1	1	2	**549**	7	5	100	50	1	5	10	28	8	2	3
		Infinite scene-agnostic instantiation	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	N/A
	Diverse scenes and objects	# Object models	**1217**	118	112	YCB	150	152		84	101 + YCB	73+	28	80	10	4	4	4	Matterport	N/A
		# Scenes / Rooms	**15 / 100**	**- / 120**	**15 / 90-120**	55 static / -	- / 30	10 / -	7 / -	- / 120	1 / -	1 / -	1 / -	1 / -	1 / -	1 / -	1 / -	1 / -	1 / -	Matterport + Gibson	572 static
	Diverse skills and activity reqs: Benchmark requires manipulating…	objects' pose	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘
		agent's global pose	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘	✔	✘	✔	✔
		objects's joint config	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✔	✘	✔	✘	✘
		objects's geom.	✔	✘	✘	✘	✘	✘	✔	✔	✘	✘	✘	✔	✘	✔	✘	✘	✘	✘
		with two hands	✔	✘	✘	✘	✘	✘	✔	✔	✘	✘	✘	✔	✘	✘	✘	✘	✘	✘
		objects's functional state (ON/OFF)	✔	✘	✘	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘	✘	✘	✘	✘
		with tools	✔	✘	✘	✘	✘	✘	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘
		object's surface	✔	✘	✘	✘	✘	✘	✘	✔	✔	✘	✘	✘	✘	✘	✔	✘	✘	✘
		objects's temp.	✔	✘	✘	✘	✘	✘	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
Complexity		Activity length (steps)	**300-20000**	<100	100-1000	100-1000	<100	100-1000	<100	<100	100-1000	<1000	<100	<100	<100	<100	<100	<100	<100	100-1000
		Objs. per activity	**3-34**	5	7-9	2-5	2-3	10	1-24	2	5-10	1-2	1-2	1	1-3	1-3	1-3	1	0-1	N/A
		# Obj. cats. in act.	2-17	1-5	7-10	2-5	1	1	**1-18**	2	1-10	1-2	1-2	1	1-2	1-3	1-3	1-2	0-1	N/A
		Diff. state changes required per activity (see A.2)	**2-8**	4	4	4	2	1-3	1-7	2-3	1	1-3	1-4	4	1	1-3	1-2	1-2	1	1
		Benchmark focus: Task-Planning and/or Control	TP+C	TP	TP+C	TP+C	TP+C	C	TP	TP	TP+C	C	TP+C	C	C	C	C	C	C	C
		# Human VR demos	**400**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		# Human VR demos	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘

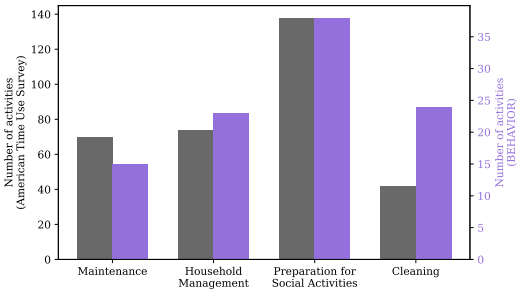
Table A.1: Comparison between BEHAVIOR and other existing benchmarks for embodied AI. Expanded version of Table 1.

A.3 Defining BEHAVIOR Activities

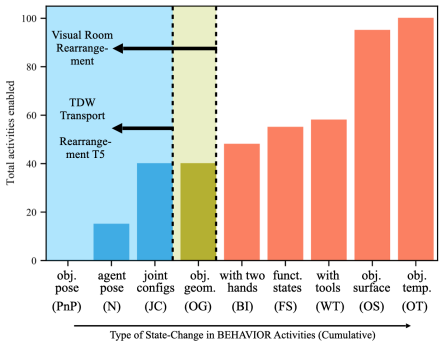
This section includes additional information on how we define the 100 activities of BEHAVIOR, including details on 1) the process to select them from the American Time Use Survey [34] (ATUS), 2) BDDL, the predicate logic language to define them, 3) the crowdsourcing process to generate definitions (initial and goal conditions) for the activities, 4) and real BDDL examples of the generated definitions.

A.3.1 Selection of 100 Activities for BEHAVIOR

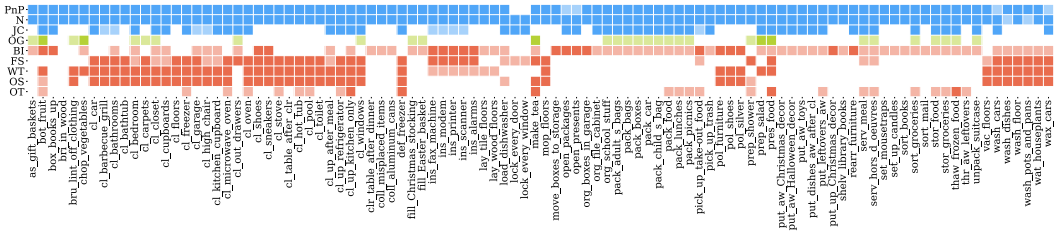
Our activities are extracted from the American Time Use Survey [34] that contains more than 2200 activities Americans spend their everyday time on. To select a subset for BEHAVIOR, we follow a set of criteria: **i) semantic diversity**: we select activities that span a wide range of semantic areas, from cleaning to food preparation, or repairing (see Fig. A.2a); **ii) diversity in the required state changes in the environment**: we select activities that requires manipulating different properties of the objects, their pose, temperature, cleanliness level, wetness level… (see Fig. A.2, b and c); and **iii) simulation feasibility**: given the current state of simulation environments, we select for BEHAVIOR activities that can be realistically simulated entirely in an indoor environment, involving only objects, most of them rigid or articulated, excluding activities outdoors, interactions with other humans or animals, or heavy simulation of flexible materials and fluids. The resulting full list of 100 BEHAVIOR activities selected can be visualized in Fig. A.1. They cover a large variety of activities such as cleaning (`CleaningBathtub`, `CleaningTheKitchenOnly`, `WashingPotsAndPans`), installing (`InstallingAScanner`, `InstallingAlarms`), waxing/polishing (`PolishingSilver`, `WaxingCarsOrOtherVehicles`), tidying (`PuttingAwayToys`, `PuttingDishesAwayAfterCleaning`), packing/assembling (`PackingPicnics`, `AssemblingGiftBaskets`), and preparing food (`PreservingFood`, `ChoppingVegetables`). Fig. A.2 depict statistics of the selected activities supporting that they approximate the semantic distribution of activities in the time use survey, and that they require a broad set changes in the environment. As comparison, Rearrangement tasks [23] and related benchmarks focus on activities that can be achieved by agent's pose (navigation), object's pose



(a) Distribution Comparison of Activities in Categories from the American Time Use Survey [34]



(b) Number of Activities in BEHAVIOR enabled by Type of State-Change (cumulative)



(c) Requirements of each Activity in BEHAVIOR

Figure A.2: **Statistics of the 100 activities in BEHAVIOR:** a) Distribution of simulatable activities in the American Time Use Survey (left axis) and BEHAVIOR (right axis) based on categories from the survey – BEHAVIOR covers a realistic distribution of activities; b) Cumulative visualization of activities enabled by different types of state changes in BEHAVIOR with comparison to recent prior work – based on requirements, some activities could be considered transport/rearrangement (blue) or visual-room rearrangement (blue and green), while others are out of their scope (red); c) We visualize the specific requirements for each of the BEHAVIOR activities, with the same coloring scheme as in b). Activities in BEHAVIOR present significantly more diverse requirements than prior work focused on transport/rearrangement tasks [23, 25, 24] enabling the evaluation of more general embodied AI solutions.

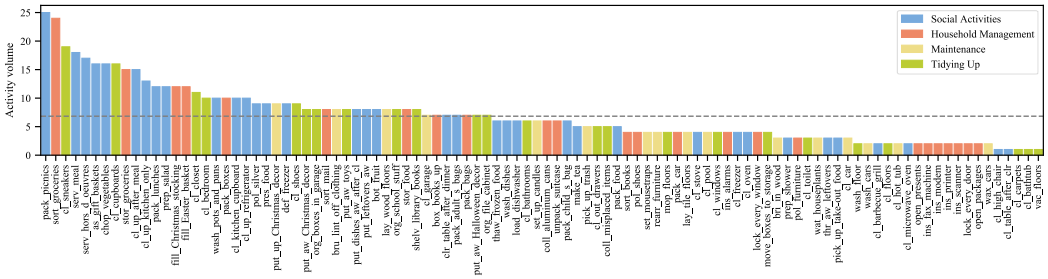


Figure A.3: **Activity volume in BEHAVIOR:** The number of literals in flattened goal conditions (volume, see Sec. A.3.2) provides a measure of the complexity of the activity and its length/horizon. The volume in BEHAVIOR activities span from one to 25 literals, very long horizon activities. Activities with one literal are often still long-horizon as they may require cleaning large surfaces (e.g. `vacuumFloors` or `cleaningBathtub`)

(pick-and-place), and joint configuration of articulated objects. VisualRoom Rearrangement [24] includes objects that can be broken (changing object geometry).

A.3.2 BDDL– BEHAVIOR Domain Definition Language

In BEHAVIOR, activities are defined using a new predicate logic language, BDDL, BEHAVIOR Domain Definition Language. BDDL creates a logic-symbolic counterpart to the physical state simulated by iGibson 2.0 through a set of logic functions (predicates). In this way, BDDL defines a set of symbols grounded into simulated objects and their states. The goal of BDDL is to enable defining activities in a unique, unifying language that connects to natural language to facilitate interpretability.

In this section, we provide additional information about the similarities and differences between BDDL and PDDL [58], a full description of the BDDL elements, syntax and grammar, and information about evaluation, grounding and "flattening" conditions, and the concept of "activity volume".

We adopt the common formalism of partially-observable Markov decision processes (POMDP) to represent activities in BDDL. Each activity is represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $\mathcal{O}$ is the observation space; $\mathcal{T}(s'|s, a), s \in \mathcal{S}, a \in \mathcal{A}$, is the state transition model; $\mathcal{R}(s, a) \in \mathbb{R}$ is the reward function; γ is the discount factor. Based on a full representation of the physical state, $\mathcal{S}$, the simulation environment generates realistic transitions to embodied AI agents' actions, $a \in \mathcal{A}$, i.e., physical interactions, and close-to-real observations, $o \in \mathcal{O}$, e.g., virtual images.

BDDL vs. PDDL: BDDL is inspired by PDDL [58] but distinct. BDDL's syntax, domain structure, and problem structure are a subset of standard PDDL [58]. Problems in various forms of PDDL and description logics are written in terms of state descriptions represented by ground predicates [73], in the same manner as BDDL's ground initial condition and solutions to its compositional and variable goal expression. The problems in both PDDL and BDDL are declarative [73]. Furthermore, such languages often use object category type hierarchies [73], like BDDL's WordNet-based object space. We use PDDL syntax to encourage adoption of BDDL as the field-standard description language.

However, goals and requirements are significantly distinct. PDDL aims to define a complete space for symbolic planning [73, 74]. BDDL aims to parametrize initial and goal physical states symbolically and process-agnostically. Therefore, PDDL requires definition of additional symbols for agent actions, while BDDL represents only the state. A key example is the use of PDDL in ALFRED [11]: BDDL and ALFRED specify activities in similar formats, though it is important to note that the sources are different (crowdsourcing vs. hand-definition). ALFRED uses PDDL for action plans annotated from demonstrations, while BDDL facilitates problem definitions without engaging in plans.

Finally, PDDLs' logic operators generally obey the axioms of first-order logic [74]. BDDL's quantifiers are a superset of standard PDDL's, with the additional quantifiers only obeying first-order logic for certain object spaces. This is because BDDL is designed for usability, so that annotators without a computational background can easily write flexible definitions.

BDDL Syntax: In BDDL, we consider the following syntactic elements, a subset of the syntax of predicate logic defined in Aho and Ullman [75]:

- **Predicate:** logic function that takes as input one (unary) or two (binary) objects and returns a boolean value. Examples in BDDL: `ontop`, `stained`, `cooked`.
- **Variable:** element in a logical expression representing an object of the indicated category, always bound by a quantifier. Categories in BDDL are defined by WordNet [35] synsets (semantic meaning), indicated by the label structure `categoryName.n.synsetEntry`. A variable is then indicated by a character `?` followed by the category. Examples in BDDL: `?apple.n.01`, `?table.n.02`.
- **Constant:** ground term, i.e., variable linked to a specific instance of an object. In BDDL, constants are identified by a numerical id suffix (`_n`) appended to the variable name. Examples in BDDL: `apple.n.01_1`, `table.n.02_3`).
- **Category:** attribute of a constant or variable indicating the class of object it belongs to, and therefore which predicates it can be given as input to (e.g., `cooked`, `sliceable`). Examples in BDDL: `apple.n.01`, `table.n.02`.
- **Type:** synonymous with **category**, conventional for PDDL and therefore defined for BDDL.
- **Argument:** variable or constant used as input in a predicate.
- **Atomic formula:** single predicate with an appropriate number of arguments. Example in BDDL: `(onTop(apple.n.01_1, table.n.02_1)`
- **Logic operator:** Function mapping logical expressions to new logical expressions. In BDDL we include all four propositional logic operators: `and` ($\wedge$), `or` ($\vee$), `not` ($\neg$), `if` ($\Rightarrow$), and `iff` ($\Leftrightarrow$).
- **Quantifier:** Function of a variable to map existing logical expressions to new logical expressions. In BDDL we include the standard universal quantification ($\forall$), and existential quantification ($\exists$), and additional operators: `for_n`, `for_pairs`, `for_n_pairs` (definitions below).

- **Logical expression:** expression obtained by composing atomic formulas with logical operators. Example in BDDL: `(and (onTop(apple.n.01_1, table.n.02_1)) (forall (?apple.n.01 - apple.n.01) cooked(apple.n.01)))`
- **Initial condition:** set of atomic formulas that are guaranteed to be `True` at the beginning of all instances of the associated BEHAVIOR activity. See examples in Listings 1 and 2.
- **Goal condition:** logical expression that must be `True` for the associated BEHAVIOR activity to be considered successfully executed. See examples in Listings 1 and 2.
- **Literal:** atomic formula or negated atomic formula. Example in BDDL: `not(onTop(apple.n.01_1, table.n.02_1))`
- **Fact:** ground atomic formula evaluated on the current state of the simulated world and returning a Boolean. Example in BDDL: `onTop(apple.n.01_1, table.n.02_1) = True`.
- **State:** set of facts about the current state of the simulated world providing a logical representation that can be evaluated wrt. the goal condition.

Initial and final conditions for household activities could be expressed using the aforementioned first order logic syntax combined with BEHAVIOR's predicates. However, our activities are defined by non-technical annotators through a crowdsourcing procedure. The annotators are not required to have background knowledge in formal logic or computer science. To facilitate their work, we include the following additional non-standard quantifiers:

- `for_n`: for some non-negative integer n and some object category C, the child condition must hold true for at least n instances of category C
- `for_pairs`: for two object categories $C1$ and $C2$, the child condition must hold true for some one-to-one mapping of object instances of $C1$ to object instances of $C2$ that covers all instances of at least one category
- `for_n_pairs`: for some non-negative integer n and two object categories $C1$ and $C2$, the child condition must hold true for at least n pairs of instances of $C1$ and instances of $C2$ that follow a one-to-one mapping.

Following the format of PDDL [58], in BDDL we consider two types of "files": a domain file shared for all activities, and problem files for each activity. The domain file defines all possible predicates, including object categories (corresponding in BEHAVIOR to categories from WordNet) and semantic symbolic states. Each activity in BEHAVIOR is defined by a different problem file that includes the object instances involved in the activity (categorized), the conditions for initial and final states.

Evaluating Logical Expressions: For a logical expression to be evaluated, we first decompose recursively it into subcomponents at the operators and quantifiers until we obtain a hierarchical structure of atomic formulae. Each atomic formula is composed of a predicate and arguments, i.e., a mathematical relationship on the simulated object(s) properties passed as arguments. For example, the atomic formula `(cooked apple.n.01_1)` is evaluated by checking the relevant thermal information of the simulated object `apple.n.01_1`. For details on the implementation of each predicate, see the attached cross-submission on the simulator iGibson 2.0. Once the atomic formulae have been evaluated into facts with queries to the grounding simulated object states, we compose the facts through the logical operators to obtain the overall binary result of the whole expression. The BDDL symbolic definition of logical expressions creates flexibility: see Fig. A.6 bottom row for examples of multiple correct solutions accepted by the same BDDL specification.

Instantiating and Grounding Initial Conditions: The initial conditions of an activity in BEHAVIOR are defined at the beginning of each BDDL problem file. They include a list of object constants and a set of ground literals based on these constants. Instances of a BEHAVIOR activity are simulated physical states that fulfill all literals in the conditions. In our implementation of BDDL in iGibson 2.0, the initial conditions are instantiated in the simulated state by assigning all object constants to physical objects of the appropriate category, either matching to physical objects already in the simulated scene or instantiating new ones in the locations specified by the binary atomic formulae (e.g., `ontop`, `inside`, etc.). The ground unary literals are satisfied by setting the physical states of the simulated objects according to the value their associated constants as given in the initial condition (e.g., `(not(cooked(chicken.n.01_1)))` sets the temperature of the associated `chicken.n.01_1` instance to a value that corresponds to `uncooked`). Our instantiation of BDDL in iGibson 2.0 provides a sampling mechanism of unary and binary predicates that can generate potentially infinite variations of each set of initial conditions (more details in the iGibson 2.0 submission

attached as supplementary) See Fig. A.6 top row for examples of multiple instantiations from the same BDDL specification.

"Flattening" a Goal Condition in an Activity Instance: BDDL provides a powerful mechanism to define the goal conditions in BEHAVIOR in their general form, e.g., `forall(?toy.n.01 - toy.n.01) inside(toy.n.01, box.n.01)`. As logical expression, BDDL goal conditions are independent of the concrete objects and the scene, and thus valid to all instances and capturing all variants of the solution. However, there are situations where grounding the goal conditions in the concrete instance of the activity at hand is helpful to understand the complexity (i.e., compute the activity volume), and the incremental progression towards the goal (i.e., compute the success score). Following on the previous example, for a possible goal condition of `PickingUpToys`, the activity's complexity would be very different when the condition is applied on an activity instance (scene) with 100 toys or with only 1 toy. We call goal condition "flattening" in an activity instance to the process of generating possible ground states of a specific simulated world fulfilling a condition. Flattening involves decomposing the nested structure of operators and quantifiers in the logical expression into a flat structure of disjointed conjunctions C_i of ground literals l_{j_i}, $\bigvee_{C_i} \bigwedge l_{j_i}$, and grounding the literals in all possible ways in the given instance. The final output of the flattening process is a list of *options*, each of which is a list of ground literals that would satisfy the goal condition. Because disjunctions, existential quantifiers, and `for_n, for_n_pairs` are satisfied as soon as one/n of their children is/are satisfied, our implementation of the flattening process for BDDL in BEHAVIOR acts lazily, generating only the minimal number of literals to fulfill each component of the goal condition. This prioritizes efficient solutions without losing any recall of possible solutions.

Activity Volume: The result of flattening a goal condition in an activity instance is a list of possible options to accomplish the activity, each option being a list of ground atomic literals. We define the *activity volume* as the length of the shortest flattened goal option for a given activity in a concrete instance. The activity volume provide a measure of the logical complexity of an activity, i.e., the number of atomic formulae that the agent needs to fulfill. For our previous example for `PickingUpToys`, the activity would have a volume of N for an activity instance with N toys, indicating the different complexity for an instance with 100 or with 1 toys.

A.3.3 Crowdsourcing the Annotation of Activities

Thanks to the connection in BDDL between the logical predicates and language semantics, BEHAVIOR activity definitions can be generated through crowdsourced annotation from non-experts workers, i.e., without background in computer science or logic. Through a visual interface, annotators can easily generate activity definitions in BDDL that reflect their idea of what the core of the activity is, and that are guaranteed to be simulatable in iGibson 2.0. We crowdsourced the generation of activity definitions to ensure that we do not introduce researcher biases in the design. The annotator pool was sourced from Upwork [76], limiting to Upwork freelancers based in the United States of America to maintain consistency and familiary with ATUS activities. Each annotator was given a salary of $15 per annotation, roughly $20-30 per hour. Because the above process constitutes a complex annotation task, we developed a custom interface to guide and facilitate annotators' work, and guarantee simulatable output.

Annotation Process and Interface: The annotation procedure is as follows. First, the annotator is presented with a BEHAVIOR activity label. When necessary, we modify the original labels to add a numerical context, e.g., "packing **four** lunches" for the original BEHAVIOR activity "packing lunches". Then, the annotator reads the annotation instructions and enters the label into the interface. As response, the interface prompts the annotator to select one or more rooms that are relevant for the activity, and to choose objects already present in these rooms that are relevant to the activity (Fig. A.4 (a)). The annotators then select small objects from the BEHAVIOR Dataset of Objects organized in the WordNet hierarchy (Fig. A.4 (b)). To facilitate the annotation, instead of presenting the hierarchy for the entire BEHAVIOR Dataset, we preselect the most possible categories per activity based on a parsing procedure on how-to articles retrieved online, primarily from wikiHow [77] (see Sec. A.5). However, annotators can access the full hierarchy if the preselected items are not sufficient.

After this first phase to select activity-relevant objects, the annotator enters the second phase to annotate initial and goal conditions. First, they are introduced to a block-based, visual tool to generate BDDL (Fig. A.4 (c)) built on Blockly [78], which makes generating logical expressions intuitive and

Figure A.4: Sections of the interface given to activity definition annotators. ⓐ shows selection of relevant rooms and scene objects. For the purpose of creating definitions compatible with multiple iGibson 2.0 scenes and likely to fit with new scenes, annotators were allowed to pick scene objects from the intersection of object sets in three pre-selected scenes. ⓑ shows selection of additional objects that would be added to the scene during activity instantiation, sourced from wikiHow [77] and taxonomized via WordNet [35]. ⓒ shows examples of the Blockly [78] version of BDDL, and ⓓ shows the prompt for initial conditions and an example for a simple "packing lunches" definition. ⓔ shows the decision of the agent's start point and the interface for "checking feasibility", i.e. confirming that the BDDL is syntactically correct, the intial and goal conditions are satisfiable, and the set-up can be physically simulated in iGibson 2.0 by attempting a sampling in an iGibson 2.0 instance on a remote server. Not shown: introductory instructions, goal condition prompt and example (similar to initial condition), some BDDL blocks, remote server communication. Full interface available online: http://verified-states.herokuapp.com (server currently disabled).

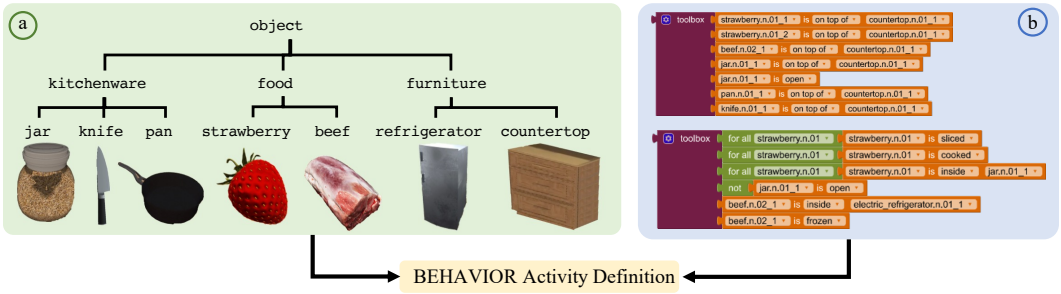


Figure A.5: **Activity annotation process for `preserving_food`:** a) annotators select objects from the WordNet organized BEHAVIOR Dataset of Objects; b) the selected objects are composed into logical predicates in BDDL for initial and final conditions using a visual interface derived from Blockly [78]; the result is a BDDL definition of the activity as logic predicates connected by logic operators and quantifiers, grounded in simulatable objects with physical properties

accessible to people without a programming background [26]. They use the tool to generate initial and goal conditions based on their concept of the activities (Fig. A.4 (d)). The resulting definitions have several guarantees: 1) they only use objects from the BEHAVIOR Dataset of Objects, 2) they only apply logical predicates to objects in a semantically meaningful manner (e.g., `cooked` can

only be applied to `cookable` objects such as food). This is because blocks' predicate fields are conditioned on entered categories that have been annotated with possible predicates in a separate manual WordNet annotation. 3) They will be in syntactically correct BDDL, through the implemented translation from Blockly. 4) They will not contain free variables or logically unsatisfiable conditions. 5) It will be possible to simulate them physically in at least three simulated scenes from iGibson 2.0. To guarantee feasibility, we assigned three possible home scenes from iGibson 2.0 to each activity and let the annotators evaluate the feasibility of their conditions at any point by clicking a button "Check feasibility" (Fig. A.4 (e)). The request will send the BDDL definition to up to three iGibson 2.0 simulators on a remote server that will attempt to sample the initial conditions and check if the goal conditions are feasible, returning real-time feedback to the annotators to correct any unfeasible condition. With the crowdsourcing procedure we obtain two alternative definitions per activity that are guarantee to be feasible in at least three simulated scenes.

A.3.4 Example Definitions

```
(define
  (problem packing_lunches_1)
  (:domain igibson)

  (:objects
    shelf.n.01_1 - shelf.n.01
    water.n.06_1 - water.n.06
    countertop.n.01_1 - countertop.n.01
    apple.n.01_1 - apple.n.01
    electric_refrigerator.n.01_1 -
        electric_refrigerator.n.01
    hamburger.n.01_1 - hamburger.n.01
    basket.n.01_1 - basket.n.01
  )

  (:init
    (ontop water.n.06_1 countertop.n.01_1)
    (inside apple.n.01_1
        electric_refrigerator.n.01_1)
    (inside hamburger.n.01_1
        electric_refrigerator.n.01_1)
    (ontop basket.n.01_1 countertop.n.01_1)
    (inroom countertop.n.01_1 kitchen)
    (inroom electric_refrigerator.n.01_1
        kitchen)
    (inroom shelf.n.01_1 kitchen)
  )

  (:goal
    (and
      (for_n_pairs
        (1)
        (?hamburger.n.01 - hamburger.n.01)
        (?basket.n.01 - basket.n.01)
        (inside ?hamburger.n.01 ?basket.n.01)
      )
      (for_n_pairs
        (1)
        (?basket.n.01 - basket.n.01)
        (?water.n.06 - water.n.06)
        (inside ?water.n.06 ?basket.n.01)
      )
      (for_n_pairs
        (1)
        (?basket.n.01 - basket.n.01)
        (?apple.n.01 - apple.n.01)
        (inside ?apple.n.01 ?basket.n.01)
      )
      (forall
        (?basket.n.01 - basket.n.01)
        (ontop ?basket.n.01 ?countertop.n.01_1)
      )
    )
  )
)
```

Listing 1: `packing_lunch`

```
(define
  (problem serving_hors_d_oeuvres_1)
  (:domain igibson)

  (:objects
    tray.n.01_1 tray.n.01_2 - tray.n.01
    countertop.n.01_1 - countertop.n.01
    oven.n.01_1 - oven.n.01
    sausage.n.01_1 sausage.n.01_2 - sausage.n.01
    cherry.n.03_1 cherry.n.03_2 - cherry.n.03
    electric_refrigerator.n.01_1 -
        electric_refrigerator.n.01
  )

  (:init
    (ontop tray.n.01_1 countertop.n.01_1)
    (ontop tray.n.01_2 countertop.n.01_1)
    (inside sausage.n.01_1 oven.n.01_1)
    (inside sausage.n.01_2 oven.n.01_1)
    (inside cherry.n.03_1
        electric_refrigerator.n.01_1)
    (inside cherry.n.03_2
        electric_refrigerator.n.01_1)
    (inroom oven.n.01_1 kitchen)
    (inroom electric_refrigerator.n.01_1
        kitchen)
    (inroom countertop.n.01_1 kitchen)
  )

  (:goal
    (and
      (exists
        (?tray.n.01 - tray.n.01)
        (and
          (forall
            (?sausage.n.01 - sausage.n.01)
            (ontop ?sausage.n.01 ?tray.n.01)
          )
          (forall
            (?cherry.n.03 - cherry.n.03)
            (not
              (ontop ?cherry.n.03 ?tray.n.01)
            )
          )
        )
      )
      (exists
        (?tray.n.01 - tray.n.01)
        (and
          (forall
            (?cherry.n.03 - cherry.n.03)
            (ontop ?cherry.n.03 ?tray.n.01)
          )
          (forall
            (?sausage.n.01 - sausage.n.01)
            (not
              (ontop ?sausage.n.01 ?tray.n.01)
            )
          )
        )
      )
    )
  )
)
```

Listing 2: `serving_hors_doeuvres`

In Listings 1 and 2, we include two examples of activity definitions (initial and goal conditions) in BDDL. They are generating by mapping the input from crowdsourcing workers in our Blockly-like interface into BDDL language. The activities include several objects and predicates in the initial and goal specifications.

A.4 iGibson 2.0

While BEHAVIOR is agnostic to the underlying simulator, we provide a fully functional instantation in iGibson 2.0. The details about iGibson 2.0 can be found in the cross-submission included in the supplementary material. Here we summarize its most important features in relation to BEHAVIOR.

Agents – Realistic Sensing and Actuation: We implement in iGibson 2.0 the two embodied agents mentioned in Sec. 5 to perform BEHAVIOR activities: a bimanual humanoid and a Fetch robot. Agents embodying the **bimanual humanoid** must control 24 degrees of freedom (DoF) to navigate, move and grasp (1 continuous DoF) with the hands, and move the pose of the head that controls the camera point of view. The continuous action space for the bimanual humanoid includes a single degree of freedom per hand, that map to a coordinated closing movement of all the revolute joints in the fingers and thumb. In the VR interface, this continuous degree of freedom corresponds to the degree of actuation of the trigger in the HTC Vive hand controllers. To facilitate the use of the complex humanoid hand and make the VR experience more natural and the efficiency of humans closer that in the real task, in iGibson grasping with the humanoid is assisted by a mechanism: the assistive grasping creates an additional kinemtic constraint between the hand and an object when 1) the degree of freedom controlling the closure of the hand goes over a certain threshold, and 2) an object is detected to be between the fingers and the palm (through ray tracing). This additional constraint breaks if the force between hand and object surpasses a threshold, forcing the agent to use grasping strategies closer to the ones required in real world (e.g. grasping a heavy object may require two hands). Additional information about the implementation of the assistive grasping can be found in the cross-submission iGibson 2.0. The bimanual humanoid is the embodiment used by humans in VR due to its similarity to human body; all demonstrations in the BEHAVIOR dataset were collected with the bimanual humanoid. Agents embodying the **Fetch robot** control 12 or 13 DoF: the navigating motion of the base, the pose of the end-effector (6 DoF), or alternatively, the joint configuration of the arm (7 DoF), one continuous prismatic joint to grasp and release (similar to the real robot), and pan/tilt motion of the head that moves the cameras. The fetch robot uses entirely physically simulated grasping, with the assisted grasping implementation disabled by default.

The sensors used by humanoid agent and Fetch leverages the realistic sensor simulation from iGibson 2.0. iGibson 2.0 features a physically-based renderer that can generate highly photorealistic RGB camera images, as well as other modalities, including depth, surface normal, semantic segmentation, instance segmentation, lidars, scene flows and optical flows. Fig. A.7 highlights a subset of the generated sensor signals.

In terms of actuation, the physical (inter)actions are simulated realistically with (py)Bullet [59], the physics engine used by iGibson 2.0. Bullet is currently one of the most used simulators in embodied AI and robotics, and has demonstrated a useful and realistic engine to develop solutions that can transfer to real world with some degree of domain adaptation. In our physics simulation we use a very small physics simulation timestep of $\frac{1}{300}$s. This help reduce physics simulation artifacts, such as objects clipping into each other, increasing realism.

Condition Checking and Sampling: The implementation of BEHAVIOR in iGibson 2.0 allows activities to be initialized, executed, and checked for completion. Given an activity definition in the BDDL, BEHAVIOR and iGibson 2.0 interface to generate a valid instance of the activity that satisfies the given object list and initial conditions. This mechanism can generate potentially infinite variation of scenes, objects and initial states to create different activity instances. In the generation of an activity instance, the goal conditions are checked for feasibility, avoiding the generation of activity instances that cannot lead to successful executions (see Fig. A.6, top). iGibson 2.0 implements all necessary checking functionalities for the logical states. These checking function execute in realtime together with the physical simulation and rendering, enabling live feedback to the agents for task completion and capturing all possible valid solutions (Fig. A.6, bottom). For more information about the condition checking and sampling, please refer to the concurrent submission iGibson 2.0 paper included as part of the supplementary material.

Figure A.6: **BDDL Initial and Goal Conditions:** Our implementation in iGibson 2.0 can generate diverse valid activity instances from each BDDL definition (top row), and detect all successful variations of the solution (bottom row), promoting diversity and semantically-meaningful activities

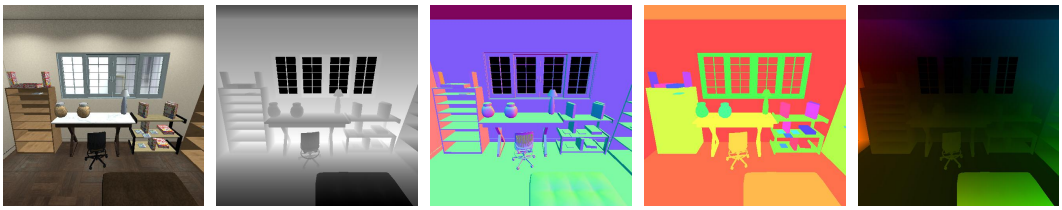


Figure A.7: **Virtual visual sensor signals generated by iGibson 2.0:** Color images are generated with a high-quality physics-based rendering procedure (PBR), exploiting the annotation of material (roughness, metallic) of all surfaces in our objects and scenes. iGibson 2.0 is able to generate RGB, depth, surface normals, semantic segmentation, instance segmentation, optical flow, scene flow and lidar (1-line and 16-line) sensors signals. Here we visualize a subset of those sensor signals, namely RGB, depth, surface normal, instance segmentation and optical flow.

Implementation of the Action Primitives: To facilitate the development of solutions and to study the effect of the activity complexity on the performance of embodied AI algorithms, we provide action primitives implemented in iGibson 2.0 and that can be used in BEHAVIOR. The action primitives are temporally extended actions. We implemented six action primitives, namely `navigate_to(obj)`, `grasp(obj)`, `place_onTop(obj)`, `place_inside(obj)`, `open(obj)`, `close(obj)`. Each primitive can be applied relative to objects in the scene. For each action primitive, we implemented two variants. The first variant is "fully-simulated motion primitive ", where we first check the feasibility of the target configuration, and then plan a full valid path between the initial and the target configurations with a sampling based motion planner [61]. The second variant is "partially-simulated motion primitive", where we only check for feasibility of the desired final configuration, and directly set the state of the world (agent and objects) to this desired configuration. This can be highly unrealistic as we do not verify if there is a valid path between the initial and the final configurations. The purpose of partially simulated motion primitive is to reduce the computation during RL training and to measure the relative complexity of generating full interactions vs. just finding the sequence of states to achieve an activity. Note that for both partially and fully simulated motion primitives, privileged information is given to the agent and the motion planner. For example, the agent knows how many activity-relevant objects are in the scene, and the motion planner knows the full geometry of the environment.

For the implementation of partially-simulated motion primitives, we only perform feasibility check when attempting to perform an action. For example, when trying to `navigate_to` an object, we will randomly sample points around the object and attempt to place the agent there: the goal is to find

a collision-free location to place the agent. The second type of feasibility check is reachability: when attempting to `grasp`, `open` or `close` an object, we will check the distance from the hand to the closest point of the object is smaller than the arm length. When we `place` an object `inside` or `onTop` of an object, we use the sampling functionality available in iGibson 2.0.

For the implementation of fully-simulated motion primitives, in additional to the feasibility check, we attempt to plan and execute a collision free. We treat all objects as obstacles except for the objects given as argument for the primitive (e.g., objects that need to be picked up, receptacles that need to be opened), and plan a collision-free path from the start configuration to the target configuration. We use Bidirectional RRT [61] for motion planning and execute the motion with position control. In our experiments, we found that fully simulated motion primitives have much lower success rate than partially simulated motion primitives (Table 2) indicating that the difficulty in BEHAVIOR strives from solving the entire interaction rather than deciding on the strategy at a task-level. Our partially-simulated primitives and other benchmarks that do not simulate the full interaction, bypass this critical challenge.

Runtime performance of iGibson 2.0: iGibson 2.0 improved performance when compared to iGibson 1.0 [79], with optimizations on both physics and rendering. To evaluate the performance of iGibson 2.0 in BEHAVIOR activities, we benchmarked the different phases of each simulation step. We benchmark the activities in "idle" setting, which means we initialize the activity, and runs the simulation and condition checking loop. The agent applies zero actions and stays still. We benchmarked in two conditions using the same action time step of t_a but different physics time step of t_s, leading to slightly different reality in the physics simulation. The action step is the simulated-time between agent's actions, while the physics time step is the simulated-time interval that the kinematics simulator (pyBullet) uses to integrate forces and compute the new kinematic states. We execute n_s queries to the simulator between agent actions, with $n_s = t_a/t_s$. The first condition we evaluate uses action time step $t_a = \frac{1}{30}$s and physics time step of $t_s = \frac{1}{300}$s, which creates high-fidelity physics simulation. The second condition uses action time step of $t_a = \frac{1}{30}$s and physics time step of $t_s = \frac{1}{120}$s, which has slightly lower physics fidelity, but has better performance and is sufficient for RL training. Both settings are benchmarked on a computer with Intel 5930k CPU and Nvidia GTX 1080 Ti GPU, in a single process setting, rendering 128×128 RGB-D images.

As shown in Table A.2, for the highest-fidelity physics setup, we can achieve 36-59 steps per second, 47-71 steps per second with larger simulated timestep, even in a very large scene with 100-200 movable objects, and with all the physical and logical states evaluated at each step. This frequencies provide pleasant experience in virtual reality. However, it only provide a $\times 2$ acceleration over clock-time to train RL agents. To increase the frequency in simulation and reduce the training time, we are exploring the parallelization of simulation and rendering and the more aggressive "sleep" of non-interacted objects.

Executing BEHAVIOR activities in iGibson 2.0: Here, we give examples of the physical requirements of possible solutions to various activities, to give a sense for what they call for. First, we consider `cleaningBathtub`: this can be solved by navigating to the tub, grasping the brush (e.g. an oblong with bristles), navigating to the sink with brush in hand, "toggling on" the faucet by making contact between the hand and the toggle button, placing the brush in contact with a falling water particle so that it becomes "soaked", navigating back to the tub, and moving the brush around so that its bristle side makes contact with at least 50% of the "stain" particles on the tub. Next, `fillingChristmasStockings`: one way to solve this, seen in VR demos, is to navigate to a bin sitting in the living room, grasp it, navigate to the kitchen with the bin in hand, open cabinet drawers to find candy and pens in one of them, grasp each of the four pens (small) and candies (small, irregular shapes) and place them into the bin, carry the bin full of candy and pens back to the living room, collect the four small blocks scattered on the floor and place each one in the bin, and then one-by-one navigate to each stocking, pick up one pen, then one candy, and one block from the bin in any order with one hand, and fit it into the narrow opening of the stocking while holding the stocking in the other hand. Finally, `preservingFood`: one possible solution is to grasp a dish from a cabinet, place it on an accessible surface, grasp two mushrooms (small, irregular shape), tomatoes (large, round), and chestnuts (small, round) off a countertop, and two onions (large, round) out of a fridge and place each one into the dish, grasp a knife by the handle, and move it so that its blade makes contact with each of the eight produce items one-by-one until it is sliced in two.

	bringing_in_wood	re-shelving_library_books	laying_tile_floors
Number of Objects	134	144	216
Simulation steps per second ($@t_s = \frac{1}{300}s$ / $@t_s = \frac{1}{120}s$)	59 / 74	51 / 68	36 / 47
Kinematic State Update Time [ms] ($@t_s = \frac{1}{300}s$ / $@t_s = \frac{1}{120}s$)	7.4 / 3.5	9.4 / 4.2	12.6 / 5.7
Non-kinematic State Update Time [ms]	3.4	3.4	5.2
Rendering Time [ms]	5.8	6.1	9.3
Logical Condition Checking Time [ms]	0.4	0.4	0.6

Table A.2: Benchmarking Simulation Time for BEHAVIOR Activities in iGibson 2.0

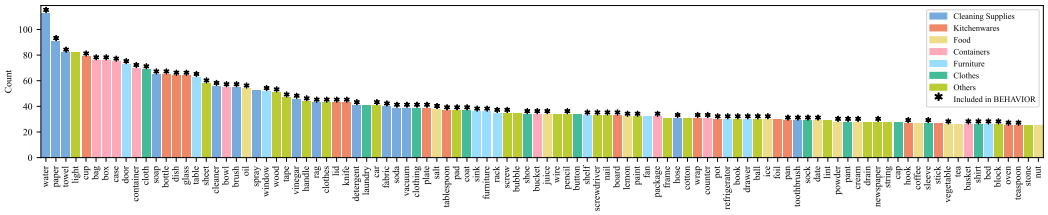


Figure A.8: **Statistics of objects in descriptions for the 100 BEHAVIOR activities:** We parse descriptions from WikiHow [77] and other online repositories of instructions for the activities in BEHAVIOR and obtain the frequencies of appearance for each noun. The nouns are mapped to corresponding synsets in WordNet [35]. The categories shown in the figure are based on the WordNet taxonomy (best seen in color). For object categories included in the BEHAVIOR Dataset of Objects, we annotate the bar with an asterisk (the plot does not depict all categories in the dataset). We include the vast majority of most frequent objects involved in the activities as indicated by the natural language descriptions.

A.5 BEHAVIOR Dataset of Objects

In order to instantiate BEHAVIOR activities in iGibson 2.0, we created a new dataset of everyday objects, the BEHAVIOR Dataset of Objects. To guide the selection of object categories, we analyze how-to articles, primarily WikiHow [77], explaining how to perform the activities included in BEHAVIOR. Specifically, we extract nouns of tangible objects from these articles that are activity-relevant, map them to WordNet synsets, and then purchase 3D models of these object categories from online marketplaces such as TurboSquid. This procedure allowed us to provide activity annotators and VR demonstrators with the most frequent objects necessary for the activities (see Fig. A.8).

The diversity of BEHAVIOR activities naturally leads to the diversity of the object dataset. In total, we curate 1217 object models across 391 object categories, to support 100 BEHAVIOR activities. The categories range from food items to tableware, from home decorations to office supplies, and from apparel to cleaning tools. In Fig. A.8, we observe that the BEHAVIOR Dataset of Objects cover a wide range of object categories.

To maintain high visual realism, all object models include material information (metallic, roughness) that can be rendered by iGibson 2.0 renderer. To maintain high physics realism, object models are annotated with size, mass, center of mass, moment of inertia, and also stable orientations. The collision mesh is a simplified version of the visual mesh, obtained with a convex decomposition using the VHACD algorithm. Object models with a shape close to a box are annotated with a primitive box collision mesh, much more efficient and robust for collision checking. For object categories that have the semantic property `openable` annotated, we make sure at least a subset of their object models have articulation, e.g. openable jars, backpacks, cars, etc. We either directly acquire them from the PartNet-Mobility Dataset [80] or acquire non-articulated models from TurboSquid, manually segment the models into parts, and then create the articulation in the URDF files. A subset of the object models are visualized in Fig. A.10.

We will publicly release the object dataset to be used for BEHAVIOR benchmarking. To preserve the rights of the model authors and the license agreement with TurboSquid, the 3D models are encrypted so that they can only be used within iGibson 2.0 and cannot be exported for other applications.

All models in the BEHAVIOR Dataset are organized following the WordNet [35], associating them to synsets. This structure allows us to define properties for all models of the same categories, but it also facilitates more general sampling of activity instances fulfilling initial conditions such as `onTop(fruit, table)` that can be achieved using any model within the branch `fruit` of WordNet. Fig. A.9 shows an example taxonomy of objects of the dataset organized in the WordNet taxonomy to perform a given household activity.

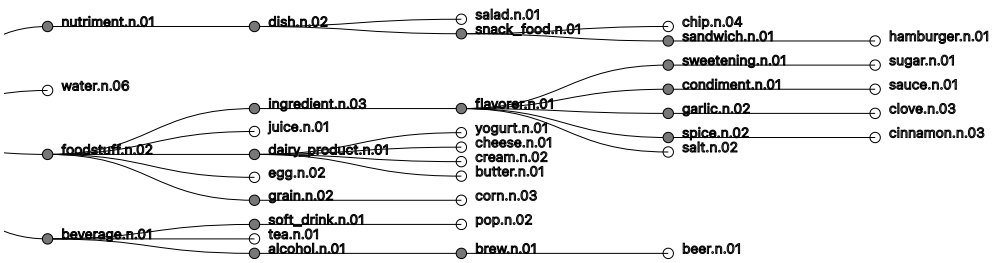


Figure A.9: **Object Taxonomy in the BEHAVIOR Dataset of Objects:** Sample extract of the objects involved in an activity in BEHAVIOR, organized based on the taxonomy from WordNet [35]; We map 3D models with annotation of physical and semantic properties to synsets in WordNet. BEHAVIOR activities in BDDL use any level entries of the WordNet taxonomy enabling the generation of more diverse instances with any object in the downstream task (e.g. any food item)

Figure A.10: **Example Models in BEHAVIOR Dataset of Objects:** A selected subset of everyday objects in the dataset to support the 100 activities in BEHAVIOR. The models present high-quality geometry, material, and texture, and are annotated with realistic physical attributes such as size, mass, center of mass and moment of inertia, and semantic properties such as `cookable`, `sliceable`, or `toggleable`.

A.6 BEHAVIOR Dataset of Human Demonstrations in Virtual Reality

The main role of the VR demonstrations in BEHAVIOR is to provide a mechanism to normalize metrics, allowing to compare different embodied AI solutions between activity instances and scenes. However, we believe that the generated dataset of VR demos has the potential to be applied to other purposes, e.g., to generate AI solutions through imitation learning, or to study the mechanisms used by humans to accomplish interactive activities. In the following, we provide additional details for users interested in the dataset of VR demonstrations. We include 1) additional information about the data collection procedure, 2) statistics of the data, and 3) information about collected human gaze data.

A.6.1 Collecting Human Demonstrations in Virtual Reality

To generate data, humans control a bimanual humanoid embodiment with a main body, two hands and a movable head based on stereo images displayed at 30 frames per second. The embodiment and the VR can be used with the most common VR hardware but for our dataset, we used a HTC Vive Pro Eye [81]. All recorded data can be deterministically replayed, achieving the same physical state transitions as reaction to the recorded physical interactions, which allows to generate any additional virtual sensor signal a posteriori. For more information about the VR interface, we provide the cross-submitted publication of iGibson 2.0 as part of the supplementary material.

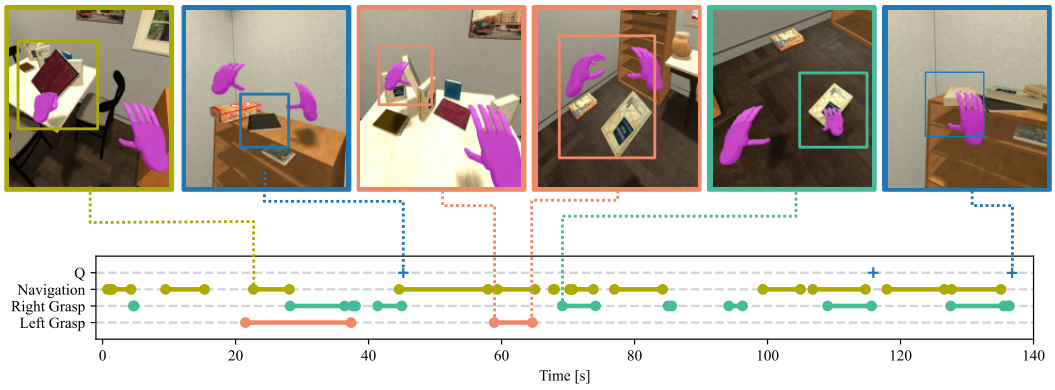


Figure A.11: **Sub-activity segmentation across activity execution for `re-shelvingLibraryBooks`:** We observe multiple cycles of long-range pick-and-place operations that eventually lead to activity success. In this figure, we show a sequence of snapshots of first-person view along with key frames (i.e. target objects placed on shelf, items dropped and picked up with alternating hands).

We collect three different demonstrations of the same activity instance (same scene, same objects, same initialization) for each of the 100 activities in BEHAVIOR, 100 additional demonstrations, one for each activity for a different instance (different objects, different initialization) in the same scene, and 100 additional demonstrations, one for each activity in a different scene. These 500 demonstrations cover both the diversity in human execution, and the dimensions of variability in activity instances of BEHAVIOR. The data has been collected by voluntary participants and our own team.

A.6.2 Analysis and Statistics of Virtual Reality Demonstrations

The BEHAVIOR Dataset of Human Demonstrations in VR provides rich data of navigation, manipulation, and problem-solving from humans for long time-horizon and multi-step activities. Analyzing the statistical characteristics of the data (duration, hand use, room visitation, etc.) provides insights on how humans achieve their level of performance combining interaction and locomotion in the large BEHAVIOR scenes. Fig. A.11 depicts the segmentation of a VR demonstration into navigation and grasping phases while performing a pick-and-place rearrangement activity. This segmentation reveals multiple initial phases of navigation as the demonstrator observes the scene and locates activity relevant objects. For example, once the demonstrator reaches the table supporting the target objects (approx. at 18s), they pick up the target object with the non-dominant hand (approx. at 20s) and navigate to the goal location, before transferring the object to their dominant hand while positioning it (approx. at 30s). The demonstrator shows a preference for moving objects one-at-a-time, instead of stacking or carrying objects with each hand; this strategy will perform more poorly on the efficiency metrics T_{sim}, L_{body}, L_{right}, and L_{left}.

Fig. A.12 includes a) the duration of the VR demonstrations, b) the time spent in different room types, c) the hand used to interact and manipulate, and d) the complexity of the activities in logical representation vs. time. We observe that BEHAVIOR activities cover a wide range of time-horizons, from 6 seconds to 11.3 minutes. This time horizon corresponds to an average of 3880 steps with a standard deviation of 3157 steps. The longest task `packingPicnics` requires the user to pack over 25 objects, which must first be discovered via exploration before packing. The activities show a bias towards living spaces (kitchen, living-room, bedroom), with the most prevalent room being the kitchen. A large portion of BEHAVIOR activities involve preparing food or cleaning appliances that are only supported in kitchens. Furthermore, as expected, the data reflect a bias towards dominant hand manipulation, followed by bimanual grasping, which is required for lifting and manipulating large objects. The high use of two hands to manipulate correlate to the use in real-world; we hope that our dataset helps exploring this type of interaction that has been traditionally less studied in embodied AI. The total number of ground predicates (activity volume) is strongly correlated with the total activity time indicating that the volume is a good measure of the complexity of an activity. Outliers include activities with a high ratio of time to goal condition such as the ones that

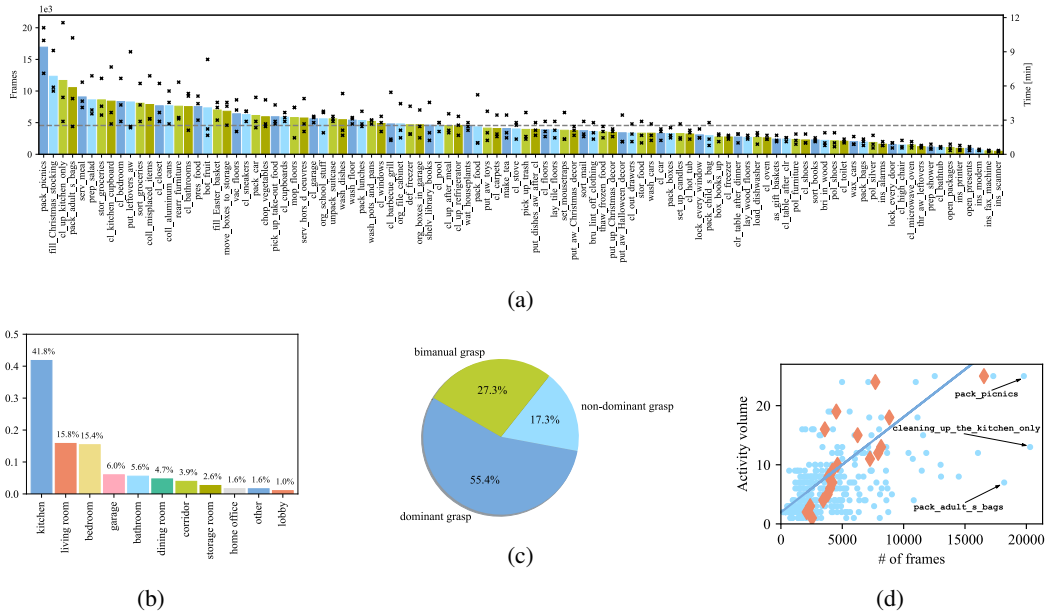


Figure A.12: **Analysis of human demonstrations of BEHAVIOR activities in virtual reality:** a) Duration of each successful demonstration (mean and individual trials, decreasing order); b) Fraction of total VR time spent in each type of room; c) Fraction of total VR time spent manipulating with the dominant, non-dominant, or both hands; d) Duration of each VR demonstration wrt. activity volume; blue dots denote individual demos and red diamonds denote the mean time for each number of ground literals (activity volume). Larger volume correlates with larger duration ($R^2 = 0.765$).

require cleaning a large area (`cleaningUpTheKitchenOnly`, `vacuumingFloors`) or searching (`packingAdultsBag`).

Analyzed individually, Fig. A.13 shows that room occupancy depends heavily on the type of activity. Room occupancy reflects common intuition about household activities; the ones associated with living-space decorations (`puttingAwayChristmasDecorations`, `puttingAwayHalloweenDecorations`) take place primarily in the living room, whereas cooking activities (`preparingSalad`, `preservingFood`) occur primarily in the kitchen. Similar activity preferences are observed in the grasping data; activities requiring installing unwieldy objects (`layingWoodFloors`, `layingTileFloors`) require the use of both hands, whereas simple cleaning activities (`cleaningThePool`) that require using a cleaning tool are performed with the dominant hand.

A.6.3 Gaze Tracking in Virtual Reality

Our preference on HTC Vive Pro to collect the BEHAVIOR Dataset of Human Demonstrations is motivated by its ability to track the gaze (pupil movement) of the demonstrator. We consider gaze information to be a valuable source to understand human performance in the activities. While other datasets of gaze are available [82], this is the largest dataset of active gaze attention during manipulation in simulation, providing synchronized ground-truth information of the object being observed, its state and full shape. Fig. A.15 depict examples of the tracked human gaze during activity execution, with the object attracting the gaze indicated in magenta. Fig. A.16 includes several statistics of the gaze attention over object categories in the entire dataset and for some example activities. Both figures indicate a clear correlation between the gaze data and the goal of the activities: we expect the dataset to be useful to study and predict human gaze attention, and to develop new embodied AI algorithms for active [83, 84] and interactive perception [85].

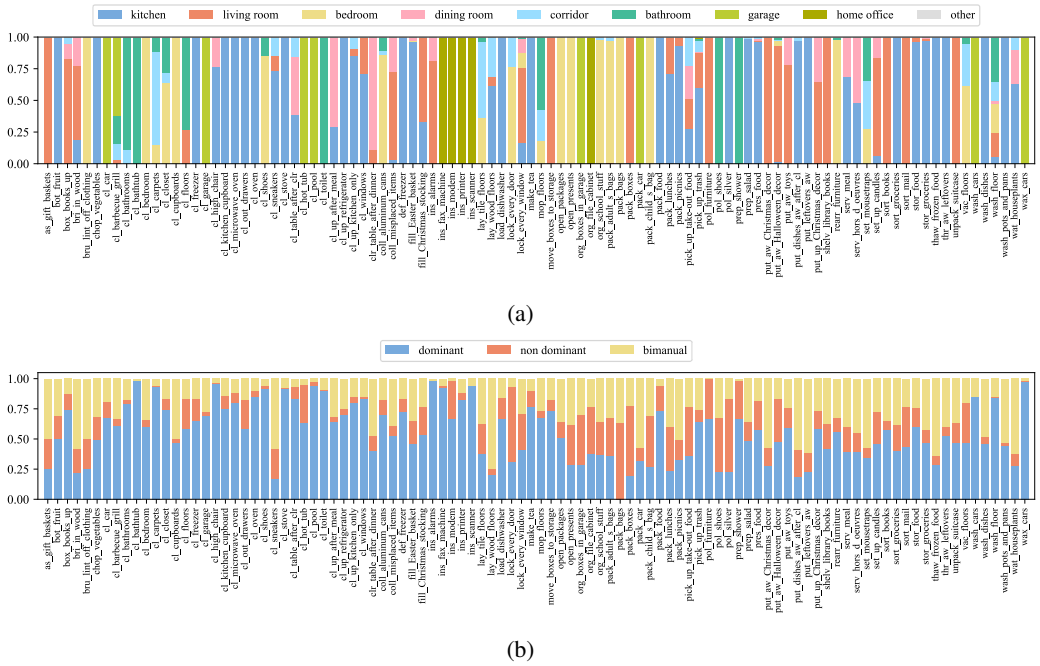


Figure A.13: **Further analysis of human demonstrations of BEHAVIOR activities in virtual reality:** a) Fraction of the duration of each activity spent in different types of room; b) Fraction of each activity spent manipulating with the right, left, or both hands; BEHAVIOR activities present a large diversity of room types: while some activities are mostly performed on a single type of room, others requires visiting different types; while the majority the activities in BEHAVIOR are performed with the dominant hand, a significant number of them require using both hands, e.g., `layingWoodFloors` or `assemblingGiftBasket`.

Figure A.14: **Navigation trajectories of humans demonstrating activities in virtual reality:** Trajectories of different demonstrators in two scenes, `Rs_int` (left, center) and `Merom_1_int` (right), for two activities, `re-shelvingLibraryBooks` (left) and `puttingAwayHalloweenDecorations` (center, right); Demonstrator trajectories present variation within activity instance and scene (each figure); Different activities in the same scene (left, center) require different rooms and areas to be explored; Trajectories differ between scenes (center, right) due to the placement of target objects and goal locations

A.7 Additional Details on the Experimental Setup

In the following, we share more details about the experimental setup and training procedure. We primarily use two different training setups: reinforcement learning (RL) with continuous action space

Figure A.15: Human gaze during activity execution; Red dot: human gaze point, Magenta: object gazed; The BEHAVIOR Dataset of Human Demonstrations in Virtual Reality includes 500 demonstrations (1077.7 min) with gaze information while humans navigate and interact (accuracy: $\pm 4^\circ$ [86]); The gaze information correlates strongly with activity; We hope that this data can support new research in visual attention and active vision to control agent's camera

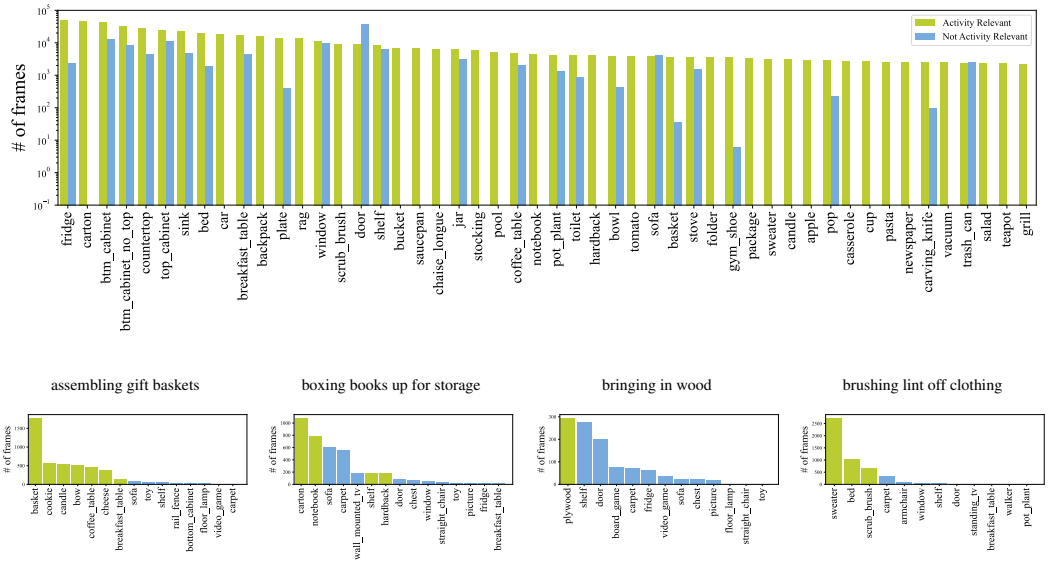


Figure A.16: Statistics of the attention over object instances of WordNet categories in the BEHAVIOR Dataset of Human Demonstrations in virtual reality aggregated for all demonstrations (top, logarithmic scale), and segregated for four activities (bottom row, linear scale), for activity-relevant (green) and not activity-relevant (blue) objects. In households, the aggregated of the visual attention goes to containers of objects (cabinets) and doors separating rooms; For individual activities, visual attention concentrate on specific activity-relevant objects

and RL with motion primitives. We will first elaborate the shared setup between the two, then go into their differences.

Shared Setup: In the normal "partial observability" setup, the observations include 128×128 RGB-D images from the onboard sensor on the agent's head and proprioceptive information (Head pose in local frame, hand poses in local frame, and a fraction indicating how much each hand is closed). The proprioceptive information is 20 dimentional. For the experiments with "full observability", the observations include the ground truth object poses for all the activity-relevant objects, the agent's pose, and the proprioceptive information.

The agent receives a reward of 1 for every ground goal condition (literal) that it satisfies during the episode. The episode terminates if the agent achieves a success score Q of 1 (achieved all literals in the goal condition) or it times out.

The policy network architecture is largely shared in the following setups. With RGB-D images as input, we use a 3-layer convolutional neural network to encode the image into a 256 dimensional vector. Proprioceptive information and/or poses for all activity-relevant objects are also encoded into a 256 dimensional vector with an MLP, respectively. The features are concatenated and pass through another MLP to generate action representation, which could be a box action space or discrete action space depending on the setup (continuous actions or action primitives).

RL with Continuous Action Space: For this agent variant, we use Soft Actor-Critic (SAC) [16] implemented by TF-Agents [87]. The action space is continuous and has a dimensionality of 18. The first three dimensions represent the locomotion actions: desired x-y translation of the robot body and the desired rotation around the vertical axis. The next seven dimensions represent the linear and angular velocities of the left hand (in Cartesian space, 6 DoF) and 1 DoF closing/opening of the hand. The last seven dimensions is the same action but for the right hand. The maximum episode length depends on the experimental setup. For instance, if the initial state corresponds to $1\,\mathrm{s}$ away from a goal state, we will give the agent three times the amount of time (i.e. $3\,\mathrm{s}$) to accomplish the activity. We train for 20K episodes, evaluate the final policy checkpoint and report the results in Table 2.

RL with Motion Primitives: For this agent variant, we use Proximal Policy Optimization (PPO) [17] implemented with TF-Agents [87]. The action space is discrete, with $n_r \times m$ choices, where n_r is the number of activity-relevant objects and m is the number of action primitives. Here we didn't allow the agent to operate on all objects in the scene, but focus on activity-relevant objects to facilitate learning. Following our implementation of motion primitives, $m = 6$. Laying out the choices on a $n_r \times m$ grid, and i-th column j-th row means to apply j-th action primitive on i-th activity-relevant object. Not all combinations of action primitive and object are compatible and action that is not feasible is converted into no-ops. The maximum episode length is set to 100 for all activities. We experiment with partially simulated motion primitives and fully simulated motion primitives, as described in Sec. A.4). We train with partially simulated motion primitives until convergence, and evaluate and report the results on partially simulated motion primitives and fully simulated motion primitives, since training with motion planning in a complex scene is very time-consuming. In the experimental results shown in Table 2, generally fully simulated motion primitives results are much worse than partially simulated motion primitives, this is intuitive because motion planning performs more rigorous checks and complies with the physical model, highlighting the complexity of BEHAVIOR.

Experimental Setup for the Effect of Diversity: To evaluate diversity, we train for individual skills instead of full BEHAVIOR activities. Here, we adopt an easier experimental setup that allows us to study the effect of diversity; the results are reported in Table 3. Specifically, we use RL with continuous action space but with a more constrained action space: 6-dimensional representing the desired linear and angular velocities of the right hand (assuming the rest of the agent is stationary). For grasping, we adopt the "sticky mitten" simplification from other works [23]: we create a fixed constraint between the hand and the object as soon as they get in contact. We also use distance-based reward shaping to encourage the hand to approach activity-relevant objects. To evaluate the effect of diversity in object poses, we use the same object models and randomize their initial poses during training. To evaluate the effect of diversity in object instances, we randomize the object models during training. For example, for the `sliced` single-predicate activity, the agent will encounter different types of fruit (e.g. peach, strawberry, pineapple, etc) during training. We train for 10K episodes, evaluate the final policy checkpoint and report the results in Table 3.

A.8 Potential to Transfer to Real-World

BEHAVIOR is a benchmark in simulation. This facilitates a continuous evaluation of solutions, fair and equal conditions, and increased accessibility without expensive robot hardware. It is also instrumental for modern robot learning procedures that require generating large amount of experiences. However, the use of simulation introduces a gap between the activities in our benchmark and the equivalent activities in real world. We argue that, while not negligible, we have taken measures

to close this gap with the goal of providing a benchmark where the performance of embodied AI solutions is close to the performance they would have in a real world system.

Our instantation of BEHAVIOR includes realistic scenes and object models, with high-quality visuals and close-to-real physical properties (mass, center of mass, friction) annotated in manual process assisted by the information obtained in the Internet. The underlying physics engine, pyBullet [59], is acknowledged as one of the standards in robotics and a high quality approximation of the underlying mechanical processes. The physics-based rendering from iGibson 2.0 generates high quality images to use as input in our evaluation. While our bimanual agent is not realistic, our second provided embodiment is a realistic robot model, a Fetch, with similar kinematics, actuation and sensing, facilitating the evaluation of solutions in BEHAVIOR that could act similarly on a real robot. Previous works have demonstrated good results developing solutions in iGibson 2.0 that could transfer to real world [2, 88], and evaluated the similarities between simulated and real-world sensor signals [22]. This indicates a high potential for the solutions evaluated in simulation in BEHAVIOR to perform similarly in the real world, a claim that we plan to evaluate experimentally after the pandemic.

A.9 Ethical Considerations

A.9.1 Ethics of the Goals of BEHAVIOR

This benchmark aims to facilitate advancement of autonomous robots. In the real world, robots that could do complex household activities like those in BEHAVIOR would have many significant social and ethical implications. Two core considerations are the labor impacts of automation and the physical safety of humans interacting with autonomous robots.

Labor and automation: Automation has a far-reaching impact on labor, especially automation of cognitive abilities like planning, goal/subgoal setting, and quality consideration—all of which are elements of this benchmark. Progress in AI will increasingly challenge the assumption that such abilities are exclusively human.

As a result, autonomous robots will cause significant job loss according to multiple labor and economics research groups. However, automation entering the labor force is inevitable and often positive in that it saves resources and facilitates otherwise unattainable growth. The core ethical consideration is in allocation of these resulting resources. If ownership of the ability to deploy cognitively capable robots and the resulting resources are given to the labor force impacted by this deployment, the harm can be turned into a benefit.

Furthermore, there may be potential to create managerial jobs in the space. The Federal Reserve of St. Louis has pointed out that managerial jobs have grown alongside automation. Secondly, in line with the Stanford Human-Centered AI Institute, we assert that all advancement in AI should work for humans and not replace them: human oversight and collaboration must be involved no matter how advanced autonomous robots get. Thirdly, the laborers whose jobs are being automated away are generally most experienced in them. Industries that use e.g. human-guided agents will require overseeing and collaboration jobs that laborers can be trained for, mitigating job loss.

Physical safety: We now consider the physical safety of humans interacting with autonomous robots. A major benefit of BEHAVIOR being in simulation is the opportunity for physically safe research, but an eventual goal is for an agent that succeeds on BEHAVIOR to end up powering a physical robot deployed in close proximity to humans. A robot can seriously physically harm a human if it cannot predict and avoid dangerous situations, but modeling human behavior is nowhere close to being solved.

Possible steps forward include physical safety as a hard constraint, increasing autonomous robots' ability to preempt dangerous situations, and careful deployment of autonomous robots that treats physical safety as paramount. BEHAVIOR can easily be extended to fail any agent that creates a predefined unsafe situation even if unrelated to the task at hand, e.g. leaving broken glass on the floor. Furthermore, we commend and encourage research in robot prediction of human behaviors and user-friendly human-robot interaction interfaces that help humans monitor safety.

A.9.2 Ethics of the Methods of BEHAVIOR

Demonstrator protection: BEHAVIOR includes virtual reality demonstrations performed by humans. After evaluation, the Stanford Institutional Review Board deemed this project exempt from review because we do not study the human data to draw conclusions about the demonstrators. Because VR can have adverse side effects regardless of the data use, we ensured that our VR activity satisfied the strictest guidelines for avoiding discomfort and all demonstrators were informed about the duration and content, risks of VR, and research purpose before consenting. All identities were kept confidential, and no identifiers (including likeness) exist in the dataset at all, nearly eliminating the risk of unforeseen identifiability that is sometimes found in computer vision research.

Dataset bias: We are aware of potential biases of the BEHAVIOR benchmark. The demonstrations and existing implementation in iGibson 2.0 are biased toward American homes that tend to be more expensive than average. We also base our activity selection on ATUS: while curated by experts, ATUS is ultimately focused on behavior of Americans. Both of these biases can be addressed by expanding the socioeconomic and cultural diversity of our sources. In the meantime, we emphasize transparency and clear communication of our sourcing and its focus/bias, and encourage users to take BEHAVIOR 's specific scope into account.

Accessibility: Our last point on method ethics is about BEHAVIOR's accessibility. Because BEHAVIOR, its assets, its dependencies, and at least one platform (iGibson 2.0) are all open-source or can be used for free, it is highly accessible. Engagement in this research requires only a computer to run on, making it as accessible as any other applied or experimental computer science research.

In conclusion, we believe BEHAVIOR's goals do not pose any unaddressed ethical risks.