This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

# *SPIN:* Simultaneous Perception, Interaction and Navigation

Shagun Uppal Ananye Agarwal Haoyu Xiong Kenneth Shaw Deepak Pathak
Carnegie Mellon University

Figure 1. **Learning to SPIN:** Our robot learns to simultaneously perceive, manipulate, and navigate cluttered unstructured environments in a whole-body fashion. The robot has an actuated camera with a limited field of view that it must control to get information about its environment. The motion and perception problem are tightly coupled since what the robot knows about the environment influences how it can move and vice versa. We show results in a large variety of scenarios both indoors and outdoors with different obstacles like boxes and furniture. Our robot can pick up different objects like cups, and utensils. Video demos at https://spin-robot.github.io

## Abstract

*While there has been remarkable progress recently in the fields of manipulation and locomotion, mobile manipulation remains a long-standing challenge. Compared to locomotion or static manipulation, a mobile system must make a diverse range of long-horizon tasks feasible in unstructured and dynamic environments. While the applications are broad and interesting, there are a plethora of challenges in developing these systems such as coordination between the base and arm, reliance on onboard perception for perceiving and interacting with the environment, and most importantly, simultaneously integrating all these parts together. Prior works approach the problem using disentangled modular skills for mobility and manipulation that are trivially tied together. This causes several limitations such as compounding errors, delays in decision-making, and no whole-body coordination. In this work, we present a reactive mobile manipulation framework that uses an active visual system to consciously perceive and react to its environment. Similar to how humans leverage whole-body and hand-eye coordination, we develop a mobile manipulator that exploits its ability to move and see, more specifically – to move in order to see and to see in order to move. This allows it to not only move around and interact with its environment but also, choose "when" to perceive "what" using an active visual system. We observe that such an agent learns to navigate around complex cluttered scenarios while displaying agile whole-body coordination using only ego-vision without needing to create environment maps. Videos are available at https://spin-robot.github.io*

## 1. Introduction

Consider the example shown in Figure 2. A person is trying to carry a coffee cup through clutter. This not only requires navigational planning from start to goal but planning of the whole body to avoid obstacles along the way. Furthermore, due to ego-centric vision, the person needs to actively look around to gather the presence of obstacles. This general form of mobile manipulation task necessitates a coupled understanding of whole-body control with active perception. This capability is one of the fundamental and frequently encountered tasks in embodied cognition.

The dominant paradigm to tackle this problem is through classical planning-based control which requires apriori knowledge about the precise location of all the obstacles along with a detailed map of the environment. In most real-world scenarios, this assumption is impractical due to computational reasons, but more importantly, because environments are dynamic and objects keep moving around in general. Furthermore, relying on precise measurement of scenes for control does not allow agents to reactively improvise to changes in their environment. Practically, even when the complete environment map is known apriori, joint planning for a system with high degrees of freedom, say a mobile base with an arm, is often intractable and too expensive to be deployed in real-time.

Humans, on the other hand, do not rely on precise known estimates of object locations and instead use ego-centric vision to navigate around obstacles in real-time. In an unfamiliar environment, where to look is informed by where they want to move (called 'active perception'), and how they move in return determines what all they can see immediately afterward. This integrated mobility and perception allows us to see, adapt, and react to maneuver through unseen heavily cluttered environments.

This paper presents ***SPIN***, an end-to-end approach to ***S***imultaneous ***P***erception, ***I***nteraction, and ***N***avigation. We train a single model that not only outputs low-level controls for the robot body and arm but also predicts where should the robot's ego-centric camera look at each time step while moving its whole body by avoiding obstacles. We train our approach via reinforcement learning (RL), and to get around the computational bottleneck of rendering depth images, we use a teacher-student training framework where robot behavior is first learned using RL with access to visible object scandots and then distilled into a policy that operates from ego-depth using supervised learning. We evaluate across 6 benchmarks in simulation ranging from easy, medium, and hard difficulty, and two real-world environments with a similar level of clutter as the hard environments in simulation and also add dynamic, adversarial obstacles. We find that our method outperforms classical methods and baselines which do not use active vision. We also observe emergent behaviors, including dynamic obstacle avoidance which the robot did not see during training time.

Figure 2. Human and robot illustration of whole-body navigation through the clutter.

Our approach presents a radical hypothesis that the traditionally non-reactive planning approach to whole-body control can indeed be cast into a reactive model – i.e. – single end-to-end policy trained by RL. Despite a big departure from optimal control literature, this hypothesis is not as surprising since agile whole-body coordination and fast obstacle avoidance in humans are developed into muscle memory over time. We now discuss our approach in detail.

## 2. Method

We want our mobile manipulator (Fig. 5) to navigate and manipulate objects while avoiding obstacles in cluttered environments. It shares anatomical similarities with a human, bringing with it many of the same challenges. First, it has a limb in the form of an arm that can be raised and lowered, so the robot must constantly move the arm to avoid any obstacles. Second, it has an actuated camera with a very limited field of view ($87°$ horizontal, $58°$ vertical), so it needs to constantly look around to simultaneously plan ahead and look out for unexpected obstacles. Imagine yourself walking through a cluttered cabinet, there are too many obstacles around to keep track of, and you can't see all of them at once, so you must keep looking all around your body to plan a path through the clutter but also make sure you don't hit anything you missed along the way. Unlike regular walking where our eyes mostly point straight ahead and the path is clear, here you must *actively* choose what to perceive for simultaneously planning ahead and also doing reactive fixes to your planned path. Since all the obstacles cannot be perceived at a single glance, you must have spatial awareness and know where the obstacle you saw some time ago is right now in relation to your body. Note that this entire process is very different from the classical approach, where perception, planning, and obstacle avoidance are separate processes executed separately and in sequence. Further, it is assumed that the output of each is perfect, whereas this is rarely the case in practice in our unstructured world.

To deal with this challenging, entangled problem setup,

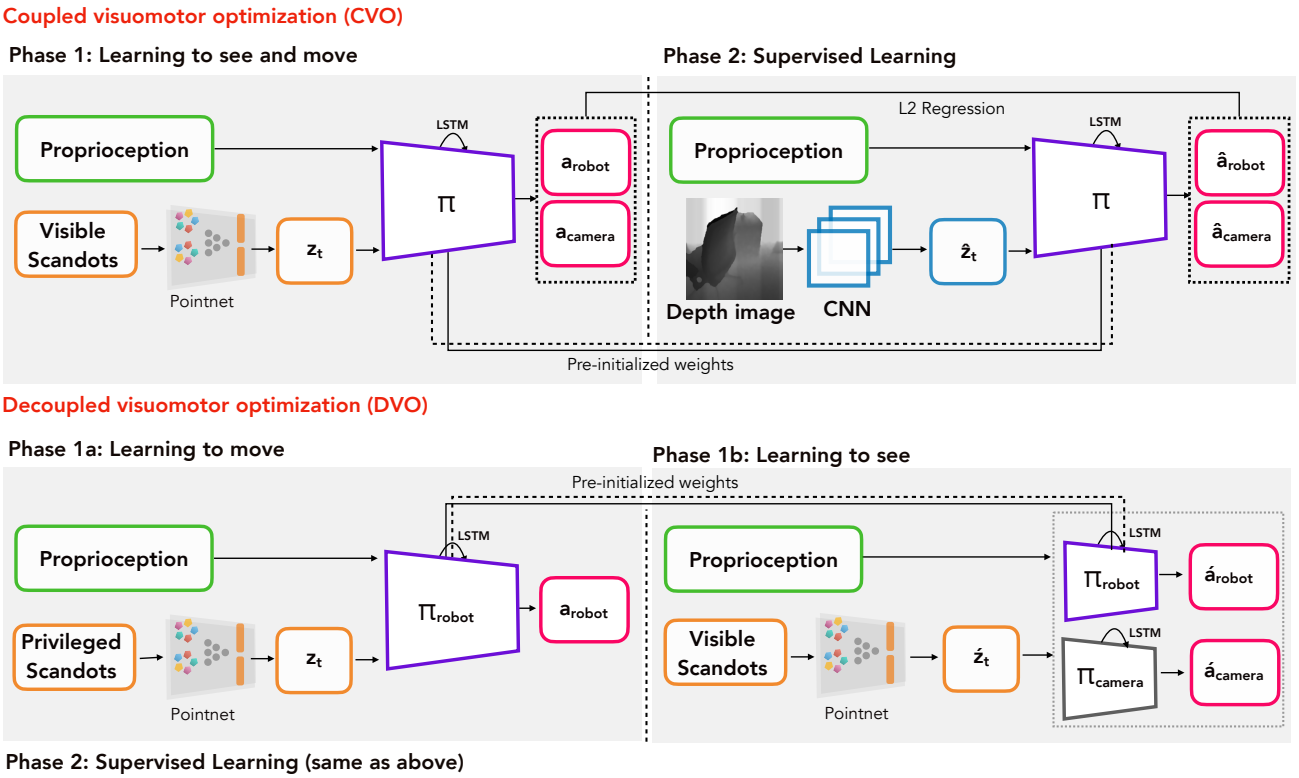



Figure 3. We learn a policy that uses ego-vision to simultaneously perceive, interact, and navigate in cluttered environments. We propose two methods: (1) Coupled Visuomotor Optimization (CVO) learns robot and camera actions at the same time. We train an RL policy to predict these. We only provide scandots if they are visible in the agent's field-of-view allowing the agent to learn to move its camera and aggregate information about its environment. This is followed by a phase-2 supervised training where this behavior is distilled into a student network that operates with ego-centric depth images (2) Decoupled Visuomotor Optimization (DVO) decouples the action and perception learning into two parts: first the agent learns to navigate across clutter assuming access to all obstacles. In phase 1b, the robot learns to move its camera to estimate the relevant information. This is followed by supervised learning same as above.

we take a data-driven approach. We train our robot to navigate inside procedurally generated clutter in simulation using RL. The robot is only allowed to perceive the part of its environment that is visible to the camera and learns to coordinate its arm, base, and camera motion such that it can plan ahead and reactively adjust to obstacles.

In practice, since training with RL requires many samples and rendering depth is inefficient (see supp. Section **??**), we divide training into two phases. In the first one, we learn mobile manipulation behaviors via RL using a cheap-to-compute variant of depth and in phase 2 we train a CNN for perception from depth images as illustrated in Figure 3.

## 2.1. Phase 1 - Learning Simultaneous Perception, Interaction and Navigation

In this stage, we use RL to learn to control all the joints of the robot to navigate clutter and pick target objects. Since rendering depth images directly from the robot camera is expensive, we must instead use an ersatz version that contains the same information and is cheap to compute. We do so using *scandots* $\mathbf{s}_t$ which are the xyz coordinates of the bounding box of each obstacle. To specify which object to pick, we give the initial location of the object (before it is touched by the robot) $\mathbf{o}_i$. In lieu of the object image we give the current location of the object $\mathbf{o}_t$. Here, scandots $\mathbf{s}_t$ and object location $\mathbf{o}_t$ are privileged information which must later be estimated from depth images. Given this information, we train two separate LSTM policies $\pi_{\text{nav}}$ and $\pi_{\text{pick}}$. At test time, the nav policy is activated to reach a target location and we switch to the pick one once the robot gets close to the object.

### 2.1.1 Pick Policy

This accesses proprioception $\mathbf{x}_t$ consisting of robot joint angles and velocities $\mathbf{q}_t, \dot{\mathbf{q}}_t$, base linear and angular velocity $v_t, \omega_t$. For perception, it gets object's initial and current location $\mathbf{o}_i$, $\mathbf{o}_t$ and predicts robot and camera actions.

$$[\mathbf{a}_{\text{robot}}, \mathbf{a}_{\text{cam}}] = \pi_{\text{pick}}(\mathbf{x}_t, F(\mathbf{o}_t, \mathbf{x}_t), \mathbf{o}_i) \quad (1)$$

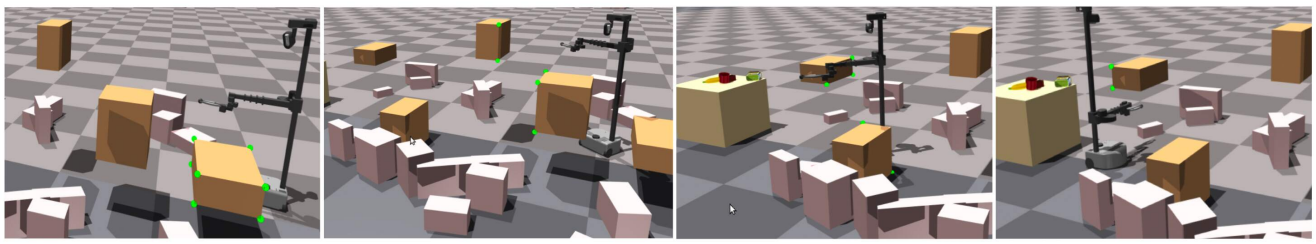


Figure 4. We illustrate one scenario of the simulation benchmark here with many obstacles in a narrow passage. The agent learns to develop whole-body coordination such as the robot's arm movement in the last two frames, in order to reactively adapt and navigate through such cluttered scenes by actively moving around the camera and aggregating information for efficient navigation without collisions.

where $F$ is a masking function that masks object position $\mathbf{o}_t$ if it is not in the field of view of the camera. This is required since object position can only be estimated from depth in phase 2 if it is visible.

#### 2.1.2 Navigation Policy

Training this policy requires a complex joint visuomotor optimization since robot motion is dependent on its knowledge of the environment which in turn depends on how the robot moves. We present two approaches to tackle this problem.

**Coupled Visuomotor Optimization (CVO)** Here, we set up a partially observable environment for the robot and let the RL algorithm do the joint optimization using large-scale data. In particular, the policy gets proprioception $\mathbf{x}_t$ and only visible scandots $\tilde{\mathbf{s}}_t = F(\mathbf{s}_t, \mathbf{x}_t)$ as observation and has to predict both the camera and the robot actions. Since the scandots are permutation invariant, we pass them through a trainable point-net architecture $P$ to obtain compressed latent $\mathbf{z}_t = P(\tilde{\mathbf{s}}_t)$ that we pass to the policy

$$[\mathbf{a}_{\text{robot}}, \mathbf{a}_{\text{cam}}] = \pi_{\text{nav}}(\mathbf{x}_t, \mathbf{z}_t) \tag{2}$$

This presents a tough optimization landscape because the observations at each step are strongly dependent on $\mathbf{a}_{\text{cam}}$. For instance, if the camera swivels around the observations at the next timestep may look completely different. Indeed, we observe that this requires billions of samples inside a GPU-accelerated simulator to optimize which may not always be feasible in practice.

**Decoupled Visuomotor Optimization (DVO)** To ease the optimization process, we learn the robot and camera actions separately. First, we learn how to move by giving the robot access to all available scandots $\mathbf{z}_t = P(\mathbf{s}_t)$ in a local vicinity. Since the robot sees everything, the camera motion is irrelevant and we just predict the robot motion

$$\mathbf{a}_{\text{robot}} = \pi^{\text{1a}}_{\text{nav}}(\mathbf{x}_t, \mathbf{z}_t, \mathbf{g}_t) \tag{3}$$

where $\mathbf{g}_t$ is the goal with respect to the base. Using this policy as supervision, we train another policy to predict both camera and robot motions with access to only visible scandots $\hat{\mathbf{z}}_t = P(F(\mathbf{s}_t, \mathbf{x}_t))$. This policy is trained via RL to predict the robot actions from phase 1 policy $\mathbf{a}_{\text{robot}}$. This optimization forces the student policy to learn camera behaviors that capture information about the environment that are needed to move in the optimal fashion. We initialize $\pi^{\text{1b}}_{\text{nav}}$ from the weights of $\pi^{\text{1a}}_{\text{nav}}$

$$\begin{aligned} \min_{\pi^{\text{1b}}_{\text{nav}}} \quad & \|\hat{\mathbf{a}}_{\text{robot}} - \mathbf{a}_{\text{robot}}\| \\ \text{s.t.} \quad & [\hat{\mathbf{a}}_{\text{robot}}, \hat{\mathbf{a}}_{\text{cam}}] = \pi^{\text{1b}}_{\text{nav}}(\mathbf{x}_t, \hat{\mathbf{z}}_t, \mathbf{g}_t) \end{aligned} \tag{4}$$

This decoupled approach learns to move and see in separate phases which eases the optimization burden. In principle, the coupled optimization is better since it is possible that the 1a policy may learn to exploit privileged information in a way that the 1b policy cannot estimate it for any set of camera movements. However, in our setting, this did not turn out to be the case.

We train using PPO [32] with backpropagation through time [37] in procedurally generated environments.

**Rewards:** For the navigation task, we use distance to goal reward $\|\mathbf{g}_t\|$ along with a forward progress reward $|(\mathbf{v}_t)_g|$ where $(\mathbf{v}_t)_g$ is velocity along the direction of the goal.

$$r_{\text{nav}} = 0.1 \cdot \|\mathbf{g}_t\| + 0.1 \cdot |(\mathbf{v}_t)_g| \tag{5}$$

For the pick task, we provide an object reaching reward, i.e., the distance between the gripper and object. This is followed by a lift reward if a successful grasp is detected (based on whether contact forces cross a threshold).

$$r_{\text{pick}} = 0.5 \cdot \|\mathbf{o}_t - \mathbf{p}_t\| + 0.5 \cdot r_{\text{lift}} \tag{6}$$

where

$$r_{\text{lift}} = \left(1 - \tanh\left(15 \cdot [(\mathbf{o}_t)_z]_+\right)\right) \mathbb{I}\left[\sum_i f_i > 10\right] \tag{7}$$

where $[x]_+ = \max(x, 0)$ and $\mathbb{I}$ is the indicator function which forces the reward to be active only when object contact forces $f_i$ exceed 10N.

**Training environments:** We procedurally generate long corridors with obstacles placed in between the robot and the goal. The initial joints and orientation of the robot are randomized. Near the edges of the corridors, we place randomized obstacles and walls to simulate distractors in the depth image. For the pick task, objects are spawned on tables of varying dimensions. We use five different objects - banana, mug, can, foambrick and a bottle. The episode is terminated if the robot reaches the goal or hits an obstacle/table.

### 2.2. Phase 2 - From Scandots to Depth

Scandots are not directly observable in the real world and must instead be estimated from the depth image. We train a convolution network $C$ to convert rendered depth images $\mathbf{d}_t$ to perception latents $\tilde{\mathbf{z}}_t$. This latent is passed to a student policy $\pi'$ to predict the actions $[\tilde{\mathbf{a}}_{\text{robot}}, \tilde{\mathbf{a}}_{\text{cam}}]$. This is supervised using L2 loss from the phase 1 actions. The weights for $\pi'$ are initialized using $\pi$. We train this policy using DAgger [29]. For the navigation policy we optimize

$$\min_{C_{\text{nav}}, \pi'_{\text{nav}}} \|\pi'_{\text{nav}}(C_{\text{nav}}(\mathbf{d}_t), \mathbf{x}_t, \mathbf{g}_t) - \pi_{\text{nav}}(\mathbf{z}_t, \mathbf{x}_t, \mathbf{g}_t)\| \quad (8)$$

Note that the teacher policy $\pi_{\text{nav}}$ can be trained using either the coupled or decoupled approach. Similarly, for the pick policy we estimate current object position $\mathbf{o}_t$ from depth

$$\min_{C_{\text{pick}}, \pi'_{\text{pick}}} \left\|\pi'_{\text{pick}}(C_{\text{pick}}(\mathbf{d}_t), \mathbf{x}_t, \mathbf{o}_i) - \pi_{\text{pick}}(\mathbf{z}_t, \mathbf{x}_t, \mathbf{o}_t, \mathbf{o}_i)\right\| \quad (9)$$

## 3. Experimental Setup

We use the Hello Robot Stretch [1] for all our experiments (Fig. 5). The robot has 10 actuated joints which include 2 degrees of freedom for the camera, 2 for base rotation and translation, 2 for the arm, 1 for the gripper fingers and 3 for the dexterous wrist. An Intel D435i depth camera is mounted on the top of the robot head which is actuated using two motors. The learned policy operates at 10Hz and we do velocity control for the robot base and position control for all the other joints. Velocity control for the robot base allows us to perform simultaneous robot translation and rotation for more agile behavior. We train using IsaacGymEnvs [26] using 8192 environments which takes 6 hours of training for phase 1 and 10 hours of training time for phase 2 on a RTX 3090. We compare against the following baselines:

- **FixCam:** The camera joints are frozen and the camera is forced to look forward. This baseline shows whether active vision is useful for the mobile manipulation problem and a fixed viewpoint is not enough.
- **Mapping:** Instead of using a moving depth camera to get a series of frames this baseline assumes exteroception is provided in the form of a map. We simulate exteroceptive noise as in [2, 27].

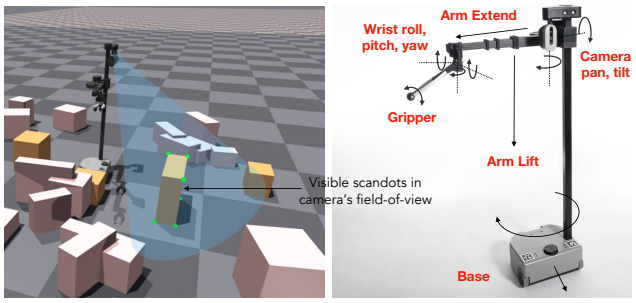



Figure 5. (Left) We compute visible scandots by projecting them to the camera frame and checking if they lie within the image plane (Right) the stretch RE1 robot that we use experiments. It has two DoFs in the base, one each for arm lift and extend, two for the camera, three for the wrist and one for the gripper.

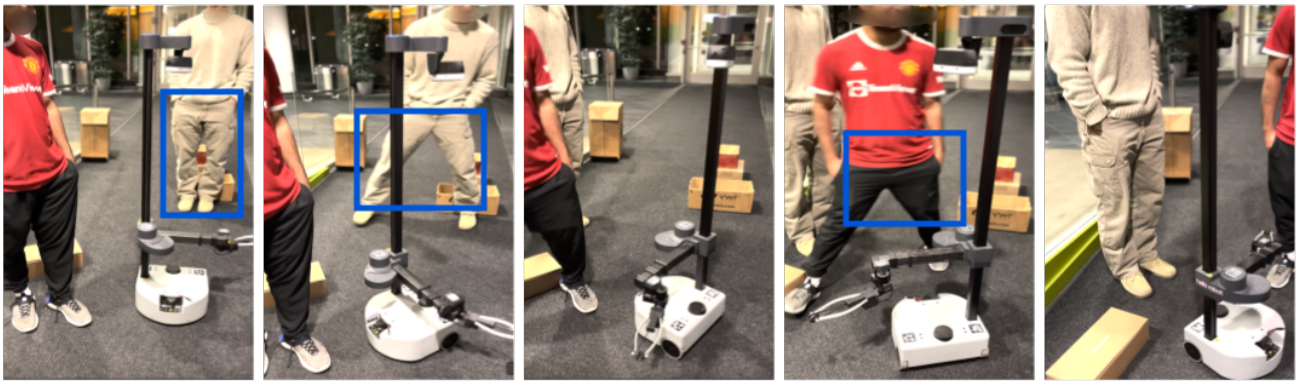
- **Classical:** This uses a classical stack to control the base motion. We first teleoperate the robot for 3-5 minutes to construct a map using the onboard 2D RPLidar using gmapping. Next, move_base is used to plan a path through the environment. Finally, we move the robot to the start, use a Monte Carlo method [21] to localize and then execute the plan. Note that this baseline gets an easier version of the problem since it assumes that the map is known in advance and does not consider arm motion due to the 2D Lidar. This is used to test whether reactive navigation is superior to planning.
- **NoPointNet:** Instead of passing object scandots through a permutation-invariant PointNet architecture, we concatenate them and use a MLP to estimate a latent.

## 4. Results and Analysis

We evaluate our approach both in simulation as well as real-world. Since doing a lot of in-the-wild real-world experiments is more time-consuming and cumbersome due to various practical reasons, we thoroughly evaluate our approach on 6 simulation benchmarks with multiple scenarios. We explain each of these benchmarks in detail in Section 4.3.

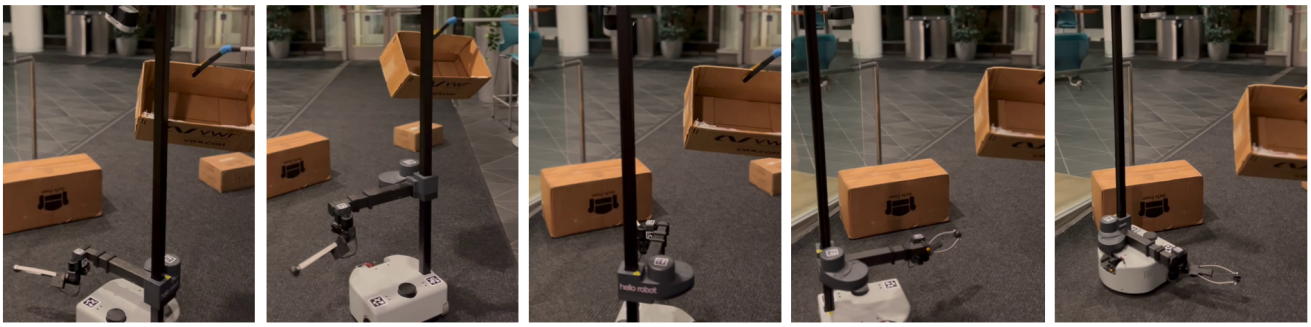
While simulation benchmarks are useful for fair comparison with baselines as well as reproducibility, real-world experimenting is essential for determining the efficacy of our system in truly unstructured and dynamic environments. For this, we test our system on various real-world environments as shown in Figure 1 and benchmark its performance on 2 real-world setups as described in Section 4.2.

Through simulation experiments, we aim to answer the following questions: (1) For a mobile agent, is active perception with an actuated camera really necessary, or is a fixed viewpoint enough? (2) Can an active visual agent outperform a classical agent that relies on pre-built maps? What are the limitations of the latter? (3) What are some practical architectural design choices for optimizing mobility and perception together? We empirically answer each of



(a) While our simulation lacks dynamic obstacles, the robot can still evade them because the policy continuously adjusts its plan.

(b) If there is an overhanging obstacle, the robot lowers its arm to avoid it instead of turning around, thereby displaying agile whole-body coordination.



(c) The robot adapts its plan on-the-fly to obstacles by turning around once the camera sees it.

Figure 6. Types of emergent behavior exhibited by *SPIN* (a) dynamic obstacle avoidance (b) whole-body movement (c) adaptive rerouting.

these questions in Section 4.3. Our real-world experiments primarily focus on comparing the capabilities of our reactive, learned system to a classical mapping and then planning approach. For our method, we observe interesting scenarios demonstrating emergent behaviors during real-world experiments detailed in Section 4.1 and 4.2.

### 4.1. Emergent Behavior

Large-scale simulation pre-training allows our robot to learn emergent behaviors to avoid obstacles in cluttered scenarios, even in the presence of dynamic obstacles. We see several such behaviors during real-world experimentation which were neither planned nor specifically trained for in simulation but emerge as a result of a large diversity of procedural environments seen during training. We illustrate three such scenarios in Figure 6. As highlighted in several frames, Figure 6a depicts robustness to adversarially placed dynamic obstacles that constantly block the path of the robot. It needs to continuously perceive its environment in multiple directions and quickly react to those changes. We observe that in cases when there is no feasible path for the robot to navigate through, it also learns to stop and look around in order to replan its path and avoid collisions. Similarly, in 6b we see that as soon as a floating obstacle is suddenly placed in front of the robot, it shows spatial awareness and whole-body coordination and lowers its arm in order to navigate through, instead of turning and replanning the entire base movement which would take more time. In Figure

| | Reach | | | | | | Pick | Place | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Scenario 1 | | | Scenario 2 | | | | Scenario 1 | | | Scenario 2 | | |
| | Easy | Medium | Hard | Easy | Medium | Hard | | Easy | Medium | Hard | Easy | Medium | Hard |
| *Success Rate*: | | | | | | | | | | | | | |
| FixCam | 1.00 | 0.53 | 0.20 | 1.00 | 0.50 | 0.26 | 0.86 | 1.00 | 0.53 | 0.16 | 0.97 | 0.50 | 0.20 |
| NoPointNet | 1.00 | 0.87 | 0.57 | 1.00 | 0.77 | 0.63 | 0.93 | 1.00 | 0.83 | 0.57 | 1.00 | 0.77 | 0.60 |
| Mapping | 1.00 | 1.00 | 1.00 | 0.86 | 1.00 | 0.97 | 0.97 | 1.00 | 1.00 | 1.00 | 1.00 | 0.90 | 0.97 |
| ***SPIN*(DVO)** | 1.00 | 1.00 | 0.96 | 1.00 | 1.00 | 0.90 | 0.97 | 1.00 | 1.00 | 0.90 | 1.00 | 0.90 | 0.90 |
| ***SPIN*(CVO)** | 1.00 | 0.97 | 0.93 | 1.00 | 1.00 | 0.93 | 0.97 | 1.00 | 0.97 | 0.90 | 1.00 | 0.97 | 0.93 |
| *Average Episode Duration (s)*: | | | | | | | | | | | | | |
| FixCam | 6.86 | 23.48 | 38.94 | 6.54 | 27.24 | 42.36 | 11.00 | 7.25 | 17.06 | 41.07 | 8.02 | 20.24 | 45.98 |
| NoPointNet | 6.30 | 14.87 | 33.25 | 7.22 | 15.04 | 34.09 | 9.25 | 4.88 | 15.22 | 32.00 | 7.89 | 18.49 | 37.42 |
| Mapping | 6.20 | 14.02 | 26.24 | 6.55 | 12.28 | 28.05 | 4.98 | 6.77 | 9.85 | 22.29 | 4.86 | 12.62 | 26.12 |
| ***SPIN*(DVO)** | 5.92 | 16.25 | 28.44 | 7.32 | 17.12 | 32.04 | 9.24 | 7.22 | 18.45 | 34.24 | 9.40 | 15.98 | 41.24 |
| ***SPIN*(CVO)** | 6.24 | 14.00 | 23.57 | 6.55 | 15.81 | 29.31 | 5.74 | 6.51 | 13.39 | 27.25 | 8.03 | 12.79 | 31.25 |

Table 1. We evaluate the success rate on 10 random environments with an average of 3 fixed seeds across all difficulty scenarios based on obstacle course. We report the success rate of each part of the task including reaching (Reach), picking (Pick), and placing (Place) the target object in the desired location. The place task requires the agent to bring back the object across the obstacles near its start location.

6c, we see an adaptive rerouting mechanism where the agent changed its straight line motion as soon as a person kicks in a box in front of it. These behaviors emerge in real-time and show the ability of our system to continuously perceive, adapt and react to changes in its environment which is very hard for a classical planner.

### 4.2. Real-world results

We test on two real-world scenes - an academic lab and an open study area with couches and a kitchenette next to it with both static as well as dynamic obstacles. Both these environments have unstructured clutter and humans as dynamic obstacles that makes it challenging for the agent to navigate through these spaces. For each environment, we have 4 static obstacles and atmost 1 dynamic obstacle thrown adversarially. We compare against a classical baseline that uses an A1 RPLidar with gmapping and move_base for planning. We first teleoperate the robot for 3-5min to create a map. Note that this provides the added advantage that this baseline knows the entire map in advance. Since the Lidar cannot see objects above the plane we only test on ground obstacles and ignore floating ones that require whole-body coordination. We run the planner to only plan the base motion. In Tab. 2 we compare success rate and average number of collisions. An episode succeeds when the robot reaches within 15cm of the specified goal position. Overall, our method is able to succeed 20-40% more than the classical baseline. This is because the classical method suffers from noise and is not able to recover from a noisy map, and gets stuck, whereas the learned policy learns to look at the obstacles again and again to improve its uncertainty estimates and constantly updates its knowledge of where obstacles are. This ability is even more apparent in the dynamic scenario (Table 2) where the classical has a near zero success rate while our method is able to succeed. It has the emergent ability to avoid a new obstacle in space, whereas the classical baseline relies on the pre-built map and fails entirely. Note that, we do not train our policy with dynamic obstacles in simulations, but this behavior comes out as a by-product of lots of diverse experience in simulation. We design the observation space such that everything is relative to the robot. This allows the agent to perceive the environment as moving within its local reference frame, allowing generalization to dynamic obstacles.

### 4.3. Simulation results

The simulation benchmarks have 6 scenes, 2 of each easy, medium and hard environments. Easy environments have 0-1 obstacles within a 5m goal range. Medium environments have 2-3 obstacles within 5m and the hard ones have heavily cluttered scenes with 5 obstacles within 5m. In each of these cases, one scene (Scenario 1) comprises of a tight 1m wide long corridor which bounds the agent to not take shortcuts and reach the goal only by navigating through obstacles. The second (Scenario 2) is an L-shaped corridor with the goal at the end. The evaluation metrics are reported as an average of 10 episodes with random agent and obstacle initialization across 3 seeds.

We compare against various baselines to study the impact of our design decisions in Tab. 1. For each scenario, we report the success rate and average episode length across 10 rollouts. Our method achieves $\approx$ 33% higher success rate than the NoPointNet baseline since permutation invariant scandots latent makes the optimization problem easier and also generalizes better at test time. Ours achieves $\approx$ 68% higher success rate than the FixCam baseline with the

| | Static Obstacles | | Dynamic Obstacles | |
|---|---|---|---|---|
| Scenario 1 | | | | |
| | Ours | Classical | Ours | Classical |
| Average Success | 0.8 | 0.6 | 0.6 | 0.0 |
| Average # Collisions | 1.0 | 0.4 | 1.6 | 1.2 |
| Scenario 2 | | | | |
| | Ours | Classical | Ours | Classical |
| Average Success | 0.8 | 0.4 | 0.6 | 0.2 |
| Average # Collisions | 0.8 | 0.6 | 1.6 | 1.0 |

Table 2. We compare our method against a classical mapping and planning baseline for navigation in cluttered scenes with both static as well as dynamic obstacles. The classical performs reasonably in static environments, it quickly breaks with dynamic obstacles like humans walking around, whereas our method shows more robust reactivity to such obstacles even without being trained with dynamic obstacles in simulation. We report the success rate of our method compared with the baseline. For the classical baseline, we teleoperate the robot for 2-3 min.

camera pointing straight ahead. This is because in some cases the robot encounters obstacles at its peripheral vision and our policy can change the camera angle to avoid them. Active vision is needed necessary for the robot to move effectively through a cluttered environment. Our method is significantly better than the Mapping baseline because the systematic noise in the object locations makes it hard for the robot to avoid them, especially in cluttered environments, whereas our method can continuously estimate the position of obstacles while it is moving and adapt the motion online. Finally, we compare between the decoupled (DVO) and coupled (CVO) variants of our method and find that they achieve similar performance. We hypothesize that the partial observability and joint optimization for camera and robot actions in CVO training allows the agent to quickly discover optimal shortcuts that are otherwise harder to distill from a privileged teacher policy.

## 5. Related Works

**Classical Approaches** The problem of navigating robots around obstacles has been studied for decades. Classical methods solve the motion and perception problem separately. First these methods build a map of the environment using the robot's onboard sensors such as cameras, proprioception and Lidar or infrared [7, 16]. Kalman-filter-based [36] techniques are often used to track positions, but they can't represent multi-modal ambiguities or recover after tracking failure [30]. Grid-based methods solve this but suffer from high memory usage [5]. Modern SLAM approaches ORBSLAM3 [28], OpenVSLAM [33] and RTAB [22] use variations of a method that relies on particle filters [35] to hold a multi-modal belief of the robot's location in the map [24, 34]. SLAM is especially challenging in dynamic environments due to the confounding motion of other agents [13, 31, 39, 42]. Once a map is built, a path can be planned over it. Exact paths can be computed using graph search algorithms [17], probabilistic methods which are faster but yield approximately optimal solutions [23] or potential-field based methods [20]. All of these assume perfect perception and re-planning is usually expensive making them susceptible to noise and precluding reactive behavior.

**Learning-based navigation** In recent years, learning has been used to improve the classical navigation stack. Modular approaches [8, 9, 15, 25] still leverage SLAM-based methods to build a map but use learning or heuristic changes to get priors for the best possible route to a goal. End-to-end approaches forgo maps entirely and train a policy to go from images to robot commands to go to a goal location [10, 11, 38]. We also take the end-to-end approach but unlike prior work where what the robot sees is fixed based on its position, in our case it must move its head and actively choose what it sees making optimization more challenging.

**Mobile Manipulation** A mobile base and arm together can complete useful in-the-wild manipulation but present a more challenging control problem. Imitation learning techniques focus on collecting large datasets in a variety of settings with a dexterous 6-dof arm and a wheeled mobile base using teleoperation [3, 4, 6, 12, 18, 40]. Because of the high-dimensionality of mobile manipulation, there is also control methods that leverage synergies between both the base and the arm and plans together. [14, 18, 19, 41].

## 6. Discussion and Limitations

We present *SPIN*, an approach to train robots that can simultaneously perceive, interact, and navigate cluttered environments using a data-driven approach. We show that our RL-based reactive approach is effective for active whole-body control-perception problem, traditionally addressed via non-reactive planning methods. With recent interest in humanoid and other mobile robots with actuated cameras, on neck for instance, *SPIN* is a cost-effective agile whole-body control solution with limited sensing and compute.

Although our robot can perceive geometry and avoid obstacles using depth, it still operates on stereo-matched depth instead of raw RGB. This leads to scenarios where it can bump into glass obstacles or shiny surfaces. In the future, we would like to use RGB for perception.

**Acknowledgements** We thank Jared Mejia and Mihir Prabhudesai for helping with stress-testing in real-world experiments. We are also grateful to Zackory Erickson and the Hello Robot team for their support with the robot hardware. This work was supported in part by grants including ONR N00014-22-1-2096, AFOSR FA9550-23-1-0747, and the Google research award to DP.

## References


[1] Stretch by hello robot. https://hello-robot.com/. 5

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023. 5

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 8

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 8

[5] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the national conference on artificial intelligence*, pages 896–901, 1996. 8

[6] Ben Burgess-Limerick, Chris Lehnert Jurgen Leitner, and Peter Corke. Enabling failure recovery for on-the-move mobile manipulation. *arXiv preprint arXiv:2305.08351*, 2023. 8

[7] Anthony R Cassandra, Leslie Pack Kaelbling, and James A Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, pages 963–972. IEEE, 1996. 8

[8] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. Goat: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023. 8

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020. 8

[10] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15691–15700, 2021. 8

[11] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *International Conference on Learning Representations*, 2019. 8

[12] Yuqing Du, Daniel Ho, Alex Alemi, Eric Jang, and Mohi Khansari. Bayesian imitation learning for end-to-end mobile manipulation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 5531–5546. PMLR, 2022. 8

[13] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of artificial intelligence research*, 11:391–427, 1999. 8

[14] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023. 8

[15] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2616–2625, 2017. 8

[16] J-S Gutmann, Wolfram Burgard, Dieter Fox, and Kurt Konolige. An experimental comparison of localization methods. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, pages 736–743. IEEE, 1998. 8

[17] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4 (2):100–107, 1968. 8

[18] Jesse Haviland, Niko Sünderhauf, and Peter Corke. A holistic approach to reactive mobile manipulation. *IEEE Robotics and Automation Letters*, 7(2):3122–3129, 2022. 8

[19] Jiaheng Hu, Peter Stone, and Roberto Martín-Martín. Causal policy gradient for whole-body mobile manipulation. *arXiv preprint arXiv:2305.04866*, 2023. 8

[20] Oussama Khatib. The potential field approach and operational space formulation in robot control. In *Adaptive and Learning Systems: Theory and Applications*, pages 367–377. Springer, 1986. 8

[21] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, pages 2149–2154 vol.3, 2004. 5

[22] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of field robotics*, 36(2):416–446, 2019. 8

[23] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5: 293–308, 2001. 8

[24] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998. 8

[25] Haokuan Luo, Albert Yue, Zhang-Wei Hong, and Pulkit Agrawal. Stubborn: A strong baseline for indoor object navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3287–3293. IEEE, 2022. 8

[26] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. 5

[27] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust

perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. 5

[28] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 8

[29] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 5

[30] Stergios I Roumeliotis and George A Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, pages 2985–2992. IEEE, 2000. 8

[31] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018. 8

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[33] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. Openvslam: A versatile visual slam framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2292–2295, 2019. 8

[34] Sebastian Thrun. Particle filters in robotics. In *UAI*, pages 511–518. Citeseer, 2002. 8

[35] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1):99–141, 2001. 8

[36] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995. 8

[37] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 4

[38] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019. 8

[39] Denis F Wolf and Gaurav S Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19:53–65, 2005. 8

[40] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022. 8

[41] Naoki Yokoyama, Alexander William Clegg, Eric Undersander, Sehoon Ha, Dhruv Batra, and Akshara Rai. Adaptive skill coordination for robotic mobile manipulation. *arXiv preprint arXiv:2304.00410*, 2023. 8

[42] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1168–1174. IEEE, 2018. 8