Learning to Navigate Using Mid-Level Visual Priors

Alexander Sax[1,3] **Jeffrey O. Zhang**[1] **Bradley Emi**[2] **Amir Zamir**[1,2]
Silvio Savarese[2] **Leonidas Guibas**[2,3] **Jitendra Malik**[1,3]

[1]University of California, Berkeley [2]Stanford University [3]Facebook AI Research

http://perceptual.actor

Abstract: How much does having *visual priors* about the world (e.g. the fact that the world is 3D) assist in learning to perform *downstream motor tasks* (e.g. navigating a complex environment)? What are the consequences of not utilizing such visual priors in learning? We study these questions by integrating a generic perceptual skill set (a distance estimator, an edge detector, etc.) within a reinforcement learning framework (see Fig. 1). This skill set ("mid-level vision") provides the policy with a more processed state of the world compared to raw images.

Our large-scale study demonstrates that using mid-level vision results in policies that *learn faster*, *generalize better*, and achieve *higher final performance*, when compared to learning from scratch and/or using state-of-the-art visual and non-visual representation learning methods. We show that conventional computer vision objectives are particularly effective in this regard and can be conveniently integrated into reinforcement learning frameworks. Finally, we found that no single visual representation was universally useful for all downstream tasks, hence we computationally derive a task-agnostic *set* of representations optimized to support arbitrary downstream tasks.

Keywords: Representation Learning, Visuomotor, Reinforcement Learning, Perception, Generalization, Sample Efficiency, Navigation.

1 Introduction

The resurgence of deep reinforcement learning (RL) began with a number of nominal works, e.g. the Atari DQN paper [1] or *pixel-to-torque* [2], which collectively showed that RL could be used to train policies directly on raw images. Although deep-RL-from-pixels can learn arbitrary policies in an elegant and end-to-end fashion, there are two phenomena endemic to this paradigm: **I.** learning requires massive amounts of data (large sample complexity), and **II.** the resulting policies do not transfer well across environments with even modest visual differences (generalization difficulties).

These two phenomena are characteristic of a type of learning that is *too general*—in that it does not make use of available *priors*. In the context of visual perception (the focus of this paper), an example of such priors is that the world is spatially 3D; or there exist certain useful groupings, i.e. "objects". These priors are basically *facts* of the world and incorporating them in learning is notably advantageous [3, 4]. That is because the assumption-free style of learning may recover the proper policy but at the expense of massive amounts of data to rediscover such facts (issue **I**); or may resort to shortcuts spawned by spurious biases in the data to superficially learn faster, which leads to generalization difficulties (issue **II**) [5].

One of the goals of computer vision is formulating useful visual priors about the world and developing methods for extracting them. Conventionally,

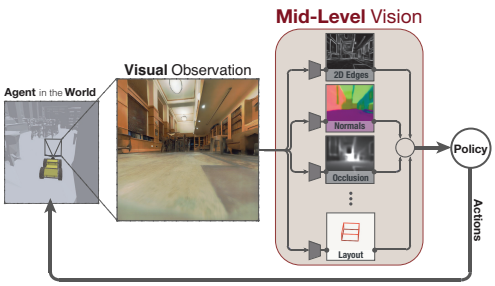

Figure 1: **Mid-level vision in an end-to-end framework for learning active robotic tasks.** We report significant advantages in *final performance*, *generalization*, and *sample efficiency* when using mid-level vision, especially compared to learning directly from raw pixels (i.e. bypassing the beige box).

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

this is done by defining a set of problems (e.g. object detection, depth estimation, etc.) and solving them independently of any ultimate downstream active task (e.g. navigation, manipulation) [6, 7]. In this paper, we study how such standard vision objectives can be used within RL frameworks as mid-level visual representations [8], in order to train effective visuomotor policies.

We show that incorporating mid-level vision can alleviate the aforementioned issues **I** & **II**, resulting in improved *final performance*, *generalization*, and *sample efficiency*. We demonstrate that mid-level vision performs significantly better than SotA state representation learning methods and learning from raw images – which we find to perform no better than a blind agent when tested on *unseen test data*. Finally, we observe that policies trained using mid-level vision exhibit desirable properties for which they were not explicitly trained, without having to do reward shaping.

Our study is done using 24 different mid-level visual representations to perform various navigation based downstream tasks in 3 different environments (Gibson [9], Habitat [10], *ViZDoom* [11]). Our mid-level vision comes from neural networks trained by existing vision techniques [12, 13, 14] using real images. We use their internal representations as the observation provided to the RL policy. We do not use synthetic data to train the visual estimators nor do we assume they are perfect.

An interactive tool for comparing various trained policies accompanied with videos and reward curves, as well as the trained models and code is available on our website.

2 Related Work

This study has connections to a range of topics, including transfer learning, un/self supervised learning, lifelong learning, reinforcement and imitation learning, control theory, active vision and several others. We overview the most relevant ones within constraints of space.

Computer Vision encompasses approaches that are conventionally designed to solve various standalone vision objectives, e.g. depth estimation [15], object classification [16], detection [17], segmentation [18], pose estimation [19, 20], etc. The approaches use various levels of supervision [16, 21, 22, 23], but the characteristic shared across these methods is that they are *offline*, i.e. trained and tested on prerecorded datasets and evaluated as a fixed pattern recognition problem. In contrast, the perception of an active agent is fundamentally used in an *online* manner, i.e. the perceptual skill is in service to a downstream goal and the current perceptual decision impacts what the next perceptual observation will follow. Here we study how conventional computer vision objectives can be plugged into such frameworks used for solving downstream active tasks, e.g. navigation.

Representation/Feature Learning shares a goal with our study: to understand how to encode images in a way that provides benefits over using just raw pixels. Many of the most popular representation learning techniques like Variational Autoencoders (VAE) [24] or alternatives [25, 26, 27] are based on Minimum Description Length (MDL) which roughly suggests that "the best representation is the one that leads to the best compression of the data." We show this assumption is not valid, as the most compressive representations are not found to support downstream tasks well.

Other techniques model the dynamics of the environment (e.g. by predicting the next state [28, 29] or by other related objectives [30, 31, 32, 33, 34]). One advantage of modeling the dynamics is that the representations could useful for planning. These dynamics are not necessarily visual and they may be specialized to the particular morphology or action space of the agent.

A compendium of several concurrent and recent works have offered supporting evidence that mid-level visual representations could be useful for realistic downstream active tasks: e.g. a specific semantic representation for semantic driving ([35, 36, 37]) or dense object descriptors for manipulation ([38]). Independently of our work, Zhou et al. [39] also studied the role of visual abstractions for learning to act. That work focused on learning policies for synthetic driving and first-person shooter environments; showing that agents equipped with intermediate representations (optical flow, depth, semantic segmentation, and albedo), train faster, achieve higher task performance, and generalize better. While the details of the environments, tasks, and representations differ, the findings are broadly aligned and appear to support each other.

Robotics has long used intermediate visual abstractions such as depth (e.g. SLAM [40]), optical flow [41], or ground-plane estimation [42]. However, these usually use the output of the vision solutions in some analytic fashion [43] which requires the representations to be easy to analyze analytically. When the presesentation is not analytically well understood or in presense of noise (e.g.

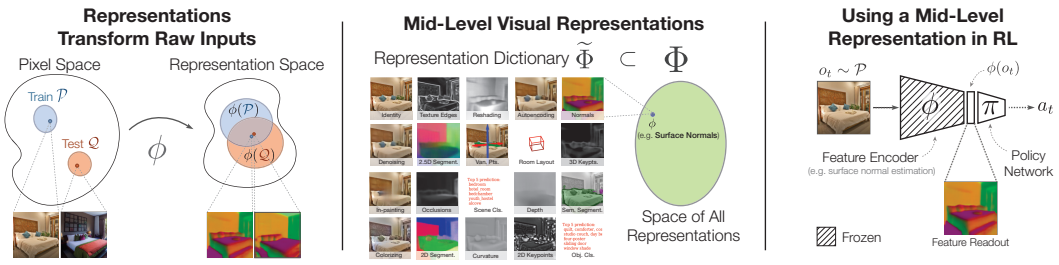

Figure 2: **Illustration of our approach.** *Left:* The job of a feature is to warp the input distribution, potentially making the train and test distributions look more similar to the agent. *Middle:* Visualizations for 19 of 24 mid-level vision objectives in our dictionary $\widetilde{\Phi}$. The dictionary is a sample of all possible transforms (Φ), and the best function for a given task must have the proper "invariance", i.e. ignore irrelevant parts of the input while retaining the information required for solving the downstream task. *Right:* Representations from fixed encoder networks are used as the observation for training policies in RL.

the latent features from a VAE, noisy depth information) such methods cannot be used. The end-to-end approach has no such constraint, but most end-to-end methods learn with simplistic visual features or *tabula rasa*. This paper studies the utility of incorporating mid-level visual features in a learning-based end-to-end frameworks.

3 Methodology

Our goal is to study the role of mid-level representations of raw visual data toward performing downstream robotic tasks better. More concretely, the goal is to maximize an agent's performance in a test setting ($\mathcal{Q}$) that is similar *but not identical to the training distribution*, $\mathcal{P}$. Our setup assumes access to a set of functions $\Phi = \{\phi_1, \ldots, \phi_m\}$ that can be used to transform raw sensory data into some (possibly useful) representation. We define $\mathcal{P}_\phi \triangleq \phi(\mathcal{P})$ as image of the the training distribution under ϕ as and $\mathcal{Q}_\phi \triangleq \phi(\mathcal{Q})$ analogously. This shown in Fig. 2, left). In this paper we show that when this dictionary is a set of mid-level visual representations, agents can achieve better final performance, generalization, and sample efficiency.

3.1 Using Mid-Level Vision for Active Tasks: The Role of Representations

Why could a visual feature, e.g. image $\rightarrow$ surfance normals, improve test-time performance on a downstream task, compared to simply using the image raw? A good representation ϕ transforms the inputs in a way that preserves the task-relevant information while eliding task-irrelevant differences between the train and test distributions. Roughly speaking, a good representation makes $\mathcal{P}_\phi \approx \mathcal{Q}_\phi$ without affecting the training reward ($R_{\mathcal{P}_\phi}$), as illustrated in Fig. 2-left. In this way, using RL to maximize the training reward also improves the test-time performance, $R_{\mathcal{P}_\phi \rightarrow \mathcal{Q}_\phi}$.

Our mid-level representations come from a set of neural networks that were each trained, offline, for a specific vision objective [12] (see Fig. 2-middle). We freeze each encoder's weights and use the network (ϕ) to transform each observed image o_t into a summary statistic $\phi(o_t)$ that we feed to the agent. During training, only the agent policy is updated (as shown in Fig. 2-right). Freezing the encoder networks has the advantage that we can reuse the same features for new active tasks without degrading the performance of already-learned policies. This approach is almost certainly not ideal, but agents trained using mid-level representations in this way still outperform the current SotA.

3.2 Core Analysis: Final Performance, Generalization, and Rank Reversal

Final Performance: We evaluate agents in a *test space* distinct from where the agents were trained. Maintaining a train/test split is crucial, and we found that training performance was not necessarily predictive of test performance.

Generalization: We provide a detailed analysis of how different agents generalize—reporting both the common metric of generalization (the gap between train and test buildings) and alternatives (e.g. performance of test episodes significantly longer or harder than the training episodes).

Sample Complexity: We examine whether an agent equipped with mid-level vision can learn faster than a comparable agent that learns *tabula rasa* (i.e. with vision, but no priors about the world). We

report sample complexity both in terms of the number of training updates/frames (the usual metric), and also as a function of how many buildings (sampling clusters) are in the training set.

3.3 Rank Reversal: Which Mid-Level Feature to Use?

*Can a **single** feature support all downstream tasks? Or is a set of features required for gaining these feature benefits in arbitrary tasks?* We demonstrate that no feature is universally useful for all downstream tasks (Sec. 5.4). We show this by demonstrating cases of *rank-reversal*—when the ideal features for one task are non-ideal for another task (and vice-versa). Formally, for tasks T_0 and T_1 with best features ϕ_0 and ϕ_1 respectively, where the test reward for each task is denoted $R_i \triangleq R_{\mathcal{P}_i \to \mathcal{Q}_i}$ we show that $R_0(\phi_0) > R_0(\phi_1)$ and $R_1(\phi_0) < R_1(\phi_1)$. For instance, we find that *depth estimation* features perform well for visual exploration and *object classification* for target-driven navigation, but neither do well vice-versa.

3.4 Max-Coverage Feature Set for Arbitrary Tasks

As a consequence of the rank-reversal phenomenon, one needs to select the mid-level feature based on the current downstream task of interest, and keep updating it every time that task changes. In this section we ask: when the downstream task is unknown or changes, could a predetermined set of features provide better worst-case (generic) perception than using any single feature? We give an example of such a set: the *Max-Coverage Feature Set*, which is a minimal covering set of the space of useful visual abstractions. In Sec. 5.5, we demonstrate that this set can capture the benefits of mid-level vision, comparable to if we had known the best feature *a priori*. Finding the Max-Coverage (M-C) feature set can be formulated as a sequence of $O(\log |\widetilde{\Phi}|)$ Boolean Integer Programs and solved efficiently in < 4 seconds. Since the main goal of our study is to examine the utility of mid-level vision and to demonstrate that no single feature is universal, we provide the detailed formulation and analysis of the M-C feature set in Appendix A.2.

4 Case Study: Vision-Based Navigation

We adopt a class of active tasks (navigation) and apply the described methodology to perform our study. The study examines representations driven by 24 different mid-level objectives, compared against 8 state-of-the-art baselines and 3 separate controls, all on 3 distinct navigation tasks. The remainder of this section describes our experimental setup, and complete details are in the appendix.

Environments: Our experimental setup is the same as in the CVPR 19 Habitat Challenge [44]; we use the Habitat platform [10] with the Gibson [9] dataset. The dataset captures the intrinsic visual and semantic complexity of real-world scenes by scanning 572 actual buildings. See our website for videos of the trained policies generalizing to real robots (with no finetuning).

To establish the universality of the results, besides Habitat, we also perform the study using two other environments: *Gibson Environments* [9] (a visually realistic simulator integrated with PyBullet dynamics and operating on Gibson dataset) and *ViZDoom* [11] (a 3D first-person game).

Figure 3: **Sample of 7 buildings from the Gibson [9] dataset.** One floor is shown from each space. Boxed images indicate sample observations from the Gibson [9] environment, while observations in Habitat [10] come directly from the (shown) mesh textures. We use 72 buildings for training and 14 for testing.

Train/Test Split: We train and test our agents in two disjoint sets of buildings (Fig. 3); test buildings are completely unseen during training. We use up to 72 building for training and 14 test buildings for testing. The train and test spaces comprise $15678.4m^2$ (square meters) and $1752.4m^2$, respectively.

4.1 Downstream Navigation Tasks

We present our case study using three common and useful navigation-type tasks: *local planning*, *visual exploration*, and *navigation to a visual target* (see videos here); described below (detailed in Appendix A.3).

Local Planning (aka Point Goal [45, 10]): The agent must direct itself to a given nonvisual target destination (specified using coordinates), avoiding obstacles and walls as it navigates. This skill might be useful for traversing sparse waypoints along a desired path. During training, the agent receives a large one-time reward for reaching the goal, and in the dense-reward variant, also receives positive reward proportional to the progress it makes (in Euclidean distance) toward the goal. There is also a small negative reward for living. The maximum episode length is 500 timesteps, and the target location is 1.4 to 15 meters from the start.

Visual Exploration: The agent must visit as many **new** parts of the space as quickly as possible. The space is partitioned into small occupancy cells that the agent "unlocks" by scanning with a myopic laser range scanner. This scanner reveals cells directly in front of the agent for up to 1.5 meters. The reward at each timestep is proportional to the number of newly revealed cells. The episode ends after 1000 timesteps.

Navigation to a Visual Target: In this scenario the agent must locate a specific target object (a wooden crate) as fast as possible with only *sparse rewards*. Upon touching the target there is a large one-time positive reward and the episode ends. Otherwise there is a small penalty for living. The target (wooden crate) is fixed between episodes although the agent must learn to identify it. The location and orientation of both the agent and target are randomized. The maximum episode length is 400 timesteps, and the shortest path is usually over 30.

Sensory Input: For each task, the sensory observation space contains RGB images of the onboard camera. In addition, the *minimum* amount of side information needed to feasibly solve the downstream task are appended; that is the target direction/location for local planning, unlocked occupancy cells for exploration, and nothing for Visual Target Navigation. Unlike the common practice, we do not include proprioception information such as the agent's joint positions, velocities, or any other unessential side information in order to strictly test the perceptual skills of the agents.

Action Space: We assume a low-level controller for robot actuation, enabling a high-level action space of $\mathcal{A} = \{\texttt{turn\_left(10°)}, \texttt{turn\_right(10°)}, \texttt{move\_forward(0.25}m\texttt{)}\}$.

4.2 Reinforcement Learning Algorithm

We use an off-policy variant of Proximal Policy Optimization [46] (PPO, details in Appendix A.6), with a small controller network. For each task and each environment we conduct hyperparameter searches optimized for *scratch* and all the state-of-the-art baselines (see section 4.4). For our mid-level agents, we use the same hyperparameters that were optimized for scratch.

4.3 Mid-Level Representations

For our experiments, we used representations derived from one of 24 different computer vision objectives (Fig. 2). This set covers various common modes of computer vision objectives: from texture-based (e.g. denoising), to 3D pixel-level (e.g. depth estimation), to low-dimensional geometry (e.g. room layout), to semantic (e.g. object classification). For these objectives we used the networks of [12] trained on a dataset of 4 million static images of indoor scenes [12]. All networks were trained with identical hyperparameters and using a ResNet-50 [47] encoder. For a full list of vision objectives and samples of the networks evaluated in our environments, see Appendix A.4.

4.4 State-Representation Baselines

We include multiple control groups in order to address possible confounding factors, and we compare against several state-of-the-art baselines. We describe the most important ones here and defer remaining descriptions to Appendix A.1.

Tabula Rasa (Scratch) Learning: The most common approach, *tabula rasa* learning trains the agent from *scratch*. In this condition, the agent receives the raw RGB image as input and uses a randomly initialized AtariNet [1] tower that is trained with the policy.

Blind Intelligent Actor: The *blind* baseline is the same as *tabula rasa* except that the visual input is a fixed image and does not depend on the state of the environment. The *blind* agent is a particularly informative and crucial baseline since it indicates how much performance can be squeezed out of the nonvisual biases, correlations, and overall structure of the environment. For instance, in a narrow straight corridor which leads the agent to the target, there should be a small performance gap between *sighted* and *blind*.

State-of-the-Art Representation Learning: We compare against several state-of-the-art representation-learning methods, including *dynamics-modeling* [48, 49, 28], *curiosity* [31], *DARLA* [50], and *ImageNet pretraining* [16], enumerated in Figure 4.

Non-Learning: Methods such as SLAM [40] have long used intermediate representations such as depth, but inside a specialized non-learning framework. SLAM may not always be applicable, but when it is it shows how much of the problem can solved with hand-engineered systems.

5 Experimental Results

This section presents results from a case study of agents trained with mid-level representations. In the main paper we primarily focus on the *Local Planning* task performed in Habitat [10]. The supplementary material provides the results of the experiments in other environments (Gibson, Doom) and with other downstream tasks (exploration, visual target navigation), which show the same trends.

5.1 Final Performance: Mid-Level Features Exhibit Better Performance

Higher reward on the test set: Agents using mid-level visual representations achieve performance significantly higher than agents trained from scratch (Fig. 4, left). This was tested in both Gibson [9] environment and Habitat [10], as shown in Fig 4. The Spearman's ρ between performance in Habitat and Gibson was 0.87, indicating a high degree of agreement between rankings in the two environments. Notably, ***scratch*** *and several of the SotA features do not perform much better than a* ***blind*** *agent in the test space*, which suggests they heavily overfitted to the training set (Appendix B.2.1).

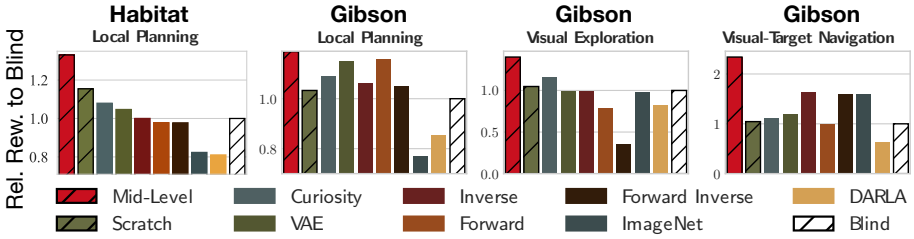

Figure 4: **Agents using mid-level vision had higher reward on the test set.** Each bar plot compares average test-set reward on a different task. On every task, agents using mid-level vision outperformed those learning *from scratch* or using alternative SotA representation-learning methods. Significance tests in Appendix B.2.

Desirable emergent behavior without reward engineering: Although agents were trained only to maximize training reward, we found that mid-level-vision-based agents exhibited other desirable properties such as *fewer collisions, less acceleration and jerk, and better performance on alternative task metrics*, as shown in Fig. 5 (a,b)[1]. Other papers [51] have specifically built in this desirable behavior, but we find that simply adding in appropriate perception goes a long way towards fixing the issue without having to hand engineer the reward.

When using mid-level vision, agents performed equally well with sparse or dense rewards (0.74 vs 0.76 SPL)[2] and significantly outperformed SotA methods–*even when the SotA methods used dense reward and/or were explicitly engineered to handle sparse reward* (e.g. Pathak et al. [31]: 0.56 SPL).

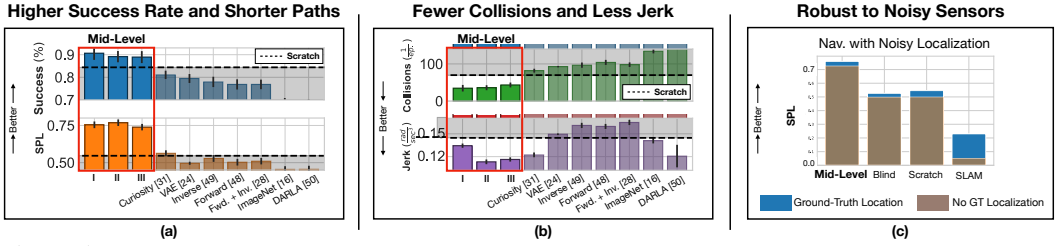

Figure 5: **Desirable emergent behavior and robustness under uncertainty.** *(a),(b):* The colorful bar charts show that agents using mid-level vision learn desirable behavior that is not explicitly coded for in the reward: they experience fewer collisions, less jerk[4], and achieve a higher success rate and shorter average path length. *(c):* Removing ground-truth agent localization (gap between the blue and brown rectangles) harms localization and hurts classical methods and *scratch* more than mid-level agents.

More robust agents: Fig. 5 (c) demonstrates that mid-level vision-based agents robustly handle noisy inputs. We removed agents' access to ground-truth agent localization (via an inertial measurement unit) and found that **(1)** the gains from using mid-level priors were much larger than the gains from using a (ground-truth) map (+0.23 SPL and +0.05 SPL vs. *scratch*), **(2)** mid-level agents without GT localization still outperformed other approaches *even when those used a map* (0.73 vs. 0.55 SPL). **(3)** classical approaches such as SLAM exhibited poor performance with estimated (instead of ground-truth) localization (0.05 SPL) or depth (0.23 SPL).

[1] The top mid-level features (I, II, and III) were chosen based on reward (see Sec. 5.4). Error bars = 1 SE.

[2] SPL = Success Weighted by Path Length (as in [45], see Appendix A.5)

5.2 Generalization: Mid-Level Features Generalize Better to New Domains and Distant Goals

In the previous section we examined test-set reward, which is a determined by how well the agent learns on the training set how well that generalizes to the test set. We find that agents using mid-level features generalize well: both when the training and test buildings are drawn from the same distribution, and also when the test episodes are much longer than anything seen during training.

Train/Test in Different Buildings: To analyze generalization in more detail, we performed a study with a notably smaller training set (4 training buildings), as a smaller training set provides a larger opportunity for overfitting and makes it more obvious which methods are prone to it. As Fig. 6 (left) shows, both *scratch* and *mid-level* agents achieve similar reward on the training data, but agents using mid-level visual representations generalize better to new buildings (SPL of 0.65 vs. 0.46).

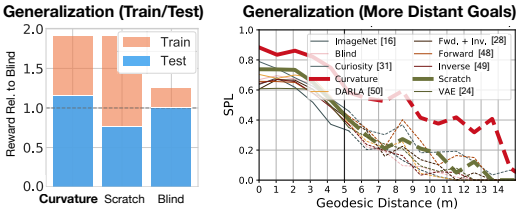

Figure 6: **Agents using mid-level vision generalize to new buildings and more distant goals.** *Left:* While both *curvature* and *scratch* achieve the same reward on the training set (4 buildings), agents using mid-level vision generalize far better than *scratch*, which performs worse than a *blind* agent (gray line). *Right:* Agents using *curvature* features retain performance on episodes longer than anything seen during training (right of black line); outperforming alternative approaches.

Generalizing to Longer Episodes: Another common definition of generalization rests on whether agents can extrapolate to novel situations unseen in the training set. Fig. 6 (right) shows that agents using mid-level vision are better able to extrapolate to episodes longer than those seen during training (under $5m \rightarrow$ over $5m$), as compared to agents trained from *scratch* (*scratch*: 0.70 SPL $\rightarrow$ 0.33 SPL vs. *curvature*: 0.84 SPL $\rightarrow$ 0.56 SPL). Agents using mid-level vision even significantly outperformed agents that explicitly model the environment dynamics (0.68 $\rightarrow$ 0.37 SPL), whereas a dynamics representation is expected to help with this type of generalization.

5.3 Sample Complexity: Mid-Level Visual Representations Result in Learning Faster

We find that agents using mid-level visual representations need less data. In our case, about an order of magnitude less. We report two different quantities to support this claim.

Performance by Number of Training Frames: We report the performance vs. number of training frames that the agent receives from the environment. Mid-level-vision-based agents achieve the final maximum reward of agents trained *tabula rasa* in 15% of the time (Fig. 7-left).

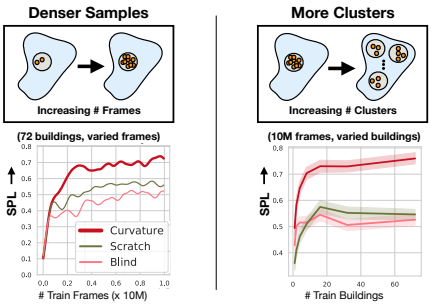

Figure 7: **Mid-level vision-based agents learn with fewer frames and fewer buildings.** *Left:* They achieve the final reward of agents trained *tabula rasa* in 15% of number of frames. Right: Agents with mid-level vision use fewer buildings (11%) to reach same performance as *scratch*.

Performance by Number of Sample Clusters (Buildings): One rarely noted but critical factor when measuring the number of training frames is that the frames are highly correlated. In our case, this implies two frames coming from the same building supply less diversity/visual learning value compared to two frames coming from two different buildings. Therefore, we also measure the performance as the number of sample clusters (buildings) increases, as opposed to just the frame count (Fig. 7, right). With only 11% (8/72) of the training buildings, agents using mid-level priors achieve the same (or better) reward as *scratch* does when trained on the 100% dataset .

5.4 No Universal Feature: Rank Reversal in Navigation and Exploration

Our results suggest there are not one or two canonical representations that consistently outperform all else. Instead, we find that the choice of representation depends upon the downstream task (Tab. 1, top). For example, the top-performing exploration agent used representations for *Distance Estimation*, perhaps because an effective explorer needs to identify large open spaces. In contrast, the top

navigation agent used representations for *Object Classification*—ostensibly because the agent needs to identify the target crate. Despite being top of their class on their preferred tasks, neither representation performed particularly well on the other task. Using 10 seeds per representation per task, this result was statistically significant (in both directions) at the $\alpha = 0.0005$ level.

The above pair is an example of rank reversal, which is actually a common phenomenon. It is so common that the feature rankings were effectively **uncorrelated** among our three tasks (Table 1, bottom), even though the three tasks seem superficially similar (all locomotion-based). Since the within-task ranks were consistent ($\rho = 0.87$ between environments), the choice of task was the primary determining factor for feature rank. Moreover, that choice determined whether whole families of related features would perform well. We found that semantic features were useful for navigation while geometric features were useful for exploration; and the semantic/geometric characterization was extremely predictive of final performance. We quantify the strong statistical significance in both Gibson and Doom using 120 pairwise significance tests in Appendix B.3.

Per-Task Top Feature (Reward)

Navigation	Exploration	Local Planning
1. Obj. Class. (5.90)	1. Distance (5.90)	1. 3D Keypts. (15.5)
2. Sem. Segm. (5.86)	2. Reshading (5.79)	2. Normals (15.1)
3. Curvature (4.74)	3. 2.5D Segm. (5.60)	3. Curvature (14.8)

Correlation (Spearman's ρ)

	Nav.	Exp.	Plan.
Nav.	-	-0.09	0.15
Exp.	-0.09	-	0.07
Plan.	0.15	0.07	-

Table 1: **Feature ranks are uncorrelated between tasks.** *Top:* Top 3 features per task (Gibson). Note that no feature is consistently on top. *Bottom:* Cells show Spearman's ρ between feature ranks on different tasks; no correlation was statistically significant.

5.5 When the Downstream Task is Unknown or Changing: Use a Set of Features

Since no single-objective representation could support all downstream tasks, we examined whether a **set** of single-objective representations could do better. We evaluated the Max-Coverage feature set for this purpose. Though it was derived in a way agnostic to the choice of task, we found that it performed nearly as well as the best feature for each task–*as if we had known which feature to select* (Fig. 9). Training agents for Local Planning in Habitat and using M-C features sets with 2, 3, or 4 features yielded SPLs of 0.730, 0.733, 0.749, respectively, while the best mid-level feature yielded 0.76 SPL. Using a set of features was crucial, as choosing the best task-agnostic *single* feature (*autoencoder*) dropped SPL to 0.58. *Scratch* and the SotA representation learning methods all scored under 0.57 SPL. We found similar behavior on when training agents for other tasks in the Gibson environment 9. The M-C feature set (with multiple features) also conferred the desirable emergent behaviors discussed in Sec. 5.2, and agents using this set exhibited some of the lowest rates of collision, acceleration, and jerk. We include the full discussion and formulation of the M-C feature set and detailed experiments (e.g. M-C vs. alternative feature sets) in Appendix B.4.

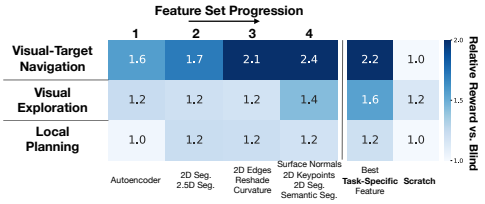

Figure 8: **Evaluation of max-coverage feature sets.** Each cell denotes the reward (relative to *blind*). The left 4 columns show agents trained with progressively larger max-coverage feature sets. The 4-feature set performs about as well as the best task-specific single feature—much better than the alternative approaches.

6 Conclusion

In this paper we showed that one of the primary challenges with learning visuomotor policies is how to represent the visual input. We showed raw pixels are unprocessed, high dimensional, noisy, and difficult to work with. We presented an approach for representing pixels using *mid-level visual representations* and demonstrated its utility in terms of improving final performance, boosting generalization, reducing sample complexity—significantly pushing the state-of-the-art. We also showed that the choice of representation generally depends on the final task. To this end we proposed a principled, task-robust method for computationally selecting a set of features, showing that the solver-selected sets outperformed state-of-the-art representations—simultaneously using an order of magnitude less data while achieving higher final performance.

Acknowledgements: This material is based upon work supported by ONR MURI (N00014-14-1-0671), Google Cloud, NSF (IIS-1763268), NVIDIA NGC beta, and TRI. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. URL http://dx.doi.org/10.1038/nature14236.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015. URL http://arxiv.org/abs/1504.00702.

[3] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, Jan 1992. ISSN 0899-7667. doi:10.1162/neco.1992.4.1.1.

[4] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[5] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

[6] F. Codevilla, A. López, V. Koltun, and A. Dosovitskiy. On offline evaluation of vision-based driving models. *CoRR*, abs/1809.04843, 2018. URL http://arxiv.org/abs/1809.04843.

[7] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018. URL http://arxiv.org/abs/1807.06757.

[8] J. W. Peirce. Understanding mid-level representations in visual processing. *Journal of Vision*, 15(7):5–5, 06 2015. ISSN 1534-7362. doi:10.1167/15.7.5. URL https://dx.doi.org/10.1167/15.7.5.

[9] F. Xia, A. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson Env: Real-world perception for embodied agents. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] O. Y. Z. E. W. B. J. J. S. J. L. V. K. J. M. D. P. Manolis Savva*, Abhishek Kadian* and D. Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019.

[11] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*, 2016.

[12] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[13] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[15] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014. URL http://arxiv.org/abs/1406.2283.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[17] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

[18] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. *Indoor Segmentation and Support Inference from RGBD Images*, pages 746–760. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33715-4. doi:10.1007/978-3-642-33715-4_54. URL https://doi.org/10.1007/978-3-642-33715-4_54.

[19] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696, Sept 2009. doi:10.1109/ICCVW.2009.5457637.

[20] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[21] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[22] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[23] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[24] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. ISSN 0036-8075. doi:10.1126/science.1127647. URL http://science.sciencemag.org/content/313/5786/504.

[26] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi:10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

[27] L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR 2017*, 2017.

[28] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

[29] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[30] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *CoRR*, abs/1611.01779, 2016. URL http://arxiv.org/abs/1611.01779.

[31] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017. URL http://arxiv.org/abs/1705.05363.

[32] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. S. Sawhney. Ten-fold improvement in visual odometry using landmark matching. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. doi:10.1109/ICCV.2007.4409062.

[33] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. D. Rodríguez, and D. Filliat. S-RL toolbox: Environments, datasets and evaluation metrics for state representation learning. *arXiv preprint arXiv:1809.09369*, 2018.

[34] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016. URL http://arxiv.org/abs/1606.07419.

[35] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *CoRR*, abs/1804.09364, 2018. URL http://arxiv.org/abs/1804.09364.

[36] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson. Visual representations for semantic target driven navigation. *CoRR*, abs/1805.06066, 2018. URL http://arxiv.org/abs/1805.06066.

[37] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. *CoRR*, abs/1810.06543, 2018. URL http://arxiv.org/abs/1810.06543.

[38] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[39] B. Zhou, P. Krähenbühl, and V. Koltun. Does computer vision matter for action? *arXiv e-prints*, art. arXiv:1905.12887, May 2019.

[40] R. Jamiruddin, A. O. Sari, J. Shabbir, and T. Anwer. Rgb-depth SLAM review. *CoRR*, abs/1805.07696, 2018. URL http://arxiv.org/abs/1805.07696.

[41] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi:10.1109/TRO.2015.2463671.

[42] R. Dragon and L. Van Gool. Ground plane estimation using a hidden markov model. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4026–4033. IEEE, 2014.

[43] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 354023957X.

[44] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[45] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[48] J. Munk, J. Kober, and R. Babuka. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4667–4673, Dec 2016. doi:10.1109/CDC.2016.7798980.

[49] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *CoRR*, abs/1612.07307, 2016. URL http://arxiv.org/abs/1612.07307.

[50] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. *arXiv e-prints*, art. arXiv:1707.08475, Jul 2017.

[51] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. *CoRR*, abs/1903.02531, 2019. URL http://arxiv.org/abs/1903.02531.

[52] A. Kumar, S. Gupta, D. Fouhey, S. Levine, and J. Malik. Visual memory for robust path following. In *Advances in Neural Information Processing Systems*, 2018.

[53] I. Gurobi Optimization. Gurobi optimizer reference manual, 2016. URL http://www.gurobi.com.

[54] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[55] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 00359246. URL http://www.jstor.org/stable/2346101.

Learning to Navigate Using Mid-Level Visual Priors

Appendix

The following items are provided in the appendices:

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

A Detailed Methodology

A.1 Baseline Description

In the main paper we described the only the most crucial baselines. We now provide descriptions for all baselines:

Tabula Rasa (Scratch) Learning: The most common approach, *tabula rasa* learning trains the agent from scratch. In this condition (sometimes called *scratch*), the agent receives the raw RGB image as input and uses a randomly initialized AtariNet [1] network (described in supplementary material).

Blind Intelligent Actor: The *blind* baseline is the same as *tabula rasa* except that the visual input is a fixed image and does not depend on the state of the environment. The *blind* agent is a particularly informative and crucial baseline since it indicates how much performance can be squeezed out of the nonvisual biases, correlations, and overall structure of the environment. For instance, in a narrow straight corridor which leads the agent to the target, there should be a small performance gap between *sighted* and *blind*.

Random Nonlinear Projections: this is identical to using *mid-level* features, except that the feature encoder is not pretrained but rather randomly initialized and then frozen. The policy then learns on top of this fixed nonlinear projection. This addresses the possibility that the ResNet architecture, not the offline perception task, is responsible the representations' success.

Pixels as Features: this is identical to using *mid-level* features, except that we downsample the input image to the same size as the features (16×16), apply a convolutional layer to match output sizes, and use it as the feature. This addresses whether the feature readout network could be an improvement over AtariNet [1].

Random Actions: this uniformly randomly samples from the action space. It calibrates how difficult the task is without any specialized method and determines how much can be obtained just from random chance.

Non-Learning Methods: For local planning, we compare against Simultaneous Localization and Mapping (SLAM) [40], a popular approach that is highly effective when depth information is accurate. However, a depth sensor is not available and we estimate depth using the same network that our mid-level representation uses for distance.

State-of-the-Art Representation Learning: We compare against several state-of-the-art representation-learning methods. They are not necessarily vision-centric, and they include dynamics-modeling [48, 49, 28], curiosity [31], DARLA [50], and ImageNet pretraining [16].

A.2 Max-Coverage Feature Set: Formulation

As we showed in the main paper, no single representation could be universal, necessitating the use of a set of representations. Employing a larger set maximizes the chance of having the correct representation for the downstream task available and in the set. However, a compact set is desirable since agents using larger sets need more data to train (for the same reason that training from raw pixels requires many samples). Therefore, we use a **Max-Coverage Feature Selector** that curates a compact subset of representations in order to ensure the *ideal* representation (encoder choice) is never too far away from one in the set.

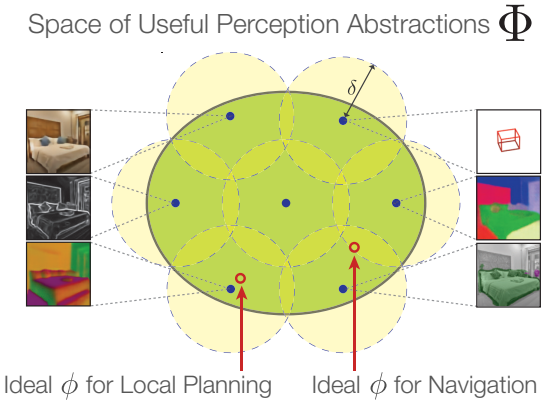

Figure 9: **Geometry of the feature set.** We select a covering set of features that minimizes the worst-case distance between the subset and the *ideal* task feature. By *Hypothesis III*, no single feature will suffice and a set is required. okay.

The question now becomes how to find the best compact set, shown in Figure 9. With a measure of distance between features, we can explicitly minimize the worst-case distance between the best

feature and our selected subset (the *perceptual risk* by finding a subset $X_\delta \subseteq \Phi = \{\phi_1, ..., \phi_m\}$ of size $|X_\delta| \leq k$ that is a δ-cover of Φ with the smallest possible δ. This is illustrated with a set of size 7 in Figure 9.

The task taxonomy method [12] defines exactly such a distance: a measure between perceptual tasks. Moreover, this measure is predictive of (indeed, derived from) transfer performance. Using this distance, minimizing worst-case transfer (*perceptual risk*) can be formulated as a sequence of Boolean Integer Programs (BIPs), parameterized by a boolean vector x indicating which features should be included in the set.

This section describes the full sequence Boolean Integer Program that yields, as its solution, the Max-Coverage Min-Distance Feature Set. It accounts for interactions between feeatures. For example, what if the combination of two features is much stronger than either feature separately? This section details a variant of the main BIP that handles these interactions. It selects a set of transfers which *may have one or more sources*.

The set of all transfers (edges), E, is indexed by i and each edge has the form $(\{s^i_1, \ldots, s^i_{m_i}\}, t^i)$ (for $\{s^i_1, \ldots, s^i_{m_i}\} \subset \mathcal{S}$ and $t^i \in \mathcal{T}$). We shall also index the target features $\oplus$ by j so that in this section, i indexes edges and j indexes target features. As in [12], we define operators returning target and sources of an edge:

$$\begin{aligned}\left(\{s^i_1, \ldots, s^i_{m_i}\}, t^i\right) &\xmapsto{sources} \{s^i_1, \ldots, s^i_{m_i}\} \\ \left(\{s^i_1, \ldots, s^i_{m_i}\}, t^i\right) &\xmapsto{target} t^i.\end{aligned}$$

We encode selecting a feature t to include in the set as including the transfer $\left(\{t\}, t\right)$.

The arguments of the problem are

1. δ, a given maximum covering distance
2. p_i, a measure of performance on a target from each of its transfers (i.e. the affinities from [12])
3. $x \in \mathbb{R}^{|E|+|\Phi|}$, a boolean variable indicating whether each transfer and each feature in our set

The BIP is parameterized by a vector x where each transfer and each feature in our set is represented by a binary variable; x indicates which nodes are selected for the covering set and a satisfying minimum-distance transfer from each unselected feature to one in the set. The canonical form for a BIP is:

$$\begin{aligned}&\text{minimize: } \mathbb{1}^T x\,, \\ &\text{subject to: } Ax \preceq b \text{ and } x \in \{0,1\}^{|E|+|\Phi|}\,.\end{aligned}$$

Each element c_i for a transfer is the product of the importance of its target task and its transfer performance:

$$c_i := r_{target(i)} \cdot p_i\,. \tag{1}$$

Hence, the *collective* performance on all targets is the summation of their individual AHP performance, p_i, weighted by the user specified importance, r_i.

Now we add three types of constraints via matrix A to enforce each feasible solution of the BIP instance corresponds to a valid subgraph for our transfer learning problem: *Constraint I:* no feature is too far away from the covering set, *Constraint II:* if a transfer is included in the subgraph, all of its source nodes/tasks must be included too, *Constraint III:* each target task has exactly one transfer in.

Constraint I: No feature is too far. We ensure that this by disallowing edges whose "distance" is too large. In our case, we use affinities (so that higher values indicate a better transfer). Therefore, we filter out edges with small affinities by redefining

$$E \triangleq \{\, e \in E \mid \text{affinity}(e) \leq \delta \,\}.$$

Constraint II: All necessary sources are present. For each row a_i in A we require $a_i \cdot x \leq b_i$, where

$$a_{i,k} \triangleq \begin{cases} |sources(i)| & \text{if } k = i \\ -1 & \text{if } (k - |E|) \in sources(i) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$b_i = 0. \tag{3}$$

Constraint II: One transfer per feature. Via two rows $a_{|E|+j}$ and $a_{|E|+j+1}$, we enforce that target j has exactly one transfer:

$$\begin{array}{llll} a_{|E|+j,i} & \triangleq & \mathbb{1}_{\{target(i)=j\}}, & b_{|E|+j} \triangleq +1, \\ a_{|E|+j+1,i} & \triangleq & -\mathbb{1}_{\{target(i)=j\}}, & b_{|E|+j+1} \triangleq -1. \end{array} \tag{4}$$

Those elements of A not explicitly defined above are set to 0. The problem is now a valid BIP and can be optimally solved in a fraction of a second [53].

The above program finds a (not necessarily unique) minimum-size covering set with a covering distance at most δ. Given that our feature set has only a finite number of distances, we can find the smallest δ by solving a sequence of these BIPs (e.g. with binary search). Since there are only m^2 distances, we can find the minimum δ with binary search, by solving $\mathcal{O}(log(m))$ BIPs. This takes under 5 seconds and the final boolean vector x specifies the feature set of size k that minimizes perceptual risk.

A.3 Downstream Active Tasks

This section contains detailed descriptions of our 3 locomotion-based active tasks. Visual depictions are shown in Fig. 10, and (environment-specific numbers are provided in supplementary material).

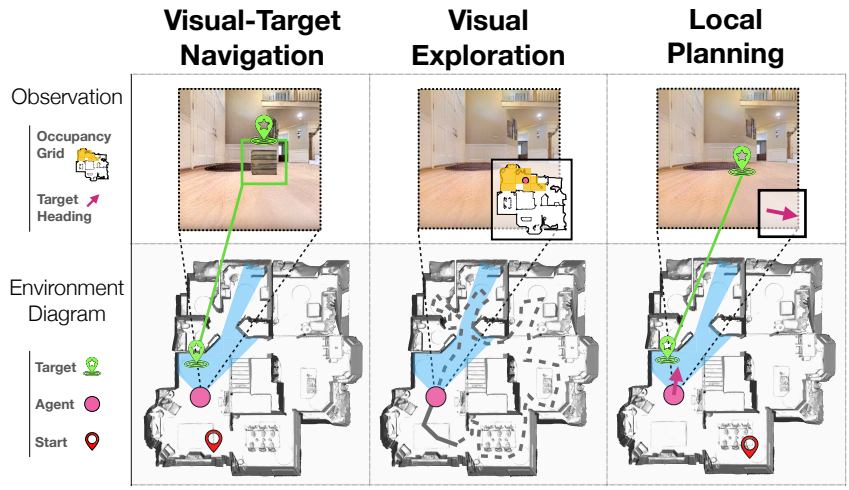

Figure 10: **Visual descriptions of the selected active tasks.** For each task, an example state of the environment is shown in the bottom row. The agent is shown as a pink dot, and its field of view is shown in light blue. Starting locaitons are shown with the red marker and a target, if it exists for the task, is shown with the green marker. The top row shows the corresponding sensory inputs, which always include some frame from the onboard RGB camera. Some tasks (local planning, visual exploration) include a additional nonvisual inputs, and these are shown in the bottom left corner of the RGB input (e.g. occupancy grid, target heading). Note that exploration uses only the revealed occupancy grid and no actual mesh boundaries.

Local Planning:

The agent must navigate to some target destination which is specified completely nonvisually (e.g. as a coordinate). This task is sometimes called "pointnav" or "point navigation".

The reward function is dense with several terms: a single positive value when the goal is reached (which also terminates the episode), a small positive value per timestep for progress towards the goal

(scaled change in distance to goal), a small negative value per timestep as a penalty for living and, for some environments, a larger negative value for obstacle collision. The reward function reflects desirable behavior of a *local* planner, which should skillfully maneuver its environment while taking the most efficient path to the goal. We implement a sparsified variant which only contains the one-time bonus and the living penalty (see the main paper for experiments)

At each timestep, the agent observed both the RGB camera images and a target vector $[r, \cos\theta, \sin\theta] \in \mathbb{R}^3$ where (r, θ) is the polar coordinates of the target in the agent coordinate system. In Habitat, we sometimes also provide a bitmap of past agent locations (to obviate the need for recurrence). The episode terminates either when the goal is reached or after a certain number of steps (usually 500). Sample frames are shown below and in the supplementary material.

Visual Exploration:

The agent is equipped with a myopic laser scanner and must use it to explore as much of the space as possible in a limited amount of time (usually 1000 timesteps). The environment is partitioned into $1m \times 1m$ occupancy cells and the reward at every timestep is the number of occupancy cells newly uncovered by the laser scanner.

The cells that the agent sees are calculated via a "myopic laser scanner" in the following way: first forming a point cloud using a narrow strip of the depth image from the midpoint of the image to the bottom, and then projecting this point cloud onto the ground plane. To determine what cells are newly uncovered, we compare this against previously unlocked cells.

At each timestep, the agent observed only the input frame from an onboard RGB camera and also the occupancy cells unlocked so far (provided as a bitmap image). The episode terminates automatically after a fixed number of agent steps.

Visual-Target Navigation:

In this task the agent must navigate to some object, specified visually. The agent must learn to both identify the target object as well as figure out how to navigate to it. The object (e.g. a wooden crate) remains fixed over training, but the agent and target locations are randomized.

The reward is sparse, with a large one-time positive bonus for reaching the object and a otherwise small negative penalty of living. At each timestep, the agent received the RGB input frame from the onboard RGB camera and nothing else. The episode terminates when the target is reached or after a certain number of steps (usually 500). A sample frame is shown below, and more frames are shown in the supplementary material.

A.4 Dictionary of Mid-Level Vision Objectives

Figure 11 contains a complete list of our studied vision objectives. A detailed description of each vision task can be found the Taskonomy [12] supplementary material (Section 14). We visualize some tasks in Figure 11 as well.

Mid-Level Vision Objectives ($\widetilde{\Phi}$)
Autoencoding
Classification, Semantic (1000-class)
Classification, Scene
Context Encoding (In-painting)
Content Prediction (Jigsaw)
Depth Estimation, Euclidean
Edge Detection, 2D
Edge Detection, 3D
Keypoint Detection, 2D
Keypoint Detection, 3D
Reshading
Room Layout Estimation
Segmentation, Unsupervised 2D
Segmentation, Unsupervised 2.5D
Segmentation, Semantic
Surface Normal Estimation
Vanishing Point Estimation

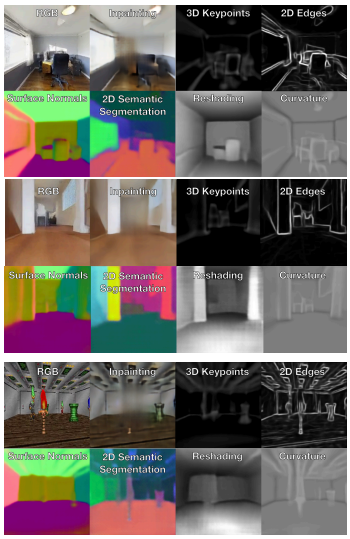

Figure 11: **Feature Bank with Sample Frames.** [Left] The mid-level features we used for all experiments. [Right] Frames and their respective mid-level features for Habitat, Gibson, and Doom for the top, middle, bottom frames respectively.

A.5 Metrics

Reward Relative to Blind RL results are typically communicated in terms of absolute reward. However, absolute reward values are uncalibrated and a high value for one task is not necessarily impressive in another. One way to calibrate rewards according to task difficulty is by comparing to a control that cannot access the state of the environment. Therefore, we propose the *reward relative to blind*:

$$RR_{\text{blind}} = \frac{r_{\text{treatment}} - r_{\text{min}}}{r_{\text{blind}} - r_{\text{min}}} \tag{5}$$

as a calibrated quantification. A *blind* agent always achieves a relative reward of 1, while a score > 1 indicates a relative improvement and score < 1 indicates this agent performs worse than a *blind* agent. We find this quantification particularly meaningful since we found agents trained from scratch often memorize the training buldings, performing no better than *blind* in the test setting (see Section 5.2 in main paper). For completeness, we provide the raw reward curves below in supplementary material.

Success Weighted by Path Length For local planning, we also use the metric of Success weighted by (normalized inverse) Path Length (SPL) introduced by [45]. The measure is defined as follows:

> We conduct N test episodes. In each episode, the agent is tasked with navigating to a goal. Let l_i be the shortest path distance from the agent's starting position to the goal in episode i and let p_i bet he length of the path actually taken by the agent. Let S_i be a binary indicator of success in episode i. We define a summary measure of the agent's navigation performance across the test set as follows:
>
> $$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)}.$$

A.6 PPO with Experience Replay

In this section we describe our off-policy PPO variant which decorrelates the samples within each batch and provides more stable, sample-efficient learning.

Off-Policy Policy Gradient

In the most general policy gradient setup, we attempt to maximize the objective function:

$$\mathbb{E}_{\tau \sim \pi_\theta(\tau)} \Big[\log \pi_\theta(a_t|s_t) \hat{A}_t \Big] \tag{6}$$

where $\hat{A}_t$ is the advantage function at timestep t (some sufficient statistic for the value of a policy at timestep t, in our experiments we choose the generalized advantage estimator [54]) and τ is a trajectory drawn from the current policy. The right side of the equation is our estimate of the objective using data sampled from the environment under the policy. Using importance sampling, we can approximate this objective by sampling from a different distribution over trajectories, and instead optimize:

$$\mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}(\tau)} \Big[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \Big] \tag{7}$$

Off-Policy PPO

In practice, if we were to directly optimize the objective in equation (2), this would lead to dramatically unstable gradient updates due to both the high variance of both the importance sampling ratio and $\hat{A}_t$ (in our actor-critic framework, we are learning $\hat{A}$ as the critic as training progresses, so in the beginning of training $\hat{A}$ is especially unstable).

Proximal policy optimization decreases variance by clipping the policy ratio, minimizing the surrogate objective:

$$\mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}(\tau)} \min \Big[r_t(\theta) \hat{A}_t, \text{clip}(1-\epsilon, 1+\epsilon, r_t(\theta)) \hat{A}_t \Big] \tag{8}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{9}$$

This objective acts as a first-order trust region, preventing large policy updates that drastically change the policy. For detailed theoretical justification of this objective, see the PPO paper [46]. In the original PPO paper, $\pi_{\theta_{\text{old}}}$ is only the distribution of on-policy trajectories from parallel workers. Since we are constrained to only one worker, sampling from on-policy trajectories only would lead to batches with highly correlated samples, leading to overfitting. Instead, we maintain a replay buffer, and draw multiple trajectories from the replay buffer at every update, treating the policy at the time when the trajectories were executed as $\pi_{\theta_{\text{old}}}$. To calculate the advantage, we reevaluate the advantage function during the update using the current critic on the states on the sampled trajectories from the replay buffer. The number of on-policy and off-policy trajectories that we sample under this formulation is a hyperparameter which we tune and report.

B Additional Experiments and Analysis

We include more results on the desired properties of midlevel representation, namely better performance and stronger generalization. We look at experiments across multiple environments and multiple downstream tasks.

B.1 Performance

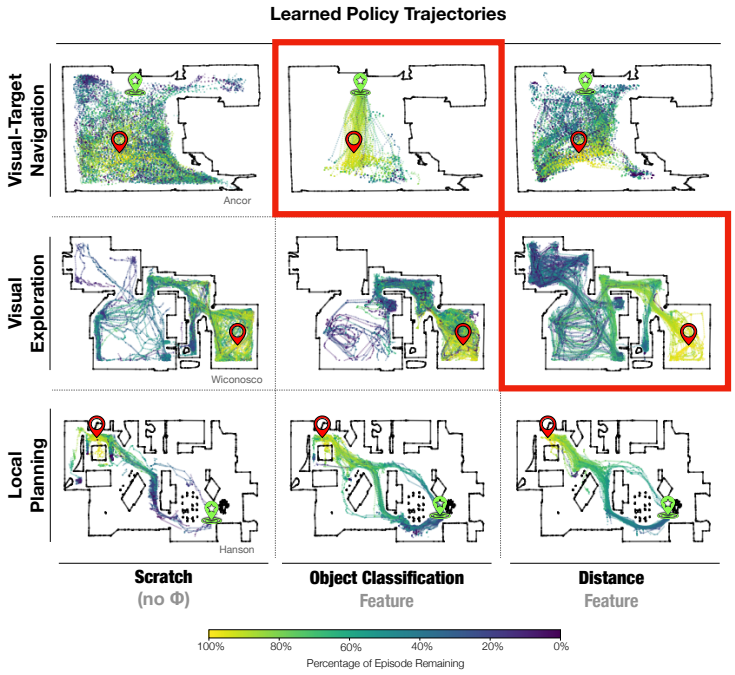

Figure 12: **Visualizations of the agents' trajectory.** While all approaches have reasonable trajectories for local planning, only certain features have desired trajectories for visual-target navigation and visual exploration.

Are the differences in the reward values meaningful? Do they led to different behaviors? In Figure B.1, we superimpose 100 evaluation trajectories in the three Gibson tasks of scratch and two different features, namely object classification and distance estimation. The scratch policy (left) completely fails to perform visual exploration or visual-target navigation with desirable trajectories, inefficiently wandering around the test space. The policy trained with object classification (middle) recognizes and converges on the navigation target (boxed), but fails to cover the entire space in exploration. This is perhaps due to the fact that semantic information is not as useful for exploration. Distance estimation features (right) help the agent cover nearly the entire space in exploration (boxed), but fail in navigation unless the agent is nearly on top of the target. This is perhaps due to the fact that geometric tasks fail to understand the semantics of the target.

Figure B.1 shows features that achieve a higher reward than scratch on a held-out set of validation buildings. Those features that do so with high confidence are highlighted in red For each task, some features outperform *tabula rasa* learning. We measure significance using the nonparametric Mann-Whitney U test, correcting for multiple comparisons by controlling the False Discovery Rate ($Q = 20\%$) with Benjamini-Hochberg [55]. These significance tests reveal that the probability of so many results being due to noise is < 0.002 per task ($< 10^{-6}$ after adding the seeds from the analysis in Section B.3).

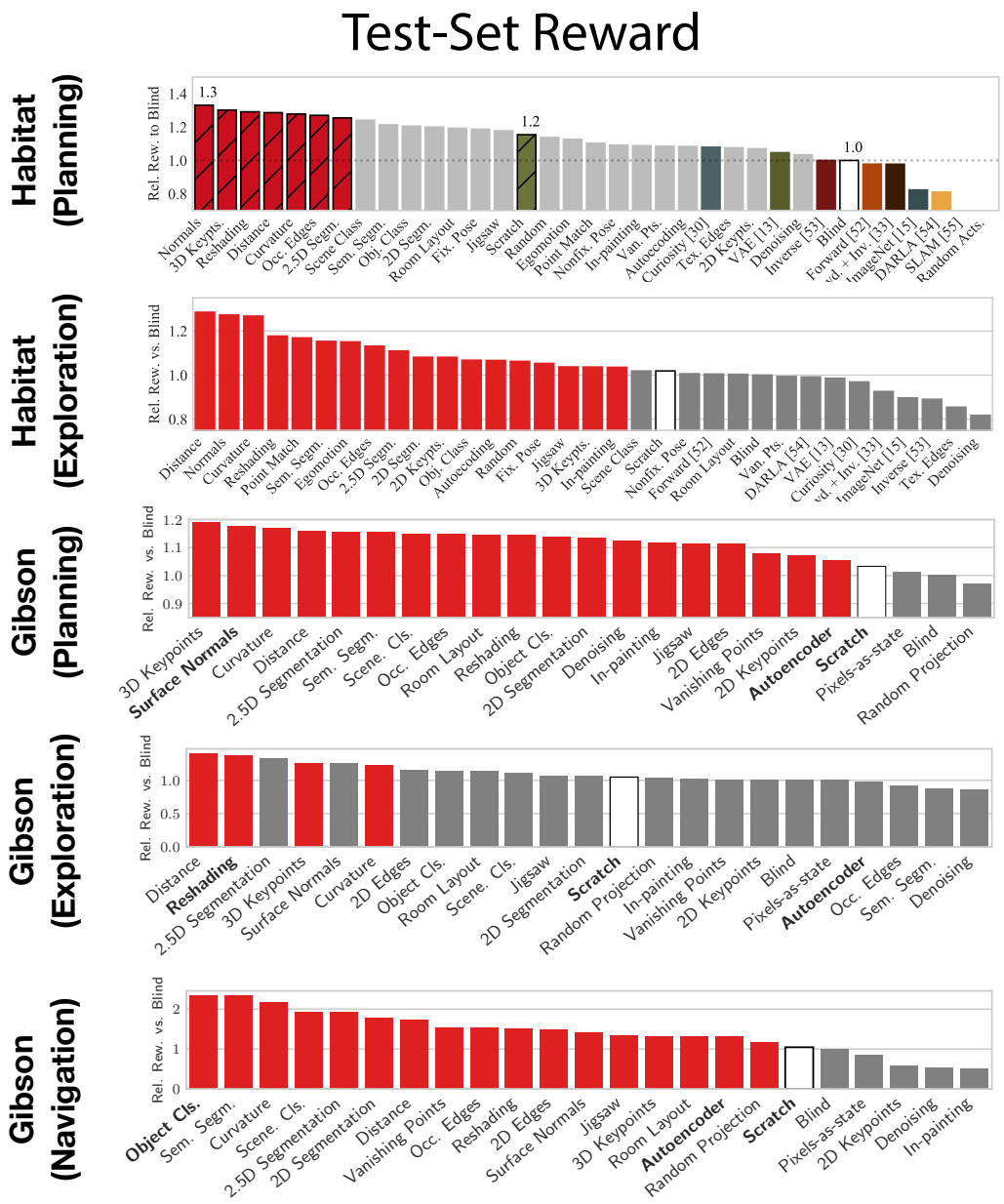

Figure 13: **Performance.** The reward relative to blind for all methods. Agents significantly better than *scratch* are shown in red.

B.2 Generalization

We investigate the generalization further in Gibson and Doom and show that our method generalizes better to new domains.

B.2.1 Generalization to Gibson Test Buildings

We find that for each of our tasks, several feature-based agent not only achieved higher final test performance than policies trained *tabula rasa* but also exhibited a smaller gap between training and testing performance. All agents exhibited some gap between training and test performance, but agents trained from scratch seem to overfit completely—rarely doing better than blind agents in

the test environment. The plots in Figure 14 show representative examples of this disparity over the course of training. Similarly, some common features like *Autoencoders* and *VAEs* have strong training curves that belie exceptionally weak test-time performance.

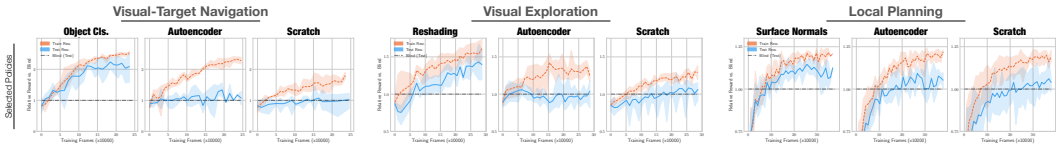

Figure 14: **Mid-level feature generalization.** The plots above show training and test performance of *scratch* vs. some selected features throughout training. For all tasks there is a significant gap between train/test performance for *scratch*, and a much smaller one for the best feature. This underscores the importance of separating the train and test environment in RL.

B.2.2 Generalization to Texture Variation in Doom

We also found that mid-level features were more robust than learning from scratch to changes in texture. While *scratch* achieves the highest final performance when the agent learns in a video game environment where there are many train textures that emulate the test textures, *scratch* fails to generalize when there is little or no variation in texture during training. On the other hand, feature-based agents were able to generalize even without texture randomization, as shown in Figure 15.

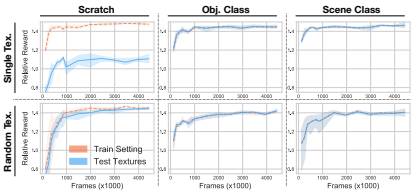

Figure 15: **Generalization in Doom Textures** In ViZDoom, feature-based agents (right two columns) generalize to new textures even when not exposed to texture variation in training (top row), while agents trained from scratch suffer a significant drop in performance (left, top).

B.3 Rank Reversal

In the main paper, we showed that geometric features were superior to semantic ones on exploration, but the opposite was true for visual navigation. We refer to this phenomenon as "rank reversal" - given any downstream task, the ranking of a feature is reversed on a different enough task. Here, we provide additional evidence of the ubiquity of rank reversal and offer evidence that there is no "universal" feature which maximizes performance for all active tasks. We start by looking at the rankings of features across tasks and follow with rigorous significance testing.

B.3.1 Feature Rankings in different tasks

In Figure 16, we plot the rankings of all the mid-level features at exploration and navigation. The lack of any feature existing in the bottom left corner of the plots shows that there is no feature which ranks the very highly for both tasks. The performance of the mid-level visual features is highly dependent on the properties of the downstream task. These results, reproduced both in Gibson and in Doom, substantiates the idea that no mid-level feature will maximize reward a priori and thus there is no universal feature. We verify this observation using significance testing in the following section.

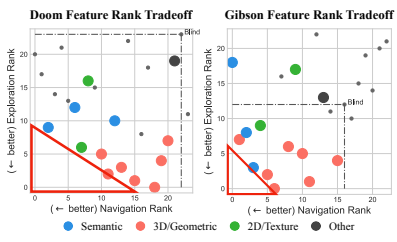

Figure 16: **Feature Ranks on Exploration and Navigation** Scatterplots of the rank of feature performance on navigation (x-axis) and exploration (y-axis) in both Gibson (right) and Doom (left). The absence of any feature in the bottom left corners implies that there is no single universal feature.

B.3.2 Analysis of Geometric and Semantic Features across tasks

Feature	Rew.	p-val.	i/m
Sem. Segm.	6.553	0.0000	0.04
Scene Cls.	5.969	0.0001	0.08
Obj. Cls.	4.212	0.0004	0.12
Reshading	0.525	0.7824	0.16
3D Keypts.	-0.196	0.9460	0.20
Distance	1.015	-	-

Feature	Rew.	p-val.	i/m
Distance	5.265	0.001	0.04
3D Keypts.	5.269	0.002	0.08
Reshading	5.072	0.011	0.12
Sem. Segm.	4.132	0.502	0.16
Scene Cls.	3.996	0.702	0.20
Obj. Cls.	4.151	-	-

Figure 17: **Rank-reversal (Gibson).** [Left] We test whether features are significantly better than distance estimation (the best feature for exploration) in navigation. [Right] We test whether features are significantly better than object classification (the best feature for navigation) in exploration. While within families (semantic, geometric), the differences are not significant, across families, the differences are significant.

We compare the top performing exploration feature (distance) to other features in navigation and visa-versa (top performing navigation feature to other features in exploration). For each feature and for each task, we train 10 agents from 10 random seeds and evaluate them at the end of training (420 updates for navigation, 480 updates for exploration). We evaluate the performance of each agent/seed combination by running the agent in the test environment for 100 random episodes. We then use a cluster-effect-adjusted Wilcoxon rank-sum test to test whether there is a significant difference in the reward-per-episode between features and show the results in Figure 17.

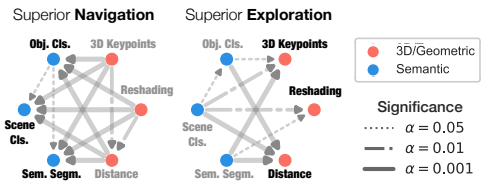

Figure 18: **Rank Reversal Significance Graphs** Results from pairwise tests evaluating 3 semantic and 3 geometric features on each task. For each task graph, arrows point towards the better-performing feature. Lack of an arrow indicates the performance difference was not statistically significant. Heavier arrowheads denote more significant results (lower α-level). The essentially complete bipartite structure in the graphs shows that navigation is characteristically semantic while exploration is geometric.

We find that within each family of tasks (e.g. semantic, geometric), there is no significant difference. However, a family performing well on one active task always has a significantly lower reward-per-episode compared to the other family on the other active task. For example, distance is statistically worse than any semantic task at navigation, cementing the rank reversal hypothesis.

In Figure 18, we perform pairwise significance tests between 3 well-performing mid-level features from each family. We find that the statistically different pairs come from different families. Additionally, we observe that while for navigation, arrows point heavily from geometric towards semantic tasks indicating significantly higher reward in the latter, the reverse is true for exploration (arrows point from semantic towards geometric tasks). Thus, there is no single task (or even family of tasks!) that consistently obtains significantly higher reward. We show results from Gibson but we came to the same conclusion in Doom.

B.4 Analysis of Max-Coverage Feature Set

This section contains additional analysis of the Max-Coverage Feature Set.

Max-Coverage Feature Set vs. Other Methods Figure 19 shows the findings (Success, SPL, Collisions, Acceleration, Jerk) for max-coverage feature set. The max-coverage feature set has similar performance with the best performing features while being task agnostic.

Max-Coverage Representation Set vs. Random Feature Set: How useful is the feature set proposed by the perception module? Will any feature set work? We randomly uniformly selected five feature sets and evaluated their performance on our active tasks. Table 2 shows that the solver-suggested feature set performs much better than the randomly selected sets on navigation, and comparably on the other two tasks. We hypothesize that the high performance of random features on exploration is due to a larger number of geometric-based tasks in our task dictionary, which tend

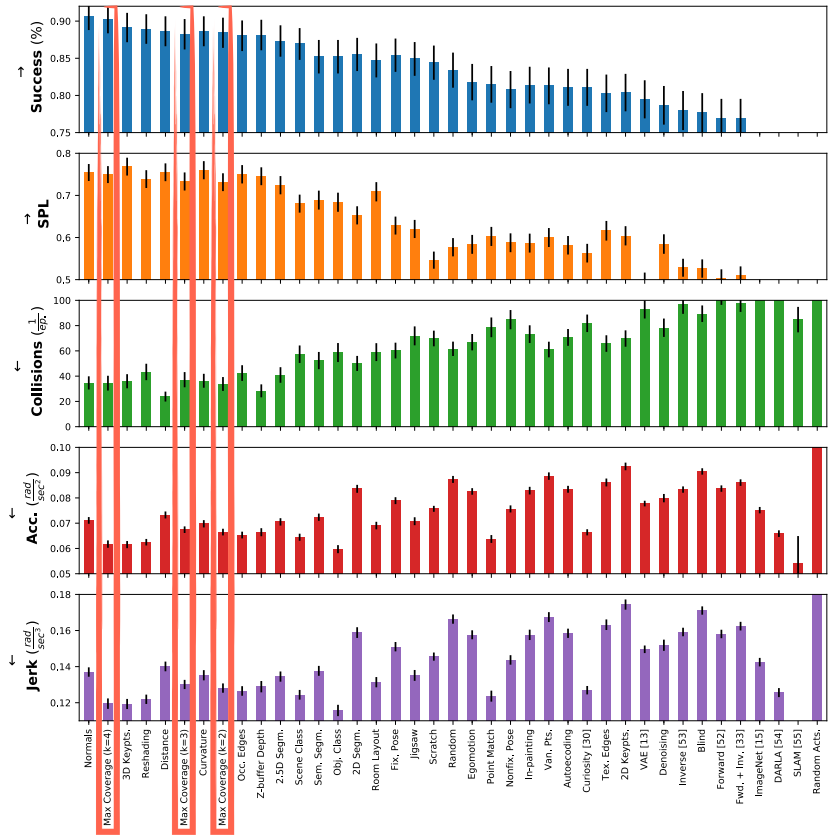

Figure 19: **Max-coverage feature sets exhibit strong performance and desirable behavior on Local Planning in Habitat.** The shows performance on the Habitat/Local Planning test set along a variety of dimensions. Features are ordered according to *test-set reward*. Max-coverage policies exhibit a strong combination of desirable properties, suggesting that they confer the benefits of mid-level vision.

Features	Navigation		Exploration		Planning	
	Ours	Rand.	*Ours*	Rand.	*Ours*	Rand.
2	**1.7**	1.5	1.2	**1.4**	1.2	1.2
3	**2.1**	1.8	1.2	**1.3**	1.2	1.2
4	**2.4**	1.9	**1.4**	1.3	1.2	1.2

Table 2: **Max Coverage Feature Set outperforms random feature set (Gibson).** We compared the Max-Coverage feature set to random feature sets, and the M-C feature set performs better than random feature sets. Each cell shows reward relative to blind.

to excel at the exploration task, while the relatively worse performance on navigation is due to the small number of semantic features in our dictionary. Our perception module ensures coverage of both kinds of features, which leads to good performance on both navigation and exploration.

It is notable that both our perception module and random sets of features outperform *tabula rasa* learning (and our other baselines) by a wide margin.