SPIN: Simultaneous Perception, Interaction and Navigation

Supplementary Material

7. Qualitative Results

We test our framework in several in-the-wild scenarios, some of which are illustrated in Figure 1. Qualitative video results are available at https://spin-robot.github.io/.

We see emergent behavior where the robot continuously avoids dynamic obstacles without seeing them during training. We also observe generalization heavily cluttered indoor to dim-lit outdoor environments. The agent also demonstrates reactive whole-body coordination where it moves its arm up or down to efficiently navigate across floating obstacles instead of re-routing and re-planning base movement, demonstrating 3D spatial awareness.

8. Implementation Details

We make several design choices for the working of our framework. Firstly, the robot is only allowed to have local visibility in order to develop highly reactive and instant behaviors. At any time instant t, the agent can perceive it's environment within a range of 2m in all 4 direction – front, back, left and right based on the camera's viewing direction and it's field of view. We also empirically observe that given larger viewing range, say $>$ 5m which contains information of more than 4-5 nearest obstacles to the agent makes it a sub-global path planning problem which becomes harder to optimize, leading to degraded behavior and performance as reported in Table 3.

Visibility Range	Success Rate ↑	Distance to Goal (m) ↓
$\leq$ 1m	0.96	0.28
$\leq$ 2m	0.96	0.26
$\leq$ 3m	0.93	0.63
$\leq$ 5m	0.86	1.21

Table 3. We report the average success rate and average distance to goal for 10 episodes across 3 seeds each with different maximum visibility range for the agent at any time instant. As reported, with broader visibility, the agent shows more frequent stalling leading to higher average distance to goal.

Secondly, contrary to standard teacher-student architectures which use privileged system information for training the teacher, we restrict the privileged information to only elements in the robot's field-of-view that can be retrieved from the ego-view in the same state. For this, we project the obstacle scandots onto the image plane and pass only those scandots to the robot observation which lie in the camera's field-of-view. Note that, if this were not the case, the robot does not learn camera movement in a relevant fashion and also becomes harder to distill into a depth-conditioned student policy through only ego-centric view. Similarly, for the 3-phase decoupled visuo-motor optimization, we induce information bottleneck with a low-dimensional latent space of size 16 for the scandots latent while training the teacher policy, which again helps it in attending to most relevant information at any time instant t in order to make student policy distillation feasible.

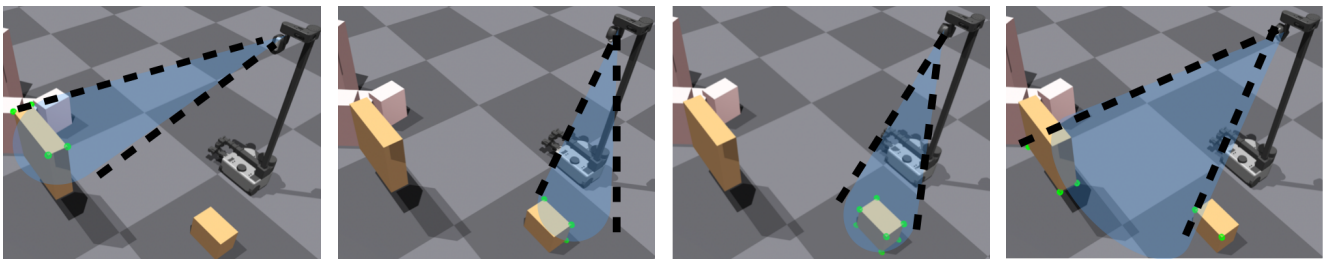
Observation Space. The observation space for the robot comprises of joint positions (q), joint velocities (q_vel), end-effector position p_{eef}, goal position (p_{goal}) and depth latent $(\hat{z})$ containing visual information about the environment. Note that during phase 1 training in simulation, scandots (z) are used as a proxy for faster depth rendering and later distilled into a egocentric depth-conditioned policy. During real-world deployment, p_{eef} is obtained via forward kinematics and other proprioception information is obtained directly from the robot.

Action Space. The action space of the robot consists of the velocity for base rotation as well as translation and joint positions for all the other joints including arm, camera as well as gripper actions. The gripper action is a continuously varying scalar which can actuate the gripper to different extents, unlike a binary action indicating open or closed gripper.

Reward Scales. We use a distance and forward progress goal for reaching, a binary reward for grasping and a continuous shaped reward for lifting the object to a certain height above the table. We also add a small penalty for the joint velocities to the arm stretch and camera joints for a temporally smooth gait which permits easier sim2real transfer as well as more appropriate behaviors leading to less jitter and more consistent movements on the real hardware.

The reward scales used for goal reaching, grasping and lift rewards are reported in Table 4. Detailed formulations of the reward functions are described in Section 2.1.

Network Architecture and Training Details. The actor and critic for teacher policy are LSTM with 256 hidden units, with input as prorpioception, goal and scandots latent. The scandots are compressed using a pointnet architecture for permutation invariance. The depth network for the student policy takes as input a low-resolution depth image of size 58×87 and comprises of 3-layer convolution backbone followed by 3 fully-connected layers. We use Adam



(a) Pose I: Facing forward (b) Pose II: Facing downward (c) Pose III: Facing downward and slightly forward (d) Pose IV: Facing forward and slightly downward

Figure 7. Different camera poses for the FixCam baseline.

	Reward Scale
Reach Reward	0.1
Grasp Reward	0.5
Lift Reward	0.8
Joint Velocity Penalty	-0.03

Table 4. We report the average success rate and average distance to goal for 10 episodes across 3 seeds each with different maximum visibility range for the agent at any time instant. As reported, with broader visibility, the agent shows more frequent stalling leading to higher average distance to goal.

Optimizer with an initial learning rate of $1e-3$, entropy coefficient of $5e-4$ and γ as 0.99.

Asynchronous DAgger Training. Since depth rendering on simulators is a computational bottleneck, we implement an asynchronous version of DAgger algorithm which simultaneously collects data in a buffer and trains the student policy with batches sampled from the collected data using 2 parallel processes. This provides a $2.5\times$ computational speedup over the non-parallelized version of the algorithm, allowing faster convergence of the student network. We also find that freezing the weights of the student actor pre-initialized from the teacher policy for first 1000 iterations helps as warm-up steps to the depth convolution backbone for stable training.

Post-processing for clean depth images. To mitigate the issues due to noisy depth, we post-process the depth obtained from the Intel RealSense Camera using a real-time fast hole-filling algorithm for depth images [23]. With the camera constantly in motion, there are additional artefacts with depth images. For this, we additionally use temporal filtering over the stream of depth images.

Object Detection for Pick Policy Once the robot reaches near the goal, we randomly select an object within its field of view in order to be grasped and fetched to a target location. For getting the target object location, we run YOLO [20], a real-time object detection model with an average inference speed of 20ms. We use the corresponding depth image to deproject the pixel point into a 3D-coordinate which is passed as the new goal position to the manipulation policy.

	Scenario 1			Scenario 2		
	E	M	H	E	M	H
I	0.93	0.40	0.20	0.97	0.40	0.20
II	0.70	0.30	0.10	0.67	0.47	0.10
III	0.86	0.33	0.10	0.77	0.30	0.10
IV	1.00	0.53	0.20	1.00	0.50	0.26

Table 5. Success rate for 4 FixCam poses in easy (E), medium (M), hard (H) envs.

9. Analysing camera and base motion

Fixed Camera Baseline We run FixCam baseline with 4 camera poses (Figure 7) – **I:** Front, **II:** Down, **III:** Down and slightly front, **IV:** Front and slightly down on easy (E), medium (M), hard (H) environments. **I**, **IV** with max fov have much lower success than SPIN, implying active vision is required in clutter. Pose (**IV** depicts the FixCam baseline referred in the paper.

Camera Movement and Camera Observations: We show camera trajectory in Figure 8. When navigating through clutter (frames 1, 2, 4), it tilts downward to maximize fov near the base, but with no nearby obstacle (frame 3), it faces *front*. Detailed movement of the camera can be seen on the website along with paired RGBD images for rollouts. RGB frames are only for analysis, the policy only observes depth images.

Necessity for Active Vision and Whole-body coordination *Active Vision:* In principle, a multi-camera system should be equivalently adequate, however most views will contain insignificant information and require large models

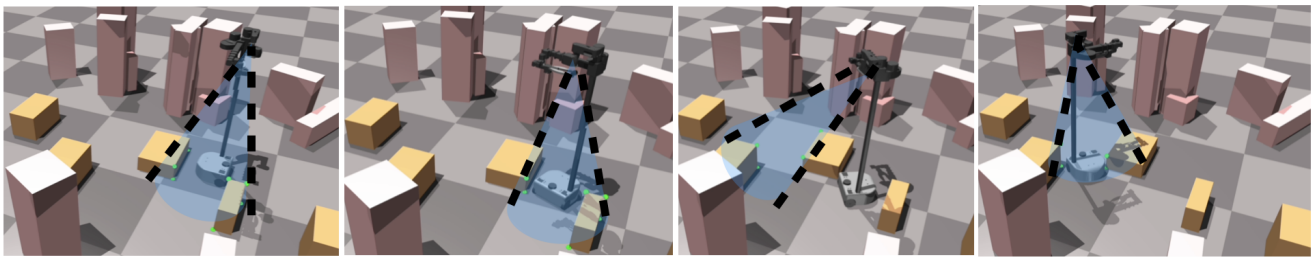


Figure 8. Camera movement analysis in a trajectory. The agent faces the camera downward when navigating through tightly cluttered vicinity as can be seen in the first, second and fourth frame, whereas the camera points more towards the front when there are no immediate obstacles in the direction of movement, as illustrated in the third frame.

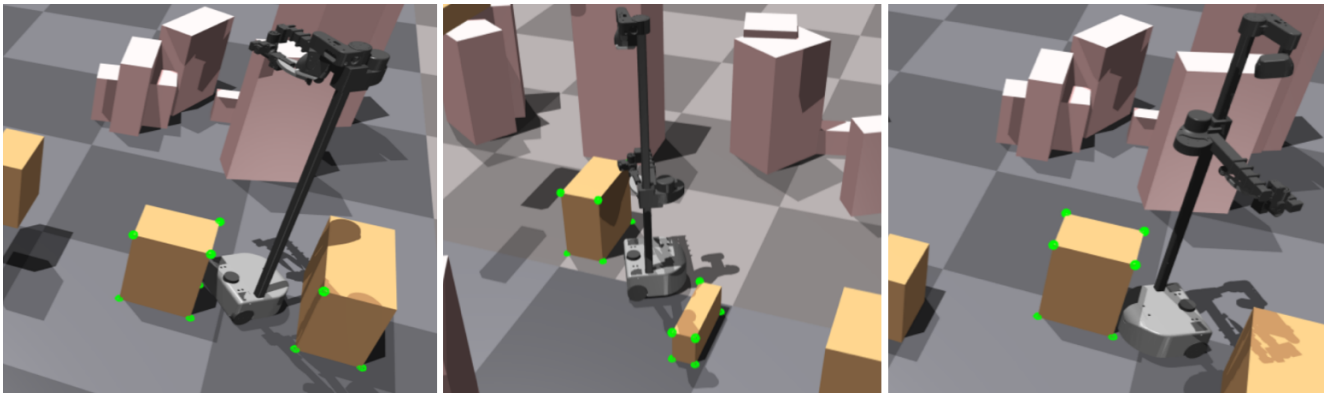


Figure 9. Scenarios where whole-body coordination is essential under heavy obstructions. In the above cases where the obstacles are tightly packed, it is not possible for the robot to navigate through them avoiding collisions without lifting the arm to an appropriate height.

to process. With limited onboard compute on most robots and requirement for real-time reactivity ($< 0.1s$), it becomes infeasible to deploy them with larger vision backbones.

Whole-body coordination (WBC): Under heavy obstructions, the robot cannot move without collision if the base & arm control are decoupled. Figure 9 shows such a scenario where a fixed arm and gripper close to base would fail without WBC, which also allows it to use the extra degree of freedom to find shorter and more efficient paths.

10. Directly training from depth images.

We compare training from depth (red) and our 2-phase method with scandots (blue) in a medium difficulty environment as illustrated in Figure 10. Depth policy has $< 1\%$ success after 22h training, whereas total (*phase1* + *2*) wall-clock time for SPIN is 16h (6+10). The simulator gives $\approx 50k$ fps for scandots (8192 envs) and ≈ 820 fps for depth (256 environments – maximum parallel environments that can fit on a single GPU), causing $61\times$ slow-down bottleneck. This shows the necessity and efficiency of our proposed 2-phased coupled visuomotor optimization approach using scandots over naively training an RL policy from depth observations.

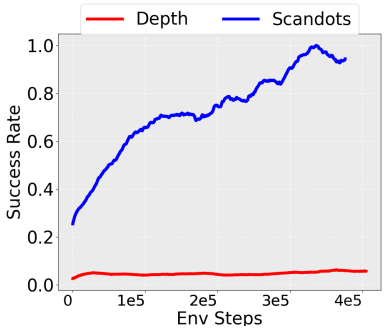


Figure 10. Success rate for scandots (blue) vs depth (red). The depth-based policy attains close to 0 performance even after 400k env steps of training, whereas the policy trained with scandots increasingly improves over time.

11. Classical Navigation Baseline

As discussed in Section 3, we compare our method with a classical map-based baseline which uses the 2D RPLidar to build an environment map and then creates a plan using Monte Carlo method. We observe that for $> 90\%$ of the

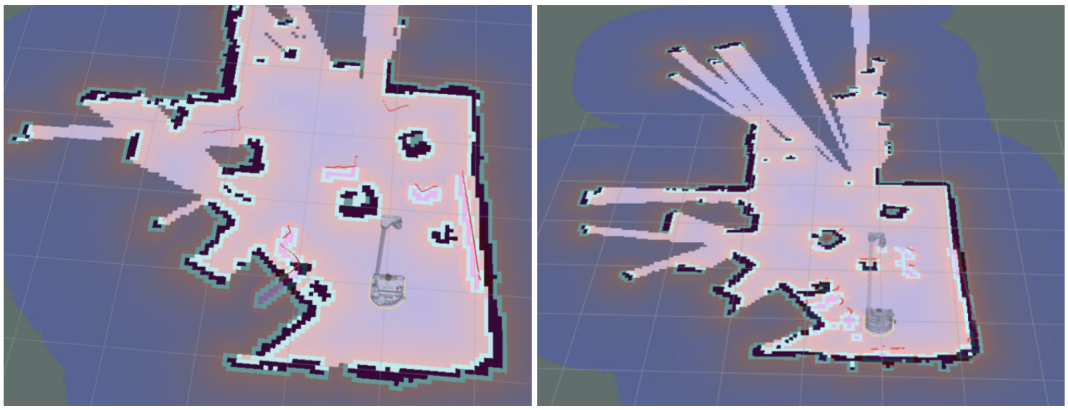


Figure 11. Visualizations of environment map built using 2D Lidar. The robot is localized as per its initial position and orientation.

cases, the robot is able to build a map and find a feasible path, however it is not able to execute the planned path for $> 85\%$ cases. This issue arises due to noisy control or unexpected wheel motion due to terrain differences. Our method is able to overcome such failures due to a constant feedback and reactive improvisation through proprioception as well as depth, which allows it to deal with uncertainties without requiring a pre-built environment map. Moreover, due to localization inaccuracies, the baseline method is often unable to reach the intended goal if not initialized in the same orientation as was used before building the map. Contrary to that, we heavily randomize all degrees of freedom as well as the robot orientation at the beginning of every rollout during test time. Moreover, since the robot has a 2D Lidar installed on it, we do not test it in environments with floating obstacles which would require 3D understanding and whole-body coordination to navigate through clutter. We show some visualizations of the map built and plans created by the robot in Figure 11.