HomeRobot: An Open Source Software Stack for Mobile Manipulation Research

Chris Paxton[1], **Austin Wang**[1], **Binit Shah**[2], **Blaine Matulevich**[2], **Dhruv Shah**[1], **Karmesh Yadav**[1,3], **Santhosh Ramakrishnan**[5], **Sriram Yenamandra**[3], **Yonatan Bisk**[1,4]
[1]FAIR, Meta AI
[2]Hello Robot
[3]Georgia Tech
[4]Carnegie Mellon
[5]UT Austin
cpaxton@fb.com

Abstract

Reproducibility in robotics research requires capable, shared hardware platforms which can be used for a wide variety of research. We have seen the power of these sorts of shared platforms in more general machine learning research (e.g., PyTorch), where there is a constant and open-sourced development over time to meet the needs of the community To be able to make rapid progress in robotics in the same way, we propose that we need: (1) shared real-world platforms which allow different teams to test and compare methods at low cost; (2) challenging simulations that reflect real-world environments and especially can drive perception and planning research; and (3) low-cost platforms with enough software to get started addressing all of these problems. To this end, we propose HomeRobot, a mobile manipulator software stack with associated benchmark in simulation, which is initially based on the low-cost, human-safe Hello Robot Stretch.[1]

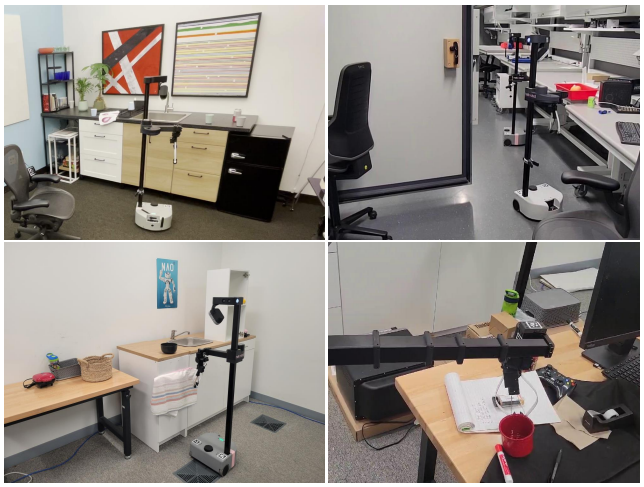

Figure 1: HomeRobot is a simple, easy-to-set-up library which works in multiple environments and requires only relatively affordable hardware. Computationally intensive operations are performed on a desktop PC with a GPU, and a dedicated consumer-grade router provides a network interface to a robot running low-level control and SLAM.

Intro

Home assistants have long been a motivating example in robotics, although relatively few teams have actually been able to deploy robots, especially mobile manipulators, in a wide range of homes. There are good reasons for this: deploying a useful mobile manipulator requires integrating a wide range of components, from grasping to object detection, task planning, and more. We propose HomeRobot, a software stack with both simulation and real-world hardware components, as a common, open-source platform for mobile manipulation research and development, in order to allow more users to perform high-impact research in home environments.

Common, shared platforms and software have spurred rapid progress in fields like language understanding (Wolf et al. 2019), image generation (von Platen et al. 2022), and are credited with the success of deep learning (Neubig et al. 2017; Al-Rfou et al. 2016; Abadi et al. 2015; Paszke et al. 2017). Robotics, unfortunately, lacks such shared platforms – the closest being the prevalent ROS (Quigley et al. 2009) software stack, often criticized for being complex and hard to use (Murali et al. 2019). Unlike PyRobot, we focus on providing a strong simulation component and implementing a variety of baselines.

We identify three goals for a platform for reproducible robotics research:

- **A motivating north star:** it must provide some guiding north-star tasks which can help shape and motivate researchers and allow for comparisons of a variety of methods on interesting, real-world problems;
- **Software Capability:** it should provide abstract interfaces that make a robot easier to use for a wide variety of tasks, including navigation and manipulation; and
- **Community:** we should incentive people to get involved, use the codebase, and build up a community around it.

We specifically propose a task called *open-vocabulary mobile manipulation (OVMM)* as our north star for this project. In OVMM, a robot must, without any prior maps, move around a new environment and find objects specified

[1]https://github.com/facebookresearch/home-robot

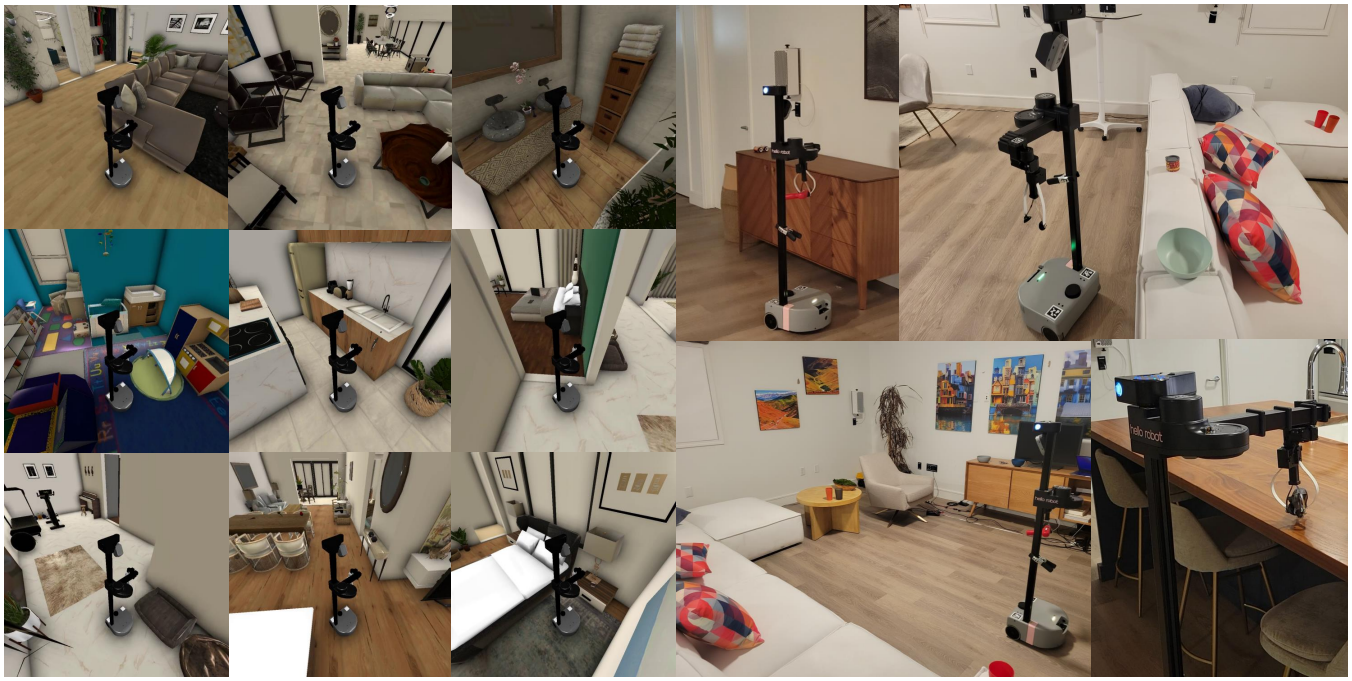

Figure 2: A low-cost home robot performing the HomeRobot OVMM task in both a simulated and a real-world environment. In HomeRobot, we provide both a challenging simulated "north star" task, wherein a mobile manipulator robot must find and grasp multiple seen and unseen objects, and a corresponding real-world robotics stack to allow others to reproduce this research and evaluation to produce useful home robot assistants.

by language. We believe this is a strong "building block" for future capabilities, and can be implemented in many different ways: using motion planning and simple heuristics or an LLM-based planner to determine goals, as per SayCan/SayPlan (Ahn et al. 2022; Rana et al. 2023), built on a task-and-motion-planning stack (Garrett et al. 2020; Curtis et al. 2022), reinforcement learning (Yenamandra et al. 2023b), or using implicit representations and pretrained models (Bolte et al. 2023). Having a common task which does not preclude any of these options allows us to share useful components across projects (e.g. detection, mapping, and grasping).

Our proposal and software stack are based around the Hello Robot Stretch, a low-cost mobile manipulation platform already in use at over 40 universities. By leveraging existing and accessible infrastructure increases the odds of adoption, code-sharing, and expediting research progress.

Projects using HomeRobot. Many research projects have already started using HomeRobot for object navigation (Ramakrishnan et al. 2022), exploration (Ramakrishnan, Al-Halah, and Grauman 2020), image-instance navigation (Krantz et al. 2023), continuous/lifelong learning (Powers, Gupta, and Paxton 2023), language-conditioned navigation (Bolte et al. 2023), and language-conditioned multi-task learning (Parashar et al. 2023).

Contributions. We describe the HomeRobot software stack: a codebase which allows for both simulated and real-world control of a Stretch robot from Hello Robots, a low-cost mobile manipulator with good manipulation and navigation capabilities (Kemp et al. 2022) that provides a strong basis for shared, mobile manipulation research in human home environments.

Open-Vocabulary Mobile Manipulation

Our HomeRobot code is released with modules for *Open Vocabulary Mobile Manipulation (OVMM)* (Yenamandra et al. 2023b), where a robot must find *any* object and place it in *any* location in an ordinary home. We chose this task because it represents a foundational capability for robots to be useful assistants: they must perceive a wide variety of objects, grasp and manipulate them, and understand large, complex scenes that may not be well mapped to begin with.

Formally, the HomeRobot OVMM task is set up as instructions of the form: "Move (`object`) from the (`start_receptacle`) to the (`goal_receptacle`)." The `object` is a small and manipulable household object (e.g., a cup, stuffed toy, or box). By contrast, `start_receptacle` and `goal_receptacle` are large pieces of furniture, which have surfaces upon which objects can be placed. The robot is placed in an unknown single-floor home environment - such as an apartment - and must, given the natural language names of `start_receptacle`, `object`, and `goal_receptacle`, pick up an `object` that is known to be on a `start_receptacle` and move it to any valid `goal_receptacle`. `start_receptacle` is always known to the robot, to help agents know where to look for the `object`.

The agent is successful if the specified `object` is indeed moved from a `start_receptacle` on which it be-

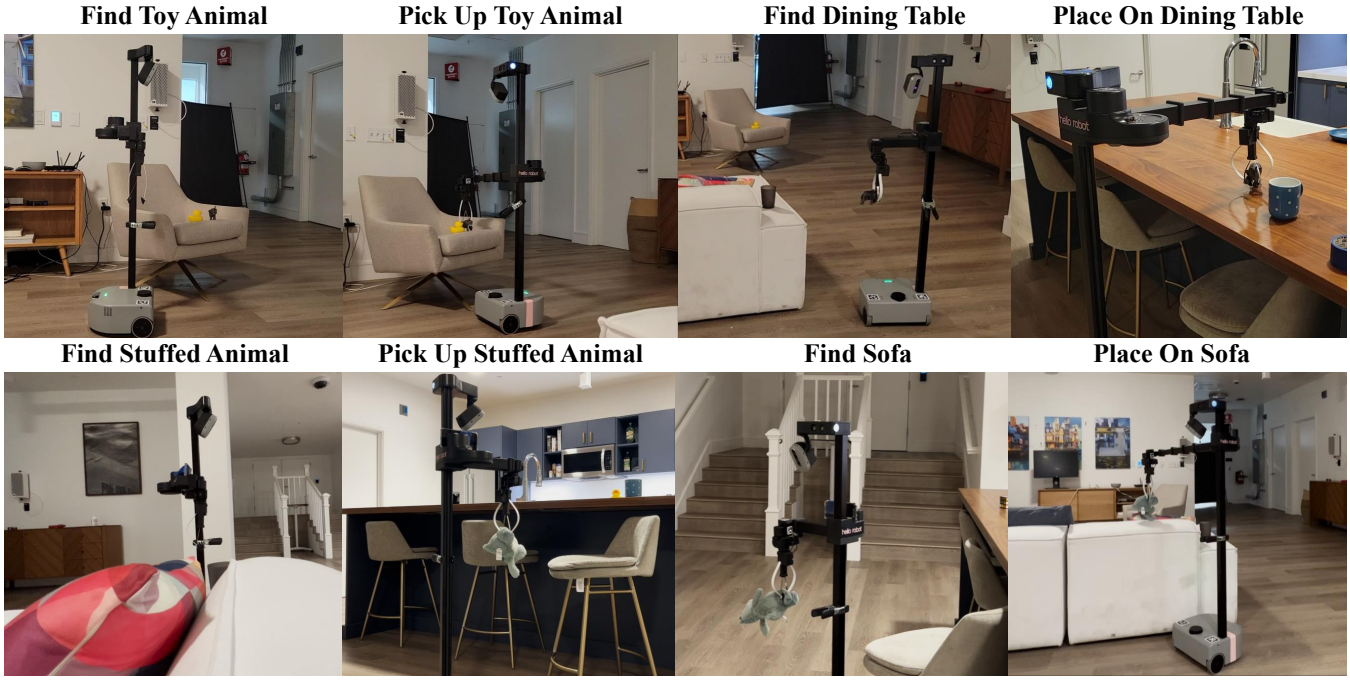

Figure 3: The HomeRobot stack being used to perform the Open Vocabulary Mobile Manipulation task in a held-out, real-world test environment. The test environment is a mock apartment which will be used for multiple versions of the benchmark, including for the HomeRobot Neurips 2023 competition (Yenamandra et al. 2023a). This allows us to perform both simulated and real-world benchmarking of new methods, and will help center the community around a broadly-available, low-cost platform with growing capabilities.

gan the episode, to any valid `goal_receptacle`. We give partial credit for each step the robot accomplishes: finding the `start_receptacle` with the `object`, picking up the `object`, finding the `goal_receptacle`, and placing the `object` on the `goal_receptacle`. There can be multiple valid objects that satisfy each query.

We implemented both a simulation and a real-world version of this HomeRobot OVMM task. In the real world, we use a controlled apartment environment with fixed lighting and furniture, within which we can reset a number of different objects drawn from both a seen (in training data) and unseen object set. Unseen objects used for testing are withheld and not released to the public.

Configuration and Setup

One challenge with low-cost mobile robots is how we can run GPU- and compute-intensive models to evaluate modern AI methods on them. The Stretch, like many similar robots, does not have onboard GPU, and will always have more limited compute than is available on a similar workstation. We this with a simple network configuration shown in Fig. 4. There are three components:

1. The **desktop** running code – in our case, the `eval_episode.py` script from HomeRobot – which connects to a remote mobile manipulator.
2. The dedicated **router** – an off-the-shelf consumer router, such as a Netgear Nighthawk router. This should ideally be dedicated for your robot and desktop setup to ensure good performance.
3. The mobile robot itself: a Hello Robot Stretch with DexWrist, as described above.

After the robot is configured, one only need run one script, a ROS launch file, as described in the HomeRobot startup instructions, which can be done over SSH. With a properly configured robot and router, you can visualize information on the desktop side, showing the robot's position, map from SLAM, and cameras. On the robot side, the only necessary command is starting the correct ROS launch file:

```
startup_stretch_hector_slam.launch
```

which brings up the robot controllers and starts running Hector SLAM to provide localization.

Checking network performance. We describe the visualization tools available briefly in the next section, but to check that the setup is working properly, you can start `rviz` and wave your hand in front of the robot – you should see minimal latency when waving a hand in front of the camera.

Timing between the robot and the remote workstation. We use ROS (Quigley et al. 2009) as our communications layer, and to implement low-level control on the robot. This also provides network communication. However, due to potential latency between the robot and desktop, we also need to make sure that observations are up to date.

We set up the robot to block after executing most navigation motions, in order to make this process simpler, until

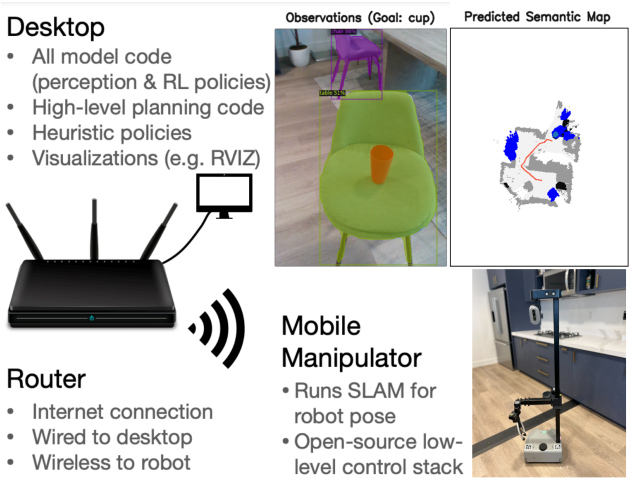

Figure 4: System overview for HomeRobot. We run visualizations and deep neural networks on a GPU-enabled workstation; the Hello Robots Stretch runs low-level controllers and mapping code. This allows us to use more compute than is strictly possible on the mobile robot for decision making and scene understanding.

there is an *up to date* image observation from the robot side. This means that timing between the robot and the workstation is extremely important: if we do not have up to date timing, we might have SLAM poses and depth measurements that do not match, which will lead to worse performance.

We solved this by having a clock on the robot side publish its time over ROS, and configure all systems to use this ROS master clock instead of system time. This prevents the user from having to worry about Linux time synchronization protocols like NTP when setting up the robot for the first time.

The HomeRobot Stack

HomeRobot is based on ROS (Quigley et al. 2009), with the goal being to provide abstractions which make it easy to learn in simulation, perform mobile manipulation experiments, and share code across labs and projects. We initially focus on the HomeRobot OVMM task, but also support multiple other types of research.

To support a wide range of machine learning reasearch projects, there are three different modules within the open-source HomeRobot library:

- `home_robot`: Shared components such as Environment interfaces, controllers, detection and segmentation.
- `home_robot_sim`: Simulation stack with Environments based on Habitat. We specifically provide environments modified from the Habitat Synthetic Scenes dataset (Yenamandra et al. 2023b; Khanna et al. 2023) as a starting point for development and testing.
- `home_robot_hw`: Hardware stack with server processes that runs on the robot, client API that runs on the GPU workstation, and Environments built using the client API.

Within HomeRobot, we also divide functionality between **Agents** and **Environments**, similar to how many reinforcement learning benchmarks are set up (Savva et al. 2019).

- **Agents** contain all of the necessary code to execute policies. We implement agents which use a mixture of heuristic policies and policies learned on our scene dataset via reinforcement learning.
- **Environments** provide **Observations** to the Agent, and a function which allows them to execute actions in the (real or simulated) environment.

Robot Control

We expand on the basic Stretch low-level control in order to make it easier to control the robot. In particular, we implemented a high-level motion planner, integrated different grasping and placement strategies, and also implemented reactive low-level control to make it easier to move the robot around in the real world for different methods.

State estimation In the real world, we use Hector SLAM (Kohlbrecher et al. 2014) to provide our robot's base pose as it moves around in a novel environment. This is as recommended in the Hello Robot documentation (Kemp et al. 2022). This is a reliable, LIDAR-based SLAM method, which in our experience works well in home-like environments such as that used in the HomeRobot OVMM real-world challenge.

However, Hector SLAM has a lot of high-frequency noise particularly when the robot is rotating. At the same time, we have access to wheel odometry signals, which provides accurate and low latency readings of displacement between time steps, but is subject to drift when integrated through time to estimate absolute position. We combine the signal coming from Hector SLAM with the pose signal from wheel odometry by effectively applying a discrete high-pass filter on the odometry signal and a discrete low-pass filter with the same cutoff frequency on the Hector SLAM signal, then blending the two signals together at a fixed rate:

$$x_t = x_{t-1} + (1-\lambda)\cdot(x_t^s - x_{t-1}) + \lambda\cdot(x_t^o - x_{t-1}^o)$$

with x as the output pose, x^s and x^o are pose estimates from Hector SLAM and wheel odometry respectively, and λ as a tuned coefficient. In our implementation, λ is computed from the cutoff frequency of 0.2 Hz, below which the pose estimate basically trusts SLAM for its unbiased absolute pose estimates, and relies on odometry signals for higher frequency, transient state estimations.

Base Velocity Controller The Hello Stretch software provides a native interface for controlling the linear and angular velocities of the differential-drive robot base. While we do expose an interface for users to control these velocities directly, it is desireable to have desired short-term goals as a more intuitive action space for policies, and to make them update-able at any instant to allow for replanning.

Thus, we implemented a velocity controller that produces continuous velocity commands that moves the robot to an input goal pose, show in Alg. 1. The controller operates in a heuristic manner: by rotating the robot so that it faces the

Algorithm 1: Base Velocity Controller Pseudocode

Require: Goal pose $p_g \in SE(2)$, Current pose $p \in SE(2)$
Compute pose error $e_p = p_g - p \in SE(2)$
Compute heading error θ_e = error between robot heading and direction of the goal location (direction of translational portion of e_p)
if translational part of $e_p < d_{threshold}$ **then**
    $v_l = 0$
    v_r : computed from the rotational part of e_p using a trapezoidal velocity profile
else
    v_r : computed from θ_e using a trapezoidal velocity profile
    v_l : computed from the translational part of e_p using a trapezoidal velocity profile
    Reduce linear velocity based on heading error $v_l = v_l^0 \sin(2\theta_e)$
    Compute $\dot{e}_l = f_l(v_l)$, the rate at which heading error increases due to linear velocity
    Compute $\dot{e}_r = f_r(v_r)$, the rate at which heading error decreases due to angular velocity
    if $\dot{e}_l < \frac{1}{2}\dot{e}_r$ **then**
        Limit linear velocity v_l so that $\dot{e}_l = \frac{1}{2}\dot{e}_r$
    end if
end if
Apply v_l, v_r

goal position at all times while moving towards the goal position, and then rotating to reach the goal orientation once goal position is reached.

Planning and Mapping

Our motion planner expands upon prior work (Gervet et al. 2022), which was shown to work in a wide range of human environments. We provide default values which tune it for navigating close to other objects and extended it to work in our continuous action space – challenging navigation aspects not present in the original paper. We implemented three components for the baseline verison of our system:

Semantic Mapping Module. The semantic map stores relevant objects, explored regions, and obstacles. To construct the map, we predict semantic categories and segmentation masks of objects from first-person observations. We use Detic (Zhou et al. 2022) for object detection and instance segmentation and backproject first-person semantic segmentation into a point cloud using the perceived depth, bin it into a 3D semantic voxel map, and finally sum over the height to compute a 2D semantic map.

We keep track of objects detected, obstacles, and explored areas in an explicit metric map of the environment from (Chaplot et al. 2020a). Concretely, it is a binary K x M x M matrix where M x M is the map size and K is the number of map channels. Each cell of this spatial map corresponds to 25 cm^2 (5 cm x 5 cm) in the physical world. Map channels $K = C + 4$ where C is the number of semantic object categories, and the remaining 4 channels represent the obstacles, the explored area, and the agent's current and past locations. An entry in the map is one if the cell contains an object of a particular semantic category, an obstacle, or is explored, and zero otherwise.

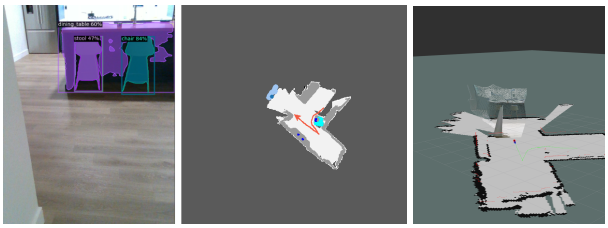

Figure 5: Exploring a real-world apartment during testing. The robot uses Detic (Zhou et al. 2022) to perceive the world and update a 2D map (center) which captures where it's seen relevant classes, and which obstacles exist; detections aren't always reliable, especially given a large and changing vocabulary of objects that we care about. In the HomeRobot stack, we provide a variety of tools for visualizing and implementing policies, including integration of RVIZ (right).

Frontier Exploration Policy. We explore the environment with a heuristic frontier-based exploration policy (Yamauchi 1997). This heuristic selects as the goal the point closest to the robot in geodesic distance within the boundary between the explored and unexplored region of the map. We further implemented and tested more advanced exploration policies in HomeRobot, and verified that they work on the real-world robot in a variety of environments (Ramakrishnan, Al-Halah, and Grauman 2020; Ramakrishnan et al. 2022).

Navigation Planner. Given a long-term goal output by the frontier exploration policy, we use the Fast Marching Method (Sethian 1999) as in (Chaplot et al. 2020a) to plan a path and the first low-level action along this path deterministically. Although the semantic exploration policy acts at a coarse time scale, the planner acts at a fine time scale: every step we update the map and replan the path to the long-term goal. The robot attempts to plan to goals if they have been seen; if it cannot get within a certain distance of the goal objects, then it will instead plan to a point on the frontier.

Navigating to objects on `start_receptacle`. Since small objects (*e.g.* `action_figure`, `apple`) can be hard to locate from a distance, we leverage the typically larger `start_receptacle` goals for finding objects. We make the following changes to the original planning policy (Chaplot et al. 2020b):

1. If object and `start_receptacle` co-occur in at least one cell of the semantic map, plan to reach the object
2. If the object is not found but `start_receptacle` appears in the semantic map after excluding the regions within 1m of the agent's past locations, plan to reach the `start_receptacle`
3. Otherwise, plan to reach the closest frontier

In step 2, we exclude the regions that the agent has been close to, to prevent it from re-visiting already visited instances of `start_receptacle`.

Visualization Tools

We use RVIZ, a part of ROS, to visualize results and progress. Fig. 5 shows three different outputs from our sys-

Name	Mobile	Human Sized	Human Safe	Commercially Available	Manipulation DOF	Approximate Cost
Boston Dynamics Spot	✓			✓	7	$200,000
Franka Emika Panda			✓	✓	7	$30,000
Locobot	✓		✓		5	$5,000
Fetch	✓	✓	✓		7	$100,000
Hello Robot Stretch	✓	✓	✓	✓	4	$19,000
Stretch with DexWrist	✓	✓	✓	✓	6	$25,000

Table 1: Notes on platform selection. We chose the **Stretch with DexWrist** as a good compromise between manipulation, navigation, and cost, while being human-safe and approximately human-sized.

tem: on the far left, an image from the test environment being processed by Detic; in the center, a top-down map generated by the navigation planner described in Sec. ; and on the right, an image from RVIZ with the point cloud from the robot's head camera registered against the 2D lidar map created by Hector SLAM.

One advantage of the HomeRobot stack is that it is designed to work with existing debugging tools - especially ROS (Quigley et al. 2009). ROS is a widely-used framework for robotics software development which comes with a lot of online resources, official support from Hello Robot, and a rich and thriving open-source community with wide industry backing.

The HomeRobot Community

Our goal with HomeRobot is to build a community which allows researchers to attempt different research projects, and compare them across a variety of different environments on the same real-world robot platform. To this end, we've taken a three-pronged approach:

- Releasing an open-source software stack with both a simulation and a real-world component, containing powerful baselines for a variety of robot tasks;
- Implementing a number of different novel research projects using HomeRobot; and
- Proposing a competition (Yenamandra et al. 2023a) which will reward the best solutions, which can then be integrated into our open-source stack and compared against other teams in a controlled environment.

Choice of Hardware

When deciding on a common platform for research, it's important that we choose a robot which has a wide range of capabilities, some of which allow for research projects that have not yet been attempted. We decided to focus on a specific model of the Hello Robot Stretch: it is relatively low cost, human safe, light weight, and a capable 6dof manipulator that has been used in numerous research projects (Gervet et al. 2022; Yenamandra et al. 2023b; Parashar et al. 2023; Haldar et al. 2023; Bahl, Gupta, and Pathak 2022).

We describe some options for commercially-available robotics hardware in Tab. 1. While the Franka Emika Panda is not a mobile robot, we include it here because it's a very commmonly used platform in both industrial research labs and at universities, making its price a fair comparison point for what is reasonable.

The Competition

We also proposed a Neurips 2023 competition in order to incentive people to begin using our software stack (Yenamandra et al. 2023a). This competition is centered around the HomeRobot OVMM task: picking up an object from a `start_receptacle` and moving it to a `goal_receptacle` somewhere else in the world, all initially without any maps.

There are two phases in this competition: a simulation phase and a real-world phase. Our goal with this two-phase structure is that it allows us to run a set of experiments on held-out test scenes, and then identify only the best few variants of our method to re-run in the real world. This allows us to run more controlled experiments for methods implemented by different teams.

At present 20 universities have begun participating both providing a testament to the community interest and providing us valuable feedback on further refining the codebase and ensuring generality. Moving forward, we will be advocating for pull requests from the community to help consolidate research efforts and insights for all to benefit.

Discussion and Conclusions

As noted at the start, within robotics we have watched the runaway success of shared frameworks with transferable and reproducible code across machine learning. While a strength of robotics is creating new hardware, custom end effectors, and beyond, there is now a critical mass of interested researchers focusing on indoor mobile manipulation and this is coupled with a low-cost commercial platform. This puts us in a unique position to begin sharing code and models to dramatically speed up the pace of research.

Our aim with HomeRobot is to provide the community a common platform from which to build and share knowledge. We focused on a widely-available, low-cost platform which can perform 6dof pick-and-place object manipulation, and implemented a modular software architecture allowing both simulation and real-world experimentation. In addition, we describe the "north star" goal of sharing solutions to Open-Vocabulary Mobile Manipulation, which we consider to be a cornerstone of robotics manipulation in homes due to its ubiquity, and to the fact that existing methods often perform quite poorly in the real world or even on realistic simulation settings (Yenamandra et al. 2023b).

The HomeRobot code and example videos of the simulation and real-world benchmarks are online at ovmm.github.io and https://github.com/facebookresearch/home-robot.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/. Software available from tensorflow.org.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Al-Rfou, R.; Alain, G.; Almahairi, A.; Angermueller, C.; Bahdanau, D.; Ballas, N.; Bastien, F.; Bayer, J.; Belikov, A.; Belopolsky, A.; Bengio, Y.; Bergeron, A.; Bergstra, J.; Bisson, V.; Snyder, J. B.; Bouchard, N.; Boulanger-Lewandowski, N.; Bouthillier, X.; de Brébisson, A.; Breuleux, O.; Carrier, P.-L.; Cho, K.; Chorowski, J.; Christiano, P.; Cooijmans, T.; Côté, M.-A.; Côté, M.; Courville, A.; Dauphin, Y. N.; Delalleau, O.; Demouth, J.; Desjardins, G.; Dieleman, S.; Dinh, L.; Ducoffe, M.; Dumoulin, V.; Kahou, S. E.; Erhan, D.; Fan, Z.; Firat, O.; Germain, M.; Glorot, X.; Goodfellow, I.; Graham, M.; Gulcehre, C.; Hamel, P.; Harlouchet, I.; Heng, J.-P.; Hidasi, B.; Honari, S.; Jain, A.; Jean, S.; Jia, K.; Korobov, M.; Kulkarni, V.; Lamb, A.; Lamblin, P.; Larsen, E.; Laurent, C.; Lee, S.; Lefrancois, S.; Lemieux, S.; Léonard, N.; Lin, Z.; Livezey, J. A.; Lorenz, C.; Lowin, J.; Ma, Q.; Manzagol, P.-A.; Mastropietro, O.; McGibbon, R. T.; Memisevic, R.; van Merriënboer, B.; Michalski, V.; Mirza, M.; Orlandi, A.; Pal, C.; Pascanu, R.; Pezeshki, M.; Raffel, C.; Renshaw, D.; Rocklin, M.; Romero, A.; Roth, M.; Sadowski, P.; Salvatier, J.; Savard, F.; Schlüter, J.; Schulman, J.; Schwartz, G.; Serban, I. V.; Serdyuk, D.; Shabanian, S.; Étienne Simon; Spieckermann, S.; Subramanyam, S. R.; Sygnowski, J.; Tanguay, J.; van Tulder, G.; Turian, J.; Urban, S.; Vincent, P.; Visin, F.; de Vries, H.; Warde-Farley, D.; Webb, D. J.; Willson, M.; Xu, K.; Xue, L.; Yao, L.; Zhang, S.; and Zhang, Y. 2016. Theano: A Python framework for fast computation of mathematical expressions. *ArXiv 1605.02688*.

Bahl, S.; Gupta, A.; and Pathak, D. 2022. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*.

Bolte, B.; Wang, A.; Yang, J.; Mukadam, M.; Kalakrishnan, M.; and Paxton, C. 2023. USA-Net: Unified Semantic and Affordance Representations for Robot Memory. *arXiv preprint arXiv:2304.12164*.

Chaplot, D. S.; Gandhi, D. P.; Gupta, A.; and Salakhutdinov, R. R. 2020a. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*.

Chaplot, D. S.; Gupta, S.; Gupta, A.; and Salakhutdinov, R. 2020b. Learning To Explore Using Active Neural Mapping. *ICLR*.

Curtis, A.; Fang, X.; Kaelbling, L. P.; Lozano-Pérez, T.; and Garrett, C. R. 2022. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)*, 1940–1946. IEEE.

Garrett, C. R.; Paxton, C.; Lozano-Pérez, T.; Kaelbling, L. P.; and Fox, D. 2020. Online replanning in belief space for partially observable task and motion problems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 5678–5684. IEEE.

Gervet, T.; Chintala, S.; Batra, D.; Malik, J.; and Chaplot, D. S. 2022. Navigating to Objects in the Real World. *arXiv*.

Haldar, S.; Pari, J.; Rai, A.; and Pinto, L. 2023. Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations. *arXiv preprint arXiv:2303.01497*.

Kemp, C. C.; Edsinger, A.; Clever, H. M.; and Matulevich, B. 2022. The design of stretch: A compact, lightweight mobile manipulator for indoor human environments. In *2022 International Conference on Robotics and Automation (ICRA)*, 3150–3157. IEEE.

Khanna, M.; Mao, Y.; Jiang, H.; Haresh, S.; Schacklett, B.; Batra, D.; Clegg, A.; Undersander, E.; Chang, A. X.; and Savva, M. 2023. Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation. *arXiv preprint arXiv:2306.11290*.

Kohlbrecher, S.; Meyer, J.; Graber, T.; Petersen, K.; Klingauf, U.; and Von Stryk, O. 2014. Hector open source modules for autonomous mapping and navigation with rescue robots. In *RoboCup 2013: Robot World Cup XVII 17*, 624–631. Springer.

Krantz, J.; Gervet, T.; Yadav, K.; Wang, A.; Paxton, C.; Mottaghi, R.; Batra, D.; Malik, J.; Lee, S.; and Chaplot, D. S. 2023. Navigating to Objects Specified by Images. *arXiv preprint arXiv:2304.01192*.

Murali, A.; Chen, T.; Alwala, K. V.; Gandhi, D.; Pinto, L.; Gupta, S.; and Gupta, A. 2019. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*.

Neubig, G.; Dyer, C.; Goldberg, Y.; Matthews, A.; Ammar, W.; Anastasopoulos, A.; Ballesteros, M.; Chiang, D.; Clothiaux, D.; Cohn, T.; Duh, K.; Faruqui, M.; Gan, C.; Garrette, D.; Ji, Y.; Kong, L.; Kuncoro, A.; Kumar, G.; Malaviya, C.; Michel, P.; Oda, Y.; Richardson, M.; Saphra, N.; Swayamdipta, S.; and Yin, P. 2017. DyNet: The Dynamic Neural Network Toolkit. *ArXiv*.

Parashar, P.; Vakil, J.; Powers, S.; and Paxton, C. 2023. Spatial-Language Attention Policies for Efficient Robot Learning. *arXiv preprint arXiv:2304.11235*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS 2017 Workshop Autodiff*.

Powers, S.; Gupta, A.; and Paxton, C. 2023. Evaluating Continual Learning on a Home Robot. *arXiv preprint arXiv:2306.02413*.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A. Y.; et al. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 5. Kobe, Japan.

Ramakrishnan, S. K.; Al-Halah, Z.; and Grauman, K. 2020. Occupancy Anticipation for Efficient Exploration and Navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ramakrishnan, S. K.; Chaplot, D. S.; Al-Halah, Z.; Malik, J.; and Grauman, K. 2022. PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning. In *Computer Vision and Pattern Recognition (CVPR), 2022 IEEE Conference on*. IEEE.

Rana, K.; Haviland, J.; Garg, S.; Abou-Chakra, J.; Reid, I.; and Suenderhauf, N. 2023. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Task Planning. *arXiv preprint arXiv:2307.06135*.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. *ICCV*.

Sethian, J. A. 1999. Fast marching methods. *SIAM review*.

von Platen, P.; Patil, S.; Lozhkov, A.; Cuenca, P.; Lambert, N.; Rasul, K.; Davaadorj, M.; and Wolf, T. 2022. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers. Accessed: 2024-01-01.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*.

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*.

Yenamandra, S.; Ramachandran, A.; Khanna, M.; Yadav, K.; Chaplot, D. S.; Chhablani, G.; Clegg, A.; Gervet, T.; Jain, V.; Partsey, R.; Ramrakhya, R.; Szot, A.; Yang, T.-Y.; Edsinger, A.; Kemp, C.; Shah, B.; Kira, Z.; Batra, D.; Mottaghi, R.; Bisk, Y.; and Paxton, C. 2023a. HomeRobot Open Vocab Mobile Manipulation Challenge. In *Thirty-seventh Conference on Neural Information Processing Systems: Competition Track*.

Yenamandra, S.; Ramachandran, A.; Yadav, K.; Wang, A.; Khanna, M.; Gervet, T.; Yang, T.-Y.; Jain, V.; Clegg, A. W.; Turner, J.; Kira, Z.; Savva, M.; Chang, A.; Chaplot, D. S.; Batra, D.; Mottaghi, R.; Bisk, Y.; and Paxton, C. 2023b. HomeRobot: Open-Vocabulary Mobile Manipulation. In *Conference on Robot Learning*.

Zhou, X.; Girdhar, R.; Joulin, A.; Krähenbühl, P.; and Misra, I. 2022. Detecting Twenty-thousand Classes using Image-level Supervision. In *ECCV*.