Pretrained Representations for Embodied AI

By

Alexander Sax

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Arthur J. Chick Professor Jitendra Malik, Co-chair
Assistant Professor (EPFL) Amir Zamir, Co-chair
Professor Alexei Efros
Professor Bruno Olshausen

Spring 2023

Pretrained Representations for Embodied AI

Abstract

Pretrained Representations for Embodied AI

By

Alexander Sax

Doctor of Philosophy in Computer Science

University of California, Berkeley

Arthur J. Chick Professor Jitendra Malik, Co-chair

Assistant Professor (EPFL) Amir Zamir, Co-chair

The world is messy and imperfect, unstructured and complex, and nonetheless we must still accomplish the basic behaviors necessary for survival. It is for this purpose, ecologically relevant behavior, that vision evolved 500-600 million years ago.

This thesis is about how learn representations of the visual world that are useful for the types of behaviors we might want an embodied AI system to do. In the first part of this thesis, we systematically study how bottlenecking visual inputs through different pretrained representations affects the ability of a robot to learn different atomic navigation skills (Chapter 2) and manipulation skills (Chapter 3) through trial-and-error. The main finding is that the appropriate pretrained representation greatly improves the sample efficiency for skill acquisition, and greatly improves the generalization of the learned skill. In the second part of the thesis, we use the lessons learned in order to improve the accuracy of the representations in a larger variety of contexts (indoors, outdoors, tabletop settings, and so on). In Chapter 4 we do this through adding cross-prediction consistency objectives. In Chapter 5 we do this by leveraging vast amounts of 3D data available on the internet and from a robot's prior experience.

The methods are primarily developed for the purpose of vision and action, but many of the ideas are general and could work for other sensory modalities and behaviors.

To my family: my parents, Barbara and Jon, and my sisters, Amelia and Addie.

Contents

Acknowledgments

I would like to express my sincere gratitude to both of my advisors, Jitendra Malik and Amir Zamir, for providing me the freedom to explore and the guidance to do so productively.

Thank you Jitendra Malik for teaching me to be a scientist. You showed me how to think about computer vision as an interdisciplinary scientific pursuit, and encouraged me to incorporate ideas from fields such as psychology, physiology, robotics, and more. Thanks for sharing with me your philosophies, and for helping me to develop my own. Like the time you sent me to the Lund sensory ecology bootcamp. You taught me that computer vision is not proof-based, and of the importance of reading both deeply and widely. Thank you for teaching me what makes a problem worthwhile and that research is the "art of the soluble". Your guidance in navigating the research community has been invaluable.

Thank you Amir Zamir for teaching me how to do research. You demonstrated how to break down complex problems into manageable pieces, and how to attack a manageable piece problem until it surrenders. You guided me in crafting clever experiments to test hypotheses, writing papers, preparing presentations, and much more. Thank you for our many discussions, brainstorming sessions, and collaborations–it is great fun to generate and analyze crazy ideas with you. And you showed me how to bring a machine learning skillset to ecological vision. You also have excellent pointers to historical papers. I have greatly benefited from (and been inspired by) your intense enthusiasm and creativity in everything you do.

Thank you Alyosha Efros for your phenomenal avuncular advising on all matters professional to personal to peripatetic (e.g. what cafes to visit while in Paris). I am grateful for your service on both my thesis and qualifying committees. Thank you to Angjoo Kanazawa for your advice throughout the years, for bringing such joy to whatever space you are in, and for serving on my thesis talk committee. And thank you also to Bruno Olshausen for serving on my qualifying and thesis committees. And thank you also to Michael Costello. Your mentorship throughout my years in high school helped to rekindle my love of learning. Your AP (and post-AP) stats classes first got me interested in statistical modeling and I draw a direct line from there to statistical modeling of behavior.

My heartfelt thanks to my incredible collaborators. Collaborating with you all has been a highlight of my PhD journey, and I look forward future collaborations and watching your careers flourish. Expressing my gratitude for everything you've taught me could fill several theses, but I would like to give special thanks to Jeffrey Zhang, Bryan Chen, and Ainaz Eftekhar. You each have remarkable research ability and have such a knack for implementing things quickly and correctly. Rediscovering research and Embodied AI alongside you has been an absolute pleasure. Thank you also to Bradley Emi, Francis (Gene) Lewis, Lerrel Pinto, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Iro Armeni, Nikhil Cheerla, Rohan Suri, Silvio Savarese, and Leonidas Guibas.

Thank you everyone at Berkeley, especially members of Jitendra's, Alyosha's, and Trevor's groups over the years: Anastasios, Andrea, Andrew, Antonio, Ashish, Ashvin, Daniel, Dave, Deepak, Devin, George, Georgia, Hang, Haozhi, Jasmine, Jathushan, Ilija, Kartik, Ke, Medhini, Neerja, Panna, Richard, Shiry, Shubham, Tete, Toru, Weicheng, Vickie, Vongani, Yossi, Young,

Zhe and others, for your guidance, support, discussions, and friendship. It has been a exciting and humbling experience learning to do research with such a kind and brilliant bunch.

Thank you the BAIR admin team: Angie, Ami, Roxana and Lena. For shielding me from Berkeley bureaucracy, providing us an excellent lab space in which to do research, for enabling BAIRs collaborations with industry, and for helping make many holidays feel festive with cards, crafting, and conversation.

Thank you to my friends. Especially, Hayley and Tobin. Hayley, our Dominion games over Zoom during the pandemic kept me sane. Tobin, thank you for encouraging me to even do a PhD in the first place.

Thank you to my family and friends for your kind support and encouragement over the years.

Chapter 1

Introduction

Every animal must perform certain tasks to survive and reproduce. These tasks generally fall into the "Four F's": *feeding*, *fleeing*, *fighting*, and *mating*. Natural selection targets these fundamental behaviors and favors effective behavioral rules.

A major breakthrough in the past half-century has been casting such selection as an optimization problem. It has been especially effective in machine learning and artificial intelligence, where advances in algorithms, computing power, and available data now enable the optimization of behavioral rules to achieve specific behaviors in given environments, using techniques like reinforcement learning [207], evolutionary algorithms [60], or "supervised learning" (learning from demonstrations [10]).

Early on in the study of behavior, Niko Tinbergen noted that the goals and methods of ethology (the study of behavior) and of cybernetics [220] (later "AI"[1]) were deeply intertwined [208]. AI could provide ethology with valuable models of behavior that allow researchers to probe how the behavior is connected to sensing and stimuli, underlying learning processes, morphology and environment. In AI, such behavioral rules that map sensory inputs to motor outputs are often called sensorimotor policies.

For example, a neural-network-based sensorimotor policy could be trained for navigation behavior (like foraging). AI's contribution here lies in the development of mechanistic models of behavior through selection, allowing researchers to interrogate the training setup and resulting sensorimotor policy: how do success rate and speed change if the policy has additional senses, like vision? What if the policy is forced to attend only to specific types of visual cues (like surface shapes, rather than color)? The idea is that sensing and learning are primarily valuable because they facilitate ecologically relevant behaviors.

This thesis systematically investigates the role of pretrained visual representations for some behaviors important to embodied AI, straddling the gap between computer vision and embodied behavior research. The research is divided into two main parts: (1) examining the types of

[1]Thank you, Amir, for pointing out that historically there were two camps: Norbert and Rosenblatt et al. in cybernetics and McCarthy and Minski et al. in AI, each with their own preferred terminology. Most of our methods today come from the first camp and most of our terminology from the second.

visual pretraining objectives and their impact on behavior, and (2) developing more robust visual representations that provide accurate and behaviorally-relevant perception.

What are "Pretrained Representaitons"

Over the past five decades, computer vision has aimed to estimate relevant aspects of the environment, such as surfaces, their properties, object recognition, detection, tracking, and camera localization (odometry). Initially, this estimation relied on hand-crafted algorithms that extracted short numerical descriptions of inputs (hand-crafted features [134, 199] of camera images, camera poses and intrinsics, etc.) before using those descriptions directly or with a small amount of machine learning (e.g., a small SVM or part-based model) learned on top of those hand-crafted features.

Nowadays, these features (also called representations) are learned from data using deep neural networks. And when these representations are trained on large and diverse data sets, the data-driven representations prove to be surprisingly general. For example, the representations learned for image classification on ImageNet can be adapted to do related tasks on different data (e.g. PASCAL object detection) [71, 189].

This technique, transfer learning, has become widespread in virtually every machine learning application. The use of such pretrained representations can substantially reduce the data needed to learn the downstream task while yielding a model that trains faster, achieves greater accuracy, and generalizes better to unseen data than would be possible by training the model from scratch using only the downstream data. Transfer learning has led to enormous successes in domains like visual recognition [71], speech recognition and synthesis [183], natural language processing [28], and protein folding [106].

Transfer learning depends on high-quality learned representations that result from large-scale training data, high-capacity network architectures, effective learning rules, and appropriate pretraining objectives. Success also depends powerfully on the match of the pretraining objective to the downstream application. As a result, transfer learning has been slower to take off in Embodied AI because it is difficult to find appropriate pretraining data, and researchers work with a variety of robot morphologies, sensor configurations and downstream tasks.

This thesis seeks to understand *when* transfer learning works for the types of tasks (skills) required of embodied agents. In particular, the thesis aims to identify what types of "pretraining" (objective, architecture, data, and learning algorithm) transfer well to different downstream embodied behaviors, like navigation and manipulation.

I. Pretrained Visual Representations and Embodied Behavior

The first part of the thesis presents a large-scale study of the types of visual pretraining objectives, and their utility for various atomic navigation and manipulation skills (learned with reinforcement learning and transfer learning, Chapters 2 and 3). One motivation for this study is practical: starting with an appropriate pretraining objective allows rapid skill acquisition. When starting from a pretrained representation, new behaviors like obstacle avoidance or object picking can be learned

from orders of magnitude less data than training from scratch. The final result is also a policy that generalizes much better to new environments and objects.

We find that no single pretrained representation is best for all navigation or manipulation skills, and the best pretrained representation depends on the desired skill. For example, when learning to pick up an object, it helps to begin with a visual system that already perceives surfaces and distances. But when searching for an object, it helps to have a visual system that can detect and recognize objects. Ideally, the pretrained representation provides the policy with a starting understanding of the world that simplifies the learning problem. However, the representation should be "aligned" with the intended behavior.

This research connects to a biological principle: animals should not sense more than they need to, since sensing comes with a cost. There is an firstly energetic cost because the neuronal tissue required to process the sensory input is expensive to maintain. The human brain, for example, accounts for about 20% of our overall energy expenditure, and about half of the human brain is devoted, directly or indirectly, to processing visual inputs. Then there is a developmental cost, as learning to process sensory inputs requires complex attentional patterns that are usually not innate. These patterns take time to develop, during which the animal is vulnerable. Such energetic and data constraints provide strong selection pressure for animals to devise minimal sufficient perception to accomplish a given behavior. Surprisingly, there has been little work on understanding sensory modalities and representations in terms of their ability to enable different behaviors with minimal learning time and minimal energy[2].

The tools of machine learning, simulation, and optimization are well-suited to uncover such relationships – to study and model the coupling between behavior, environment, morphology, and optimal perception under different functional constraints. This thesis demonstrates one way to do so, and in Chapters 2 and 3 we find results that match our biological understanding. In particular, we find distinct representations useful for different navigation skills: recognition for object-finding skills and geometry for obstacle-avoidance skills. This echoes the distinct "what" and "where" pathways in the primate visual system (the "dorsal" and "ventral" streams). We also find that for manipulation, representations of surface normals are much better than representations of depth, which matches findings that the human visual system is attuned to surfaces and shapes.

II. Pretraining Robust Visual Representations

One lesson from ethology is that even complex behaviors tend to be composed of responses that are highly-stereotyped and modulated by specific contextual stimuli. Problems like this–those that are narrowly-scoped, repeated, and dependent on a limited context–are an excellent fit for deep representation learning.

The second part of the thesis then focuses on developing better representations for Embodied AI, by defining appropriate objectives (scoping) and developing larger and more diverse datasets. The

[2]There is a small and vibrant field called "sensory ecology" that studies exactly this. Every two years, Lund University hosts a 2-week seminar for about 40 PhD students, mostly from that field. I was fortunate enough to get to attend, and I highly recommend it.

resulting representations provide accurate and useful information in a variety of contexts (indoors, outdoors, object close-ups, and scene-level views).

Chapter 4 develops methods to enforce consistency between different objectives by incorporating cross-prediction consistency constraints. Adding these constraints aids in generalization to out-of-distribution data and offers a natural way to estimate domain shift and perform adaptation. Chapter 5 outlines a pipeline to utilize new sources of 3D data (assets from the internet and 3D reconstructions from a robot's past experience) and to parametrically resample that data into large vision datasets capable of training accurate and robust visual representations. These improved vision models can then be used to enhance 3D reconstructions, leading to better resampled data, better vision models, and so on. Some follow-up works use the improved monocular estimators trained in Chapter 5 to achieve better 3D reconstruction [233, 251], closing the loop.

In summary, this thesis concentrates on identifying how *visual* representations enable downstream behavior and on developing more robust visual representations that provide accurate, behaviorally-relevant perception in various contexts. The focus on vision, rather than other senses, stems from my interest in vision and our *relatively* good understanding of visual perception, thanks to decades of computer vision research and centuries of research in optics and biology. Although vision is relatively well-understood, we are only beginning to comprehend other types of sensory inputs, such as touch, magnetoreception, or Wi-Fi sensing. The methods in this thesis are developed and validated for vision, but are not specific to it and similar approaches could be applied to understand how other representations of the environment and internal state of the agent affect the success and failure, as well as acquisition speeds, of downstream behavior.

Chapter 2

Visual Representations for Navigation

If the primary purpose of visual representations is to describe relevant aspects of visual inputs in ways that enable behavior, how well do different types of representations actually accomplish that goal? In this chapter, we examine various types of representations (estimating surface normals, depth, object masks, etc.) in terms of how well they support different types of navigation behaviors in indoor scenes (finding objects in a house, avoiding obstacles, exploring an unknown building). To achieve this, we use reinforcement learning (trial and error) to train policies that map these different representations to actions. With enough training data, the agents typically learn something reasonable in the training buildings. We find that some representations are better than others, in the sense that they enable agents to learn faster and generalize more effectively to new environments. The purpose of this chapter is to understand the connection between pretraining objectives and downstream behaviors, and one of our main findings is that the optimal type of perception depends on the specific behavior the agent needs to accomplish.

2.1 Introduction

The resurgence of deep reinforcement learning (RL) began with a number of nominal works, e.g. the Atari DQN paper [146] or *pixel-to-torque* [127], 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,

This chapter is based on joint work with Jeffrey Zhang, Bradley Emi, Amir Zamir, Silvio Savarese, Leonidas Guibas, and Jitendra Malik [182], and is presented much as it appeared in the CoRL 2019 proceedings.

i.e. "objects". These priors are basically *facts* of the world and incorporating them in learning is notably advantageous [68, 21]. 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, 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) [43, 6]. In this paper, we study how such standard vision objectives can be used within RL frameworks as mid-level visual representations [160], in order to train effective visuomotor policies.

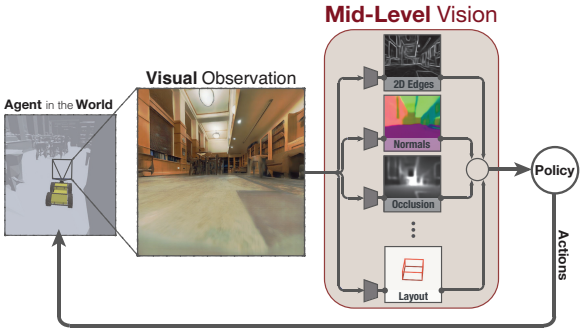


Figure 2.1: **Mid-level vision in an end-to-end framework for learning active robotic tasks.** We report significant advantages in *final performance*, *generalization*, and *sample efficiency* when using mid-level vision, especially compared to learning directly from raw pixels (i.e. bypassing the beige box).

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 [223], Habitat [181], *ViZDoom* [107]). Our mid-level vision comes from neural networks trained by existing vision techniques [238, 246, 49] 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.

2.2 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 stand-alone vision objectives, e.g. depth estimation [61], object classification [119], detection [72],

segmentation [193], pose estimation [245, 31], etc. The approaches use various levels of supervision [119, 154, 52, 24], 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) [108] or alternatives [89, 215, 140] 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 [105, 155] or by other related objectives [53, 159, 250, 169, 2]). 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 ([150, 149, 227]) or dense object descriptors for manipulation ([63]). Independently of our work, [247] 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 [100]), optical flow [153], or ground-plane estimation [56]. However, these usually use the output of the vision solutions in some analytic fashion [192] which requires the representations to be easy to analyze analytically. When the presesentation is not analytically well understood or in presence of noise (e.g. 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.

2.3 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

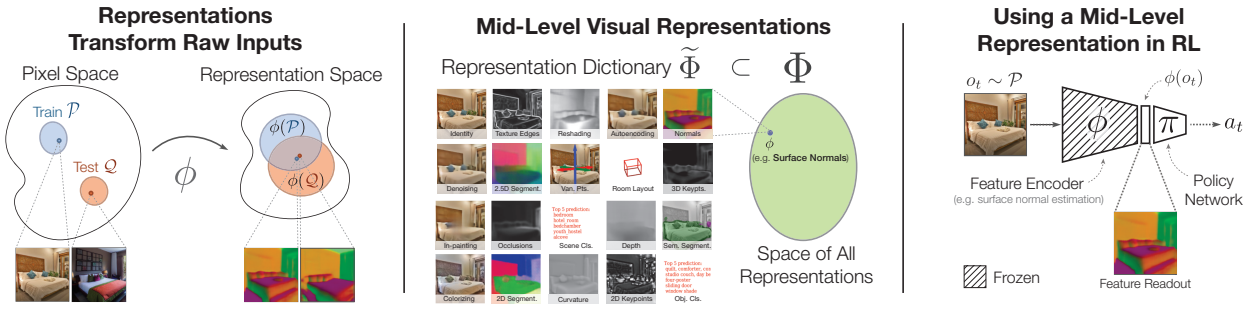


Figure 2.2: **Illustration of our approach.** *Left:* The job of a feature is to warp the input distribution, potentially making the train and test distributions look more similar to the agent. *Middle:* Visualizations for 19 of 24 mid-level vision objectives in our dictionary $\widetilde{\Phi}$. The 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.

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. 2.2, 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.

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

Why could a visual feature, e.g. image $\rightarrow$ surface 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. 2.2-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 [238] (see Fig. 2.2-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. 2.2-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.

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 generalizelreporting 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.

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. 2.5). We show this by demonstrating cases of *rank-reversal*lwhen 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.

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. 2.5, 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 Max-Coverage feature set in Appendix A.1.

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 [137]; we use the Habitat platform [181] with the Gibson [223] 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* [223] (a visually realistic simulator integrated with PyBullet dynamics and operating on Gibson dataset) and *ViZDoom* [107] (a 3D first-person game).

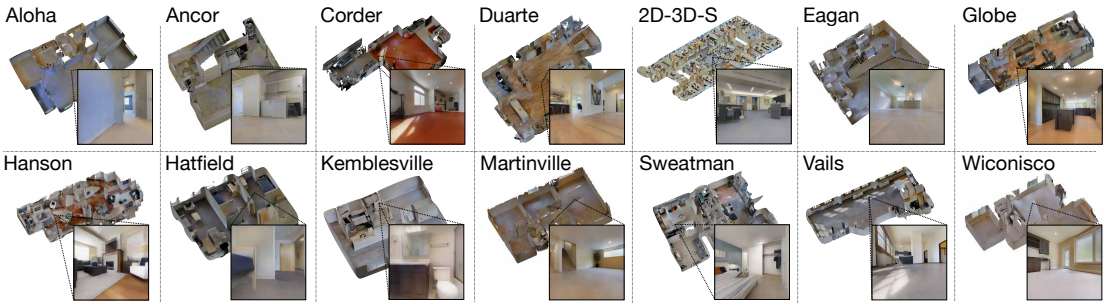


Figure 2.3: **Sample of 14 buildings from the Gibson [223] dataset.** One floor is shown from each space. Boxed images indicate sample observations from the Gibson [223] environment, while observations in Habitat [181] come directly from the (shown) mesh textures. We use 72 buildings for training and 14 for testing.

Train/Test Split: We train and test our agents in two disjoint sets of buildings (Fig. 2.3); 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.

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.1).

Local Planning (aka Point Goal [7, 181]): 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 $\mathcal{A} = \{\texttt{turn\_left}(10^\circ), \texttt{turn\_right}(10^\circ), \texttt{move\_forward}(0.25m)\}$.

Reinforcement Learning Algorithm

We use an off-policy variant of Proximal Policy Optimization [186] (PPO, details in Appendix A.1), 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 2.4). For our mid-level agents, we use the same hyperparameters that were optimized for scratch.

Mid-Level Representations

For our experiments, we used representations derived from one of 24 different computer vision objectives (Fig. 2.2). 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 [238] trained on a dataset of 4 million static images of indoor scenes [238]. All networks were trained with identical hyperparameters and using a ResNet-50 [82] encoder. For a full list of vision objectives and samples of the networks evaluated in our environments, see Appendix A.1.

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 [146] 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* [152, 190, 105], *curiosity* [159], *DARLA* [88], and *ImageNet pretraining* [119], enumerated in Figure 2.4.

Non-Learning: Methods such as SLAM [100] 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.

2.5 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 [181]. 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.

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. 2.4, left). This was tested in both Gibson [223] environment and Habitat [181], as shown in Fig 2.4. 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 A.2).

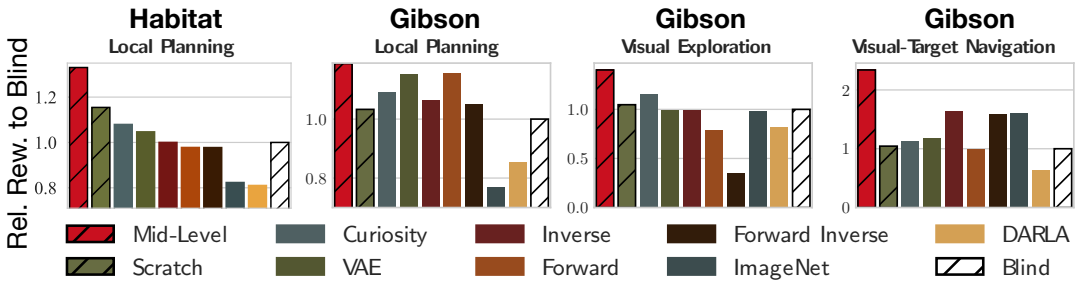


Figure 2.4: **Agents using mid-level vision had higher reward on the test set.** Each bar plot compares average test-set reward on a different task. On every task, agents using mid-level vision outperformed those learning *from scratch* or using alternative SotA representation-learning methods. Significance tests in Appendix A.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. 2.5 (a,b)[1]. Other papers [18] 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], see the project appendix and significantly outperformed SotA methods–*even when the SotA methods used dense reward and/or were explicitly engineered to handle sparse reward* (e.g. [159]: 0.56 SPL).

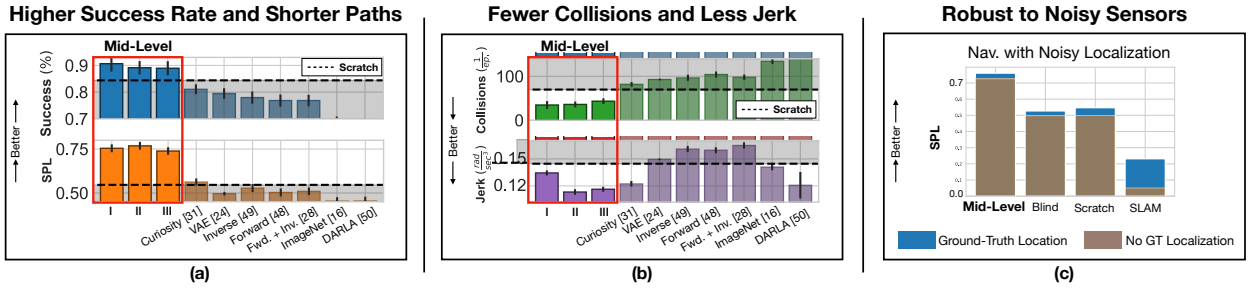


Figure 2.5: **Desirable emergent behavior and robustness under uncertainty.** *(a),(b):* The colorful bar charts show that agents using mid-level vision learn desirable behavior that is not explicitly coded for in the reward: they experience fewer collisions, less jerk [3], and achieve a higher success 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. 2.5 (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

[1]The top mid-level features (I, II, and III) were chosen based on reward (see Sec. 2.5). Error bars = 1 SE.

[2]SPL = Success Weighted by Path Length (as in [7]

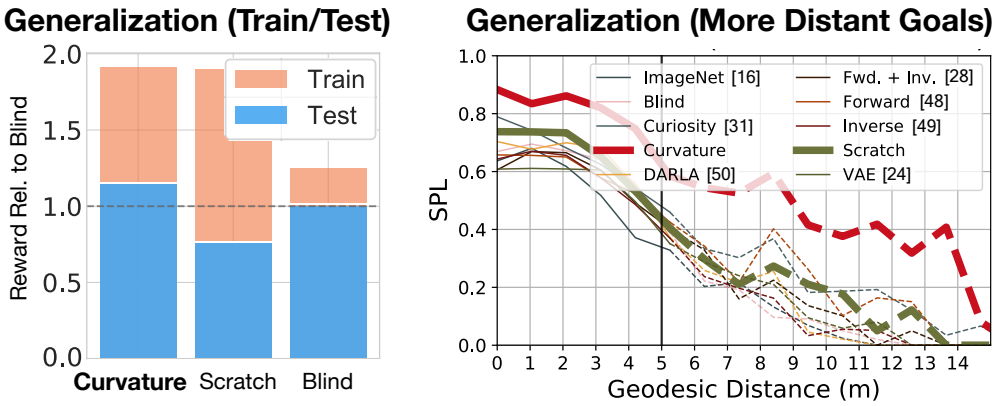


Figure 2.6: **Agents using mid-level vision generalize to new buildings and more distant goals.** *Left:* While both *curvature* and *scratch* achieve the same reward on the training set (4 buildings), agents using mid-level vision generalize far better than *scratch*, which performs worse than a *blind* 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.

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).

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. 2.6 (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).

Generalizing to Longer Episodes: Another common definition of generalization rests on whether agents can extrapolate to novel situations unseen in the training set. Fig. 2.6 (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

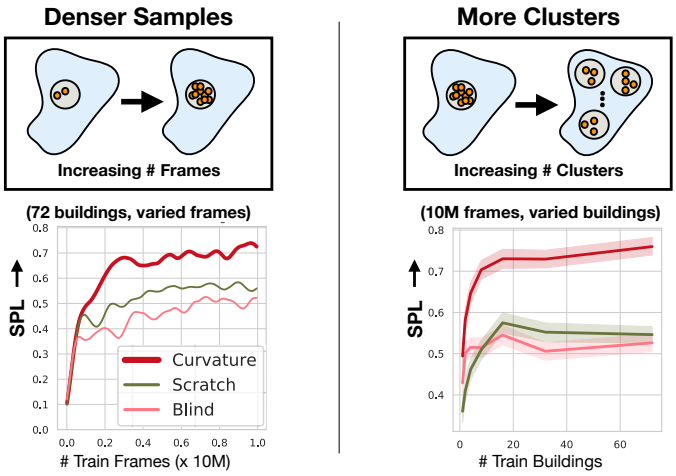


Figure 2.7: **Mid-level vision-based agents learn with fewer frames and fewer buildings.** *Left:* They achieve the final reward of agents trained *tabula rasa* in 15% of number of frames. Right: Agents with mid-level vision use fewer buildings (11%) to reach same performance as *scratch*.

SPL), whereas a dynamics representation is expected to help with this type of generalization.

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. 2.7-left).

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. 2.7, 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 .

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. 2.1, 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 2.1, 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 A.2.

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 2.1: **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.

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. 2.8). 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 2.8. The M-C feature set (with multiple features) also conferred the desirable emergent behaviors discussed in Sec. 2.5, 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 A.2.

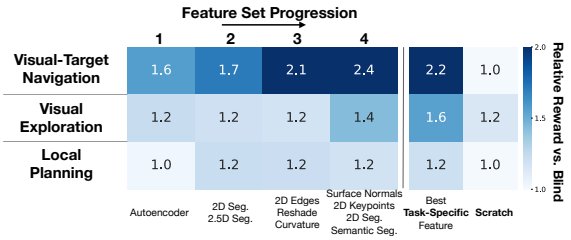


Figure 2.8: **Evaluation of max-coverage feature sets.** Each cell denotes the reward (relative to *blind*). The left 4 columns show agents trained with progressively larger max-coverage feature sets. The 4-feature set performs about as well as the best task-specific single feature—much better than the alternative approaches.

2.6 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.

Chapter 3

Representations for Manipulation and Real-World Navigation

In the previous chapter, we discovered that pretrained visual representations can significantly enhance both the sample efficiency and generalization of policies trained using RL for navigation in indoor environments. This served as a proof-of-concept. In this chapter, we expand on those findings in two ways:

1. Do the same representations that aided navigation in simulations also help navigation in real-world environments? (yes)

2. Do pretrained visual representations assist in manipulation skills, which require a deeper understanding of contacts? (yes)

The results in this chapter provide further breadth to the findings in Chapter 2. For navigation, we experiment with varying the amount of training data, the simulator's appearance and physics, the robot morphology, and the learning algorithm. Despite these modifications, the ranking of which representations work best for each skill remains largely consistent.

We then repeat the basic analysis for manipulation skills and find that pretrained visual representations are helpful here as well, once again improving sample efficiency and generalization.

3.1 Introduction

Over the past few years, impressive success stories such as [147, 128] have helped deep reinforcement learning (deep RL) make inroads into various fields. Deep RL from pixels, in particular, has drawn attention [132, 125, 195] as a unified method for training agents, but it requires that agents can access virtually unlimited data covering every possible input.

This chapter is based on joint work with Bryan Chen, Francis (Gene) E. Lewis, Iro Armeni, Silvio Savarese, Jitendra Malik, Amir Zamir, and Lerrel Pinto [40], and is presented much as it appeared in the CoRL 2020 proceedings.

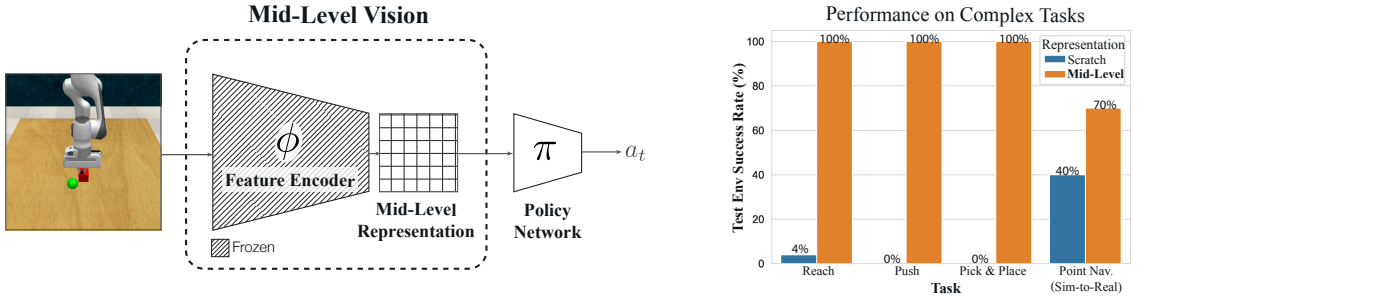


Figure 3.1: **Mid-level visual representations used for RL. Left:** A feature encoder trained for some mid-level objective provides representations to the agent. **Right:** Agents trained using these mid-level representations were able to generalize, without additional adaptation, to distribution shifts and deployment on physical robots.

In scenarios where agents can access large, but still finite amounts of data, the deep-RL-from-pixels approach proves hard to train, however, and the resulting policies are brittle in the real world [78]. Agents trained this way fail, sometimes spectacularly, under mild visual shifts relative to the training environment. For example, simply placing a water bottle in view of the robot [128], or operating in morning light instead of midday sun [206, 81] can be sufficient to completely degrade performance.

This brittleness reflects the fact that the policies have not captured the invariances (and equivariances) necessary to generalize and operate in the real world. The policies do not learn these invariances because there is usually no reason for them to do so. Biased or insufficient training data might permit spurious "cheating" shortcuts or make it easier to memorize features of the training environment rather than learn how to extract those features from the vast space of possible inputs. In any case, agents end up without the right *priors* about the world.

Computer vision provides us with tools that parse complex visual scenes and extract usable perceptual representations. In this paper, we study some of those representations used as a form of mid-level vision. That is, instead of training directly on raw pixels, we first extract representations driven by traditional computer vision objectives and use those as the input observations to RL instead (see Fig. 3.2-left). The networks which extract the mid-level visual representations are *asynchronously trained*, meaning that they can be trained independently and on a schedule different from the RL training. This approach has shown promise [182, 247, 229], especially in navigation contexts where agents using mid-level vision are able to generalize to unseen (simulated) buildings [182].

The main contribution of our study is experimental. We show that this approach scales to harder tasks than previously shown, even when control is nontrivial (e.g. manipulation with continuous control), or there are drastic domain shifts such as training in simulation and then deploying on physical robots with no additional training. Specifically, we test the following hypotheses:

a Do mid-level visual representations provide a useful way to incorporate invariances for "hard" tasks, when compared to training from scratch? (Answer: **Yes**)

b Do the representations simplify the learning problem (opening up the possibility to successfully train on harder problems that fail otherwise)? (Answer: **Yes**)

c Do the representations improve robustness to distribution shifts? (Answer: **Yes**: both sim-to-real and within simulation)

d How does the mid-level approach compare to other approaches for incorporating invariances? (Answer: **It scales significantly better**)

In summary, we present a large-scale study evaluating the effect of using invariances learned from computer vision objectives plugged into active RL frameworks. We find that the mid-level approach actually performs even better in the harder contexts, relative to alternatives, and provides an avenue for solving harder problems. We analyze this behavior in terms of training performance, generalization performance, and sample complexity. The mid-level approach was able to achieve a 100% success rate in certain test environments when alternatives approaches like learning-from-scratch, using *ground-truth low-dimensional state*, and domain randomization do not learn at all (0% train and 0% test, even after multiple hyperparameter sweeps).The mid-level approach trains faster than the alternatives, almost as fast as using ground truth low-dimensional state (when state succeeds). Further, we compare mid-level representations to other methods of learning invariances (domain randomization), and show that as the tasks become more difficult, domain randomization makes learning the task harder (100% train success $\rightarrow$ 70% with domain randomization, $4\% \rightarrow 20\%$ test) while mid-level simplifies it (near-perfect performance on the both train and test domains).

3.2 Related Works

Our study is connected to a range of relevant fields and we review the most important ones, within space constraints.

Computer Vision. Computer vision approaches are typically designed to solve stand-alone vision objectives such as depth estimation [61], object classification [119], detection [72], segmentation [193], pose estimation [245, 31]. While approaches may use various levels of supervision [119, 154, 52, 24, 162], the common characteristic across conventional computer vision methods is that they are trained and evaluated on static datasets collected for that stand-alone objective. In contrast, the perception of active agents is fundamentally in service of some downstream goal, and agents are evaluated on that goal in an *online* manner so that a single decision early on in an episode impacts the observations that will follow. In this paper we study how conventional computer vision objectives impact these downstream tasks, especially when the downstream tasks have strong temporal dependencies and domains different than those used for training the conventional vision approaches.

Representation/Feature Learning. The goal of representation learning is to encode observations in a way that provides benefits over using raw sensor data. One of the most popular approaches is based on Minimum Description Length (MDL) which suggests good representations are those which most compactly describe the data. This includes variational autoencoders [108] and alternatives [89, 215, 140] and has been applied to robotics (e.g. [62]). However, MDL

representations tend to be exceedingly sensitive to tiny domain shifts [182], an obervation that we also make. Likelihood-based approaches often try to predict the probability that two patches come from the same image–e.g. Constrastive Predictive Coding (CPC) [155] and variants [84, 36, 83, 75]. Dynamics-based approaches model the environment (e.g. by predicting the next state [105] or related objectives [53, 159, 250, 169, 2, 226]). Dynamics models have the possible advantage that they could be reused for planning. Dynamics are not necessarily visual and are sometimes specialized to the morphology or action space of the agent.

Mid-Level Vision in Other Contexts. Recent works have shown that mid-level visual representations can offer advantages in terms of generalization and sample complexity for downstream active tasks. Many works show one or a couple objectives for single tasks–e.g. a specific semantic representation for semantic driving ([150, 149, 227]) or dense object descriptors for manipulation ([63]). Outside of RL, [229] uses mid-level vision in conjunction with hard-coded grasping policies.

Recently, [182, 247] presented comprehensive studies in simulated navigation contexts using multiple objectives and multiple tasks. This many tasks/many objectives setup allows them to conclude that which objectives perform well depends on the task, which we also find. [182] then computationally derives a generic subset of mid-level representations that should perform as well as the best one in the whole set, but does not show when that best-possible feature is useful. Unlike previous studies, we demonstrate mid-level vision's ability to scale to harder tasks, to handle sim-to-real transfer, and we compare it to other approaches for incorporating invariances (e.g. domain randomization). By examining more complex domains, we find that we are able to train agents using mid-level features even in cases where other approaches fail.

Domain Randomization. One way of building agents with useful invariances is to make them learn it directly from data. [209, 180, 161, 221, 9] suggests using simulators to augment training with all the variation likely to be present at testing. However, training then takes longer and, in practice, varying more than one or two factors complicates the learning problem to the point that learning completely fails. Most importantly, domain randomization involves making the unrealistic assumption that we can enumerate all the invariances that are needed. We cannot, and for those which we can define, the corresponding variation is often difficult to build into a simulator (as evidenced by the numerous open problems in computer graphics). Our study shows that incorporating invariances via mid-level representations has multiple critical advantages over doing so via domain randomization. In particular, the mid-level representations can be trained asynchronously on large static datasets, making RL training simpler and more scalable. In contrast, domain randomization (unnecessarily) delays learning the invariances to be done in conjunction with RL training.

3.3 Methodology

Our goal is to study the utility of mid-level visual representations for performing downstream motor tasks. Concretely, motor tasks require mapping observation histories to actions. Agents are trained in a scenario $\mathcal{P}$ but evaluated on a test scenario $\mathcal{Q}$. The goal is to maximize agent's performance in a

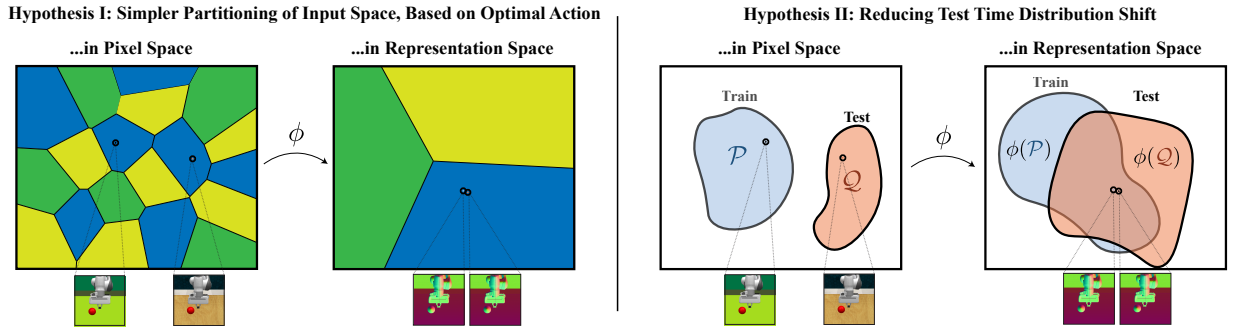


Figure 3.2: **Mid-level representations transform pixel inputs.** There are two major ways that invariances from mid-level representations could be useful for downstream tasks. **(I) Left:** The invariances simplify the decision boundaries for downstream tasks. In this case, we would expect (i) training on representations to be faster than training on pixels and (ii) to allow us to train agents for more difficult tasks. **(II) Right:** The invariances in representation space align the train and test distributions. In this case we would expect generalization performance to improve relative to training on pixels. In practice, we see behavior consistent with both hypotheses.

test setting ($\mathcal{Q}$) that it is related, *but not identical*, to the training distribution. For example, $\mathcal{Q}$ might be a physical robot in scenes contain unseen objects, while training on $\mathcal{P}$ took place in a simulator.

We assume access to a set of 'mid-level' functions, $\Phi = \{\phi_1, \ldots, \phi_m\}$, that transform raw sensory data into potentially useful mid-level representations. The goal is to determine whether agents using Φ could perform better in the test setting $\mathcal{Q}$ than if they never had access to Φ. In this paper, we show that when Φ is a set of mid-level functions, agents access to Φ improves generalization and generalize better final perfomance.

Using Mid-Level Representations in Active Contexts

Why might a mid-level feature (e.g. image $\rightarrow$ surface normals) aid in downstream tasks, compared to end-to-end learning? This is an important question as preprocessing observations with ϕ might potentially discard information (e.g. color information in surface normals). Good representations preserve important information while discarding spurious details, providing "invariances" that make the train and test set more similar ($\phi(\mathcal{P}) \approx \phi(\mathcal{Q})$). When the train and test set become similar, improving performance on the train set generally improves test-time performance, too.

Really good representations also simplify training by using $\phi(\mathcal{P})$ instead of $\mathcal{P}$. By throwing away unimportant information and providing easily decodable outputs (e.g. linearly separable), great representations can reduce sample complexity and boost performance even on the training set, relative to learning *tabula rasa*. These ideas are illustrated in Figure 3.2.

We use representations derived from neural networks trained, offline, for computer vision objectives. Figure 3.1 shows how we use these networks in our active framework. While training agents from mid-level representations we freeze the mid-level network (ϕ) and use it to transform each observed image o_t into a summary statistic $\phi(o_t)$ that is then provided to the agent instead of pixels. Only the policy network is updated during agent training. This has the advantage that

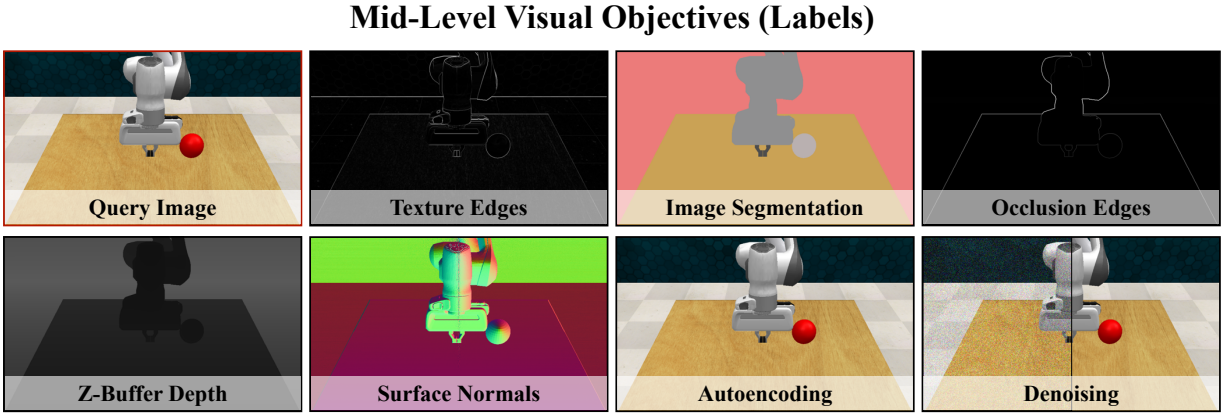


Figure 3.3: **Sample labels for mid-level visual objectives in the RLBench environment.** The objectives cover various modes of computer vision tasks including 2D, 3D, and semantic tasks.

we can reuse the same mid-level features for new tasks without degrading the performance of already-learned policies. Freezing the representations is a naïve approach and the features could be updated (e.g. via [173, 239]), but even with the naïve approach the representations yield notable benefits.

Studied Mid-Level Representations In this paper, we study representations from neural networks that were trained, offline, each for a specific vision objective from [238]. 7 of them are visualized in (Figure 3.3) and they cover various common modes of computer vision objectives: from texture-based (e.g. denoising), to 3D pixel-level (e.g. depth estimation), to semantic (e.g. image segmentation). As the networks were trained on data from indoor scenes significantly different than our manipulation setting, we fine-tune the networks, offline, with images from our simulator. We study the effects of domain shift and feature robustness in Section 3.4. All networks were trained with identical hyperparameters and using a ResNet-50 [82] encoder. For a full list of vision objectives and samples of the networks evaluated in our environments, see the supplementary.

Metrics

Final performance: We report train and test performance at the end of training. That not all agents converge to the same training reward may be due to the limitations of current learning approaches, data and compute; an agent might still fail to learn *anything* even when it is given every available resource. We find this occurs regularly for complex tasks, even when agents can access the true low-dimensional environment state (see Section 3.4). Specific mid-level features might also be ill-suited to particular tasks; discarding necessary information and further limiting final performance.

Generalization: We break down final performance in terms of generalization from train to various test sets. We report both test-set performance with no shift (evaluated in the same simulator as training), under mild domain shift (with various colors swapped out at test-time), and under more

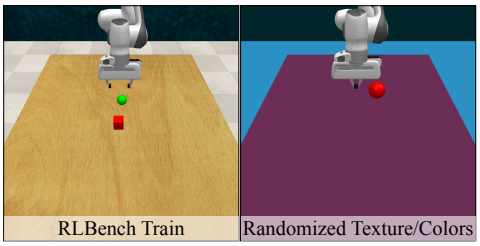


Figure 3.4: **Train vs. test observations in RLBench. Left:** The default environment for the Pick + Place task. The goal is colored green, object in red. **Right:** Sample observation from the test environment showing held-out randomized textures.

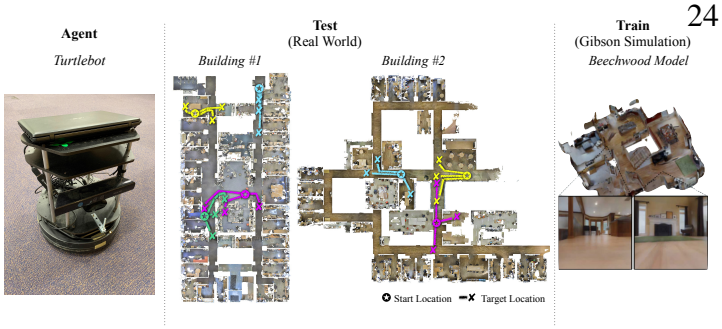


Figure 3.5: **Zero-Shot Visual Sim-to-Real. Right:** We train policies in a single building in simulation. **Middle:** We then directly test them in novel real world buildings, where agents have no prior knowledge of the building and no adaption period. **Left:** The TurtleBot uses only an RGB camera for vision and an IMU for localization, No depth/LiDAR sensors are used.

drastic domain shift (unseen objects or under zero-shot sim-to-real transfer).

Sample efficiency: We examine whether agents equipped with mid-level vision can learn faster than a comparable agent learning *tabula rasa* (i.e. with raw vision, but no priors about the world). We report this in terms of the number of interactions with the environment.

Manipulation with Continuous Control

To study mid-level vision on harder tasks, we test the mid-level representations against baselines in three different manipulation tasks requiring continuous control from vision. This section describes the setup at a high level, and complete details for each subsection can be found in the supplementary.

Tasks: We use three common manipulation tasks: *Reach*, *Push*, and *Pick + Place* with sparse rewards (0 if within ϵ of the target, -1 otherwise). All tasks terminate episodes after 50 timesteps. Brief descriptions of each task are below, and full details for each of the the tasks are in the supplementary.

Reach: The agent must move the end effector to a random target position marked by a sphere.

Push: The agent must push a cube to a random target position marked by a sphere.

Pick and Place: The agent must pick up a cube and move it to a target position marked by a sphere. Both object and target locations are randomized.

Observation Space: For each task agents receive visual input from a single camera. Agents trained from pixels receive $64\times64\times3$ RGB images while mid-level approaches receive $16\times16\times8$ latent features. Except for the state baselines, no other information (including proprioception) is supplied.

Action Space: For all tasks, actions are XYZ+gripper values in [-1, 1] specifying end-effector deltas for a position controller. Gripper open/close is calculated via thresholding.

Environment: We adopt the RLBench environment [98] which is built on PyRep [97] and CoppeliaSim [177]. RLBench is suited for vision-based manipulation and offers more visually

realistic observations compared to popular environments such as OpenAI Gym [163]. We use the Franka Emika Panda arm in our experiments.

Train/Test Split Agents are trained in the default RLBench tabletop environment shown in Figure 3.4. All policies are tested on a test set of held-out flat color textures that replace the table, floor, and/or background (Fig 3.4, right). We also show experiments with domain randomization during training (no texture overlap with the test set) or with held-out objects during testing.

Learning Algorithm. We train all agents using TD3 [65] and HER [8] using hyperparameters optimized for the *tabula rasa* baseline. Mid-level agents used the learning rate optimized for scratch in [182], but we ran additional hyperparameters sweeps for agents trained from scratch on the Reach and Pick+Place tasks. Because all hyperparameter searches were done using the from-scratch approach and the settings then applied everywhere, this setup should be biased *against* mid-level vision.

We followed the guidelines for training laid out in [8]. For some tasks we could not get the from-scratch approach to train using sparse rewards and we had to add additional reward shaping. The shaped dense rewards often help the mid-level approach too (see supplementary). While in the main paper we present the best approach for the pixel-based methods, we only show the sparse-trained policies for mid-level based agents since sparse rewards are usually easier to define in practice.

Generalization to the Real World

In cases where collecting real-world data makes training policies prohibitively expensive, the plentiful and cheap data from simulation provides a path to train policies that can then be deployed in the real world. However, the domain shift between simulators and the real world means that policies trained this way usually fail to generalize. We study this sim-to-real capability for agents trained with mid-level representations via RL.

As we are primarily focused on perception, we focus on navigation contexts where the visual gap is responsible for the primary domain shift. Manipulation is usually non-quasistatic and, in practice, simulators will trade off accuracy (simulating all contact forces) for simulation throughput, resulting in a large sim-to-real gap for environment dynamics. Simulators can more accurately model the simpler dynamics of navigation environments and physical robots have good low-level locomotion controllers that can handle any discrepancies. Even so, navigation-based sim-to-real is highly non-trivial from a vision perspective. Any number of discrepancies between simulated images and real images could cause the agent to fail to generalize; potential discrepancies include lighting variations, stitching artifacts and semantic distribution complexity.

We test sim-to-real generalization for the *point navigation* task from the CVPR19 Habitat Challenge. The task requires an agent to navigate to a target position (specified by coordinates) using visual observations and the agent's onboard odometry. Actions are discrete `{forward, pivot right, pivot left}` and episodes cap at 400 timesteps. We use policies from [182] trained with the same architecture as our manipulation tasks but using PPO [186] in a *single* building in the Gibson environment [223] and are tested in *different* (unseen) real-world buildings. We

evaluate the mid-level vision based policies trained in [182] on a Turtlebot with Kobuki base and a Microsoft Kinect camera. Full details are provided in the supplementary.

State Representation Baselines

To address confounding factors, we compare against several controls. We describe the most important ones here and defer remaining descriptions to the supplementary.

Tabula Rasa Learning (aka *from scratch*): This is the most common approach for end-to-end learning. The agent receives the raw RGB image as input and uses a randomly initialized AtariNet [147] architecture that is updated during training, along with the policy architecture.

Blind: The blind agent is the same as *scratch* but instead of an RGB image, receives a constant zero tensor. This shows how much can be learned by just exploiting the biases of the task.

State: In this setting the agent has access to the complete environment state: joint positions, the goal centroid and, if applicable, the object center. Since objects are always the same, this should be sufficient.

3.4 Results

Given that no two setups outside of simulation will be exactly the same, building in invariances into visuomotor policies could be helpful for bridging the gap. The rest of this section covers experiments that dissect whether such invariances are necessary, finding that incorporating them offers notable advantages in terms of final performance, generalization, and sample complexity. We then compare two main methods of building in invariances: either co-learning them during training (via domain randomization) or asyncronously learning them in different stages (via mid-level representations). As the agent needs to learn more invariances, we show that the co-learning becomes complicated and learning can collapse. We find that the mid-level approach scales and performs better.

Final Performance

We find that agents trained using mid-level representations achieve significantly higher success rates than agents learning from pixels, especially in harder tasks such as *Pick + Place*. The mid-level agents performed much better than scratch in the test environment, as shown in Fig. 3.6. Consistent with Hypothesis II (*mid-level representations simplify training*) they performed better during training, too, especially for harder tasks.[1]

[1]The mid-level policies in Fig. 3.6 used vision networks fine-tuned from [238]. When they were retrained for longer (and from scratch), performance for the mid-level agents improved further and they roughly matched the performance of the *state* oracle (Section 3.4). Results shown in Table 3.1.

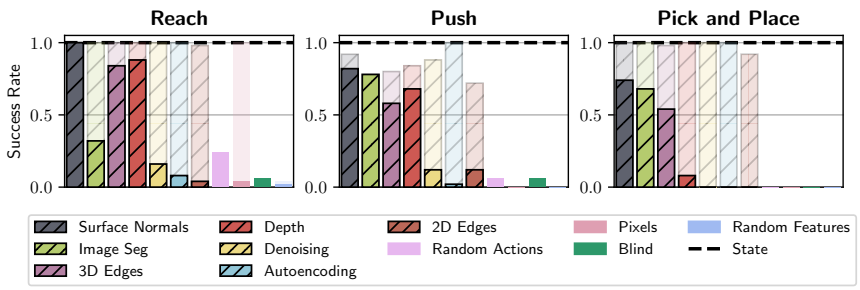


Figure 3.6: **Final performance on manipulation tasks.** Translucent bars indicate training performance; opaque bars show performance on the test set. Hatched bars indicate agents are trained with mid-level representations.

We found the above results despite the fact that the hyperparameters were optimized for the scratch baseline. In general, we found that the mid-level agents were less sensitive to choices of hyperparameters, and that the same or similar hyperparameters worked across multiple architectures, downstream learning algorithms, and simulators. They also did not require reward shaping.

Generalization

Comparison to generic state for unseen objects. In order to test whether mid-level representations could provide an easily decodable representation that enables both learning and generalization to unseen objects, we train agents for Pick+Place with 10 red, procedurally generated objects of different shapes. In contrast to the standard environment which only used a red cube, encoding the salient parts of the environment is now more complicated; we represent the object shape by its mesh vertex positions centered by the object centroid. In the more complex training environment, the state-based agent gets a 2% success rate during training (0% on unseen test), shown in Fig. 3.7. In contrast, using mid-level representations (normals), the agent has a 96% success rate on the training objects (90% on the unseen objects, 96% training objects colored green, and 88% on unseen green objects).

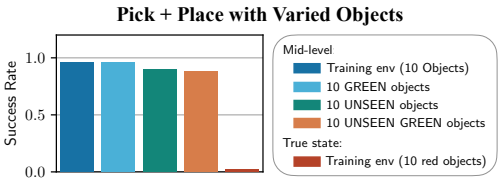


Figure 3.7: **Generalization to unseen objects.** In all tested environments, the mid-level policy trained to pick+place varied objects outperformed an agent using true state.

Invariance Method		Success Rate			
Mid-Level Representation	**Domain Rand.**	**Reach** Train	Test	**Pick + Place** Train	Test
None (*Scratch*)	No	100%	4%	0%	0%
Image Segmentation		100%	88%	100%	92%
Surface Normals		**100%**	**100%**	**100%**	**100%**
None (*Scratch*)	Yes	70%	20%	0%	0%
Image Segmentation		100%	100%	98%	98%
Surface Normals		**100%**	**100%**	**100%**	**100%**

Table 3.1: **Policies trained via different invariance-learning approaches.** The mid-level approach scales better to harder tasks, compared to *tabula rasa* or domain randomization.

Learning invariances with mid-level representations vs. domain randomization. Table 3.1 compares the performance of agents trained with different methods of incorporating invariances.

Agents using the asyncronous (mid-level) approach perform better across train and all test environments. In particular, when tested on colors not seen in the training, mid-level vision has a success rate of 100% versus 20% when using pixels with domain randomization. The domain randomization approach trained from scratch also showed signs of learning collapse (100% $\rightarrow$ 70% success rate) as the randomization made the learning problem more difficult, a problem also found in [99, 4].

Sim-to-real transfer. Agents using mid-level vision generalize to new axes of variation not present during training and across large gaps of the simulator vs. physical world. After training in a single simulated building, we test in 24 scenarios in two unseen buildings in the real world. Scenarios vary significantly in length, complexity, and visual characteristics (mean length 5.24m; variance 3.65m^2). In 594 evaluation runs and over 13 hours of execution time, we found that agents trained from scratch achieved an SPL [7] of 0.319 and a completion rate of 0.4 in the test environment, which was not significantly different than blind agents, as shown in Figure 3.9. Agents using mid-level features achieved a significantly higher SPL of 0.608 and a completion rate of 0.7. The use of the best features therefore provides a 90.6% increase in SPL and 75.3% increase in completion rate over scratch. Because we did no fine-tuning here, we could evaluate a slightly larger set of features here than for the simulation experiments. A full description of the experiment is available in the supplementary, and we provide videos from the agents' onboard cameras during the sim-to-real test episodes on our website.

Sample Efficiency

Training agents with mid-level representations dramatically increases sample efficiency, allowing for policies to be trained on difficult tasks using sparse rewards. Across all tasks, agents using mid-level representations converge within 2x the number of steps required to train from state (e.g. 450k steps vs 250k in Pick + Place). This is several times faster than learning from scratch (when it is even possible for scratch to learn anything—even with reward engineering, the from-scratch approach never succeeds on the test set). We provide train/test curves in the supplementary.

Analysis of Features for Downstream Tasks

Relationship between mid-level objectives and downstream tasks.

Given that mid-level objectives are typically defined irrespective of any downstream task, we ask whether representations that perform better on their objective also perform better on downstream tasks. Figure 3.8 shows the downstream performance of agents trained using surface normal and image segmentations features when those features are from various checkpoints during training. We

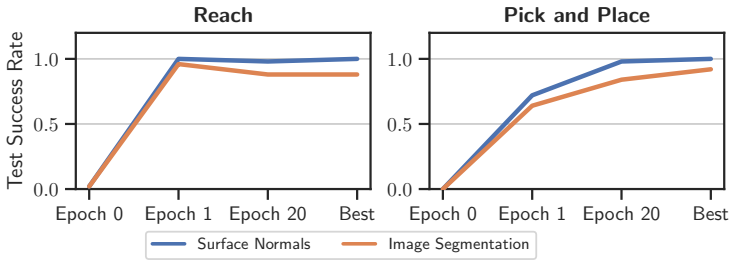


Figure 3.8: **Performance on mid-level vs. downstream tasks.** If a feature initially performed well on the downstream task, a better version further improved downstream performance.

found that generally, when the feature is useful for the task, the two performances are correlated (both features on pick+place).

Feature rank stability in different environments.

We found that within each task, feature order was notably stable, even across large sim-to-real domain gaps: the Spearman's rank correlation between the feature rankings found by testing in the real world vs. testing in simulation (Gibson) was ρ=0.77. [182] compares across simulators, but not in a zero-shot manner, finding that feature rank correlation was ρ=0.88 between Habitat [137] and Gibson when mid-level agents were trained in the respective environments. This inter-environment correlation is not simply because some features are more useful than others (i.e. the same features are always useful): feature ranking in [182] was uncorrelated across tasks.

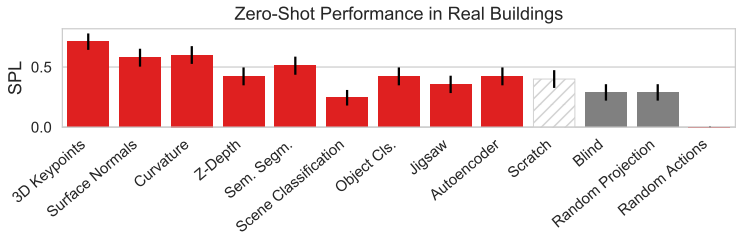


Figure 3.9: **Performance in simulation vs. zero-shot transfer to the real world.** Agents using mid-level representations (red) significantly outperform agents trained from scratch. Agents are ordered on the x-axis by descending performance in simulation. Those trained from scratch do not significantly outperform blind agents in the real environment (standard error shown in chart). Performance for top-performing features (e.g. curvature) was about the same in simulation as the on a physical robot [182].

3.5 Conclusion

We presented a comprehensive experimental study on using mid-level visual representations with RL to train agents to complete complex tasks over varying levels of distribution shifts.

High-level takeaway: Mid-level representations simplify training, improve generalization, and aid training speed. **Mid-level representations should be the preferred input to policies**, especially for harder tasks, where they are more viable than alternative methods of invariance learning.

Lessons learned: First, we found that agents trained using mid-level vision could be successfully trained for harder tasks than possible when training from scratch or using domain randomization. Training was less sensitive to the choice of hyperparameters, and training speed improved. Overall, these results ere consistent with the hypothesis that mid-level representations can simplify the input space and make the learning problem easier.

Second, we found the agents trained using mid-level representations were significantly more robust to domain shifts than agents trained from scratch (and also those trained using domain randomization). We showed this in simulation on multiple manipulation tasks under multiple types of domain shift (new objects, textures). We also showed this in the sim-to-real setting, successfully deploying mid-level-based simulator trained policies in unseen real-world buildings. The robust generalization is consistent with our second hypothesis, that mid-level representations also align training and test distributions to improve test-time performance. While approaches for solving mid-level objectives are generally less sensitive than methods trained for robotics using RL, they

are still susceptible to domain shift. Improvements to methods for approximating these individual objectives would probably carry good knock-on effects for agents trained using mid-level vision.

Third, which features performed well depended on the choice of task, but not so much on the environment used for training. The advantage of picking a good feature (vs. training from scratch) grew as tasks became more difficult, underscoring the importance of picking a good feature. While we did not explore how to pick a generic set of features, this would be an important avenue and an [182] has proposed an initial (computationally-derived example). Without the dependence on task, these features would be expected to work well in most environments.

Future work: That mid-level features work well in harder contexts suggests that we are ready to "close the loop" between features and downstream tasks. Features are currently defined irrespective of any downstream task. Choosing a representative set of benchmark tasks that "cover" downstream robotic tasks would make it possible to choose new computer vision objectives that better benefit downstream motor tasks. This same strategy of a defining of benchmark tasks has made it possible to design ever-better architectures (e.g. ResNets in computer vision and Transformers in language) that work well for most perception tasks in that domain.

Chapter 4

Cross-Task Consistency

In earlier chapters, we utilized representations that were trained independently, leading to the discovery that no single representation was optimal for all downstream behaviors. This is a common scenario in most machine learning systems, which often involve several intermediate predictions used for a subsequent task. For instance, segmentation masks are typically used as inputs for another system responsible for image editing or decision-making based on the mask input. Self-driving cars frequently estimate numerous quantities that are eventually employed for downstream control. So, how can we train these systems to ensure consistency among different predictions and maximize their usefulness for downstream applications? This chapter explores one approach to achieving this goal, particularly by employing a clever technique to model the joint predicted distribution conditioned on the inputs using pairwise consistency between predictions.

4.1 Introduction

What is consistency: suppose an object detector detects a ball in a particular region of an image, while a depth estimator returns a flat surface for the same region. This presents an issue – at least one of them has to be wrong, because they are *inconsistent*. More concretely, the first prediction domain (objects) and the second prediction domain (depth) are not independent and consequently enforce some constraints on each other, often referred to as *consistency constraints*.

Why is it important to incorporate consistency in learning: first, desired learning tasks are usually predictions of different aspects of one underlying reality (the scene that underlies an image). Hence inconsistency among predictions implies contradiction and is inherently undesirable. Second, consistency constraints are informative and can be used to better fit the data or lower the sample complexity. Also, they may reduce the tendency of neural networks to learn "surface statistics" (superficial cues) [102], by enforcing constraints rooted in different physical or geometric rules. This

This chapter is based on joint work with Amir Zamir, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas Guibas [236], and is presented much as it appeared in the CVPR 2020 proceedings.

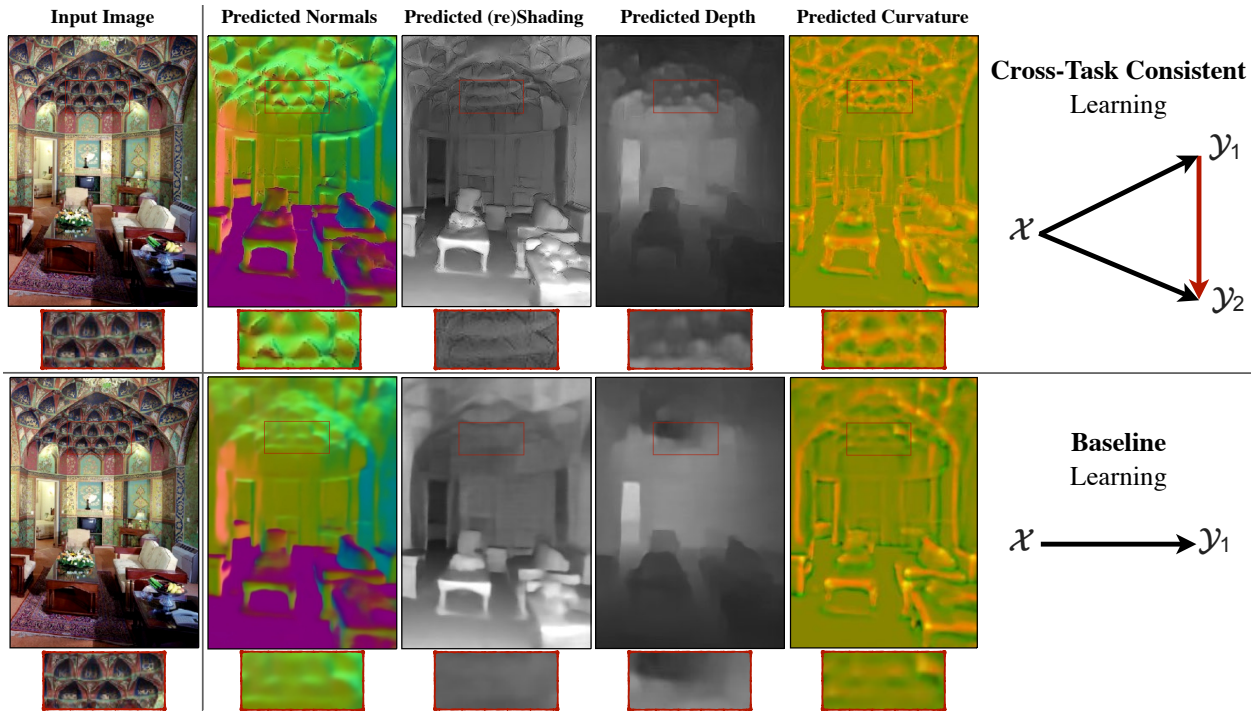


Figure 4.1: **Cross-Task Consistent Learning.** The predictions made for different tasks out of one image are expected to be *consistent*, as the underlying scene is the same. This is exemplified by a challenging query and four sample predictions out of it. We propose a general method for learning utilizing data-driven cross-task consistency constraints. The lower and upper rows show the results of the baseline (independent learning) and learning with consistency, which yields higher quality and more consistent predictions. Red boxes provide magnifications. [Best seen on screen]

is empirically supported by the improved generalization of models when trained with consistency constraints (Sec. 4.5).

How can we design a learning system that makes consistent predictions: this paper proposes a method which, given an arbitrary dictionary of tasks, augments the learning objective with explicit constraints for cross-task consistency. The constraints are learned from data rather than apriori given relationships.[1] This makes the method applicable to any pairs of tasks as long as they are not statistically independent; *even if their analytical relationship is unknown, hard to program, or non-differentiable*. The primary concept behind the method is 'inference-path invariance'. That is, the result of inferring an output domain from an input domain should be the same, regardless of the intermediate domains mediating the inference (e.g., RGB→normals and RGB→depth→normals and RGB→shading→normals are expected to yield the same normals result). When inference paths with the same endpoints, but different intermediate domains, yield similar results, this implies the intermediate domain predictions did not conflict as far as the output was concerned. We apply this concept over paths in a graph of tasks, where the nodes and edges are prediction domains

[1]For instance, it is not necessary to encode that surface normals are the 3D derivative of depth or occlusion edges are discontinuities in depth.

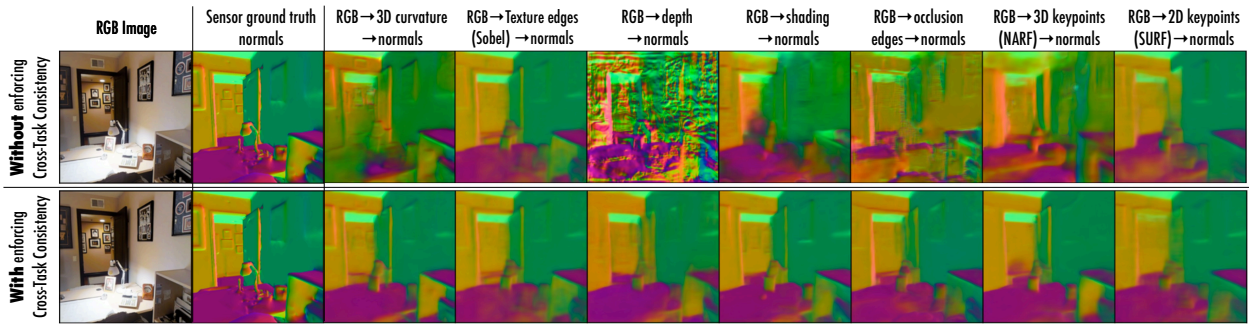


Figure 4.2: **Impact of disregarding cross-task consistency in learning**, illustrated using surface normals domain. Each subfigure shows the results of predicting surface normals out of the prediction of an intermediate domain; using the notation $x \rightarrow y_1 \rightarrow y_2$, here x is RGB image, y_2 is surface normals, and each column represents a different y_1. The **upper row** demonstrates the normals are noisy and dissimilar when cross-task consistency is not incorporated in learning of $x \rightarrow y_1$ networks. Whereas enforcing consistency when learning $x \rightarrow y_1$ results in more consistent and better normals (the **lower row**). We will show this causes the predictions for the intermediate domains themselves to be more accurate and consistent. More examples available in supplementary material. The *Consistency Energy* (Sec. 4.4) captures the variance among predictions in each row.

and neural network mappings between them, respectively (Fig. 4.3(d)). Satisfying this invariance constraint over *all* paths in the graph ensures the predictions for all domains are in global cross-task agreement.[2]

To make the associated large optimization job manageable, we reduce the problem to a 'separable' one, devise a tractable training schedule, and use a 'perceptual loss' based formulation. The last enables mitigating residual errors in networks and potential ill-posed/one-to-many mappings between domains; this is crucial as one may not be able to always infer one domain from another with certainty (Sec. 4.3).

Interactive visualizations, trained models, code, and a live demo are available at: http://consistency.epfl.ch/.

4.2 Related Work

The concept of consistency and methods for enforcing it are related to various topics, including structured prediction, graphical models [117], functional maps [157], and certain topics in vector calculus and differential topology [76]. We review the most relevant ones in context of computer vision.

Utilizing consistency: Various consistency constraints have been commonly found beneficial across different fields, e.g., in language as 'back-translation' [26, 15, 124, 58] or in vision over the

[2]inference-path invariance was inspired by **Conservative Vector Fields** in vector calculus and physics that are (at a high level) fields in which integration along *different paths yield the same results, as long as their endpoints are the same* [76]. Many key concepts in physics are 'conservative', e.g., gravitational force: the work done against gravity when moving between two points is independent of the path taken.

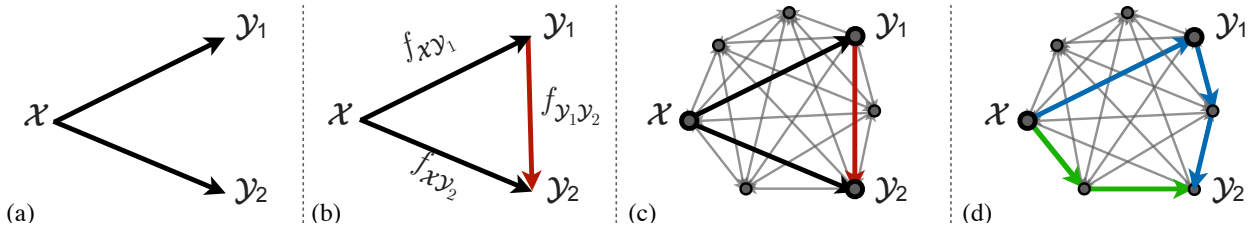


Figure 4.3: **Enforcing Cross-Task Consistency: (a)** shows the typical multitask setup where predictions $\mathcal{x}\rightarrow\mathcal{y}_1$ and $\mathcal{x}\rightarrow\mathcal{y}_2$ are trained without a notation of consistency. **(b)** depicts the elementary *triangle consistency constraint* where the prediction $\mathcal{x}\rightarrow\mathcal{y}_1$ is enforced to be consistent with $\mathcal{x}\rightarrow\mathcal{y}_2$ using a function that relates $\mathcal{y}_1$ to $\mathcal{y}_2$ (i.e. $\mathcal{y}_1\rightarrow\mathcal{y}_2$). **(c)** shows how the triangle unit from (b) can be an element of a larger system of domains. Finally, **(d)** illustrates the generalized case where in the larger system of domains, consistency can be enforced using invariance along arbitrary paths, as long as their endpoints are the same (here the blue and green paths). This is the general concept behind *inference-path invariance*. The triangle in (b) is the smallest unit of such paths.

temporal domain [217, 57], 3D geometry [73, 166, 66, 87, 248, 242, 92, 230, 252, 244, 120, 47], and in recognition and (conditional/unconditional) image translation [86, 144, 96, 249, 91, 41]. In computer vision, consistency has been extensively utilized in the cycle form and often between two or few domains [249, 91]. In contrast, we consider consistency in the more general form of arbitrary paths with varied-lengths over a large task set, rather than the special cases of short cyclic paths. Also, the proposed approach needs *no prior explicit knowledge* about task relationships [166, 120, 230, 252].

Multi-task learning: In the most conventional form, multi-task learning predicts multiple output domains out of a shared encoder/representation for an input. It has been speculated that the predictions of a multi-task network may be automatically cross-task consistent as the representation from which the predictions are made are shared. This has been observed to not be necessarily true in several works [116, 243, 225, 198], as consistency is not directly enforced during training. We also make the same observation (see visuals here) and quantify it (see Fig. 4.8a), which signifies the need for explicit augmentation of consistency in learning.

Transfer learning predicts the output of a target task given another task's solution as a source. The predictions made using transfer learning are sometimes assumed to be cross-task consistent, which is often found to not be the case [237, 189], as transfer learning does not have a specific mechanism to impose consistency by default. Unlike basic multi-task learning and transfer learning, the proposed method includes explicit mechanisms for learning with general data-driven consistency constraints.

Uncertainty metrics: Among the existing approaches to measuring prediction uncertainty, the proposed Consistency Energy (Sec. 4.4) is most related to Ensemble Averaging [122], with the key difference that the estimations in our ensemble are from *different cues/paths*, rather than retraining/reevaluating the same network with different random initializations or parameters. Using multiple cues is expected to make the ensemble more effective at capturing uncertainty.

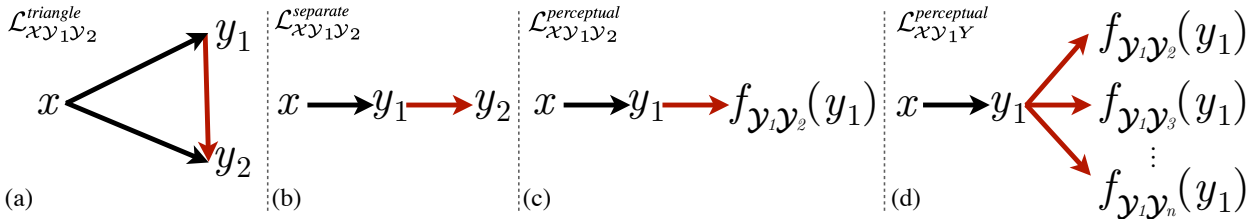


Figure 4.4: **Schematic summary of derived losses for $f_{x y_1}$.(a)**: $\mathcal{L}^{triangle}_{x y_1 y_2}$ (Eq.4.1). **(b)**: $\mathcal{L}^{separate}_{x y_1 y_2}$ (Eq.4.4). **(c)**: $\mathcal{L}^{perceptual}_{x y_1 y_2}$ (Eq.4.7). **(d)**: $\mathcal{L}^{perceptual}_{x y_1 Y}$ (Eq.4.8).

4.3 Method

We define the problem as follows: suppose x denotes the query domain (e.g., RGB images) and $\mathcal{Y}=\{y_1,...,y_n\}$ is the set of n desired prediction domains (e.g., normals, depth, objects, etc). An individual datapoint from domains $(x, y_1,..., y_n)$ is denoted by $(x, y_1,..., y_n)$. The goal is to learn functions that map the query domain onto the prediction domains, i.e. $\mathcal{F}_x=\{f_{x y_j} | y_j \in \mathcal{Y}\}$ where $f_{x y_j}(x)$ outputs y_j given x. We also define $\mathcal{F}_{\mathcal{Y}}=\{f_{y_i y_j} | y_i, y_j \in \mathcal{Y}, i \neq j\}$, which is the set of 'cross-task' functions that map the prediction domains onto each other; we use them in the consistency constraints. For now assume $\mathcal{F}_{\mathcal{Y}}$ is given apriori and frozen; in Sec. 4.3 we discuss all functions fs are neural networks in this paper, and we learn $\mathcal{F}_{\mathcal{Y}}$ just like $\mathcal{F}_x$.

Triangle: The Elementary Consistency Unit

The typical supervised way of training the neural networks in $\mathcal{F}_x$, e.g., $f_{x y_1}(x)$, is to find parameters of $f_{x y_1}$ that minimize a loss with the general form $|f_{x y_1}(x)\text{-}y_1|$ using a distance function as $|.|$, e.g., ℓ_1 norm.

This standard *independent* learning of $f_{x y_i}$s satisfies various desirable properties, including cross-task consistency, if given infinite amount of data, but not under the practical finite data regime. This is qualitatively illustrated in Fig. 4.2 (upper). Thus we introduce additional constraints to guide the training toward cross-task consistency. We define the loss for predicting domain y_1 from x *while enforcing consistency with domain* y_2 as a directed triangle depicted in Fig. 4.3(b):

$$\mathcal{L}^{triangle}_{x y_1 y_2} \triangleq |f_{x y_1}(x) - y_1| + |f_{y_1 y_2} \circ f_{x y_1}(x) - f_{x y_2}(x)| + |f_{x y_2}(x) - y_2|. \tag{4.1}$$

The first and last terms are the standard *direct* losses for training $f_{x y_1}$ and $f_{x y_2}$. The middle term is the *consistency term* which enforces that predicting y_2 out of the predicted y_1 yields the same result as directly predicting y_2 out of x (done via the given cross-task function $f_{y_1 y_2}$).[3] Thus learning to predict y_1 and y_2 are not independent anymore.

The triangle loss 4.1 is the smallest unit of enforcing cross-task consistency. Below we make two improving modifications on it via function 'separability' and 'perceptual losses'.

[3] Operator $\circ$ denotes function composition: $g \circ h(x) \triangleq g(h(x))$.

Separability of Optimization Parameters

The loss $\mathcal{L}^{triangle}_{x y_1 y_2}$ involves *simultaneous* training of two networks $f_{x y_1}$ and $f_{x y_2}$, thus it is resource demanding. We show $\mathcal{L}^{triangle}_{x y_1 y_2}$ can be reduced to a 'separable' function [201] resulting in two terms that can be optimized independently. From the triangle inequality we can derive:

$$|f_{y_1 y_2} \circ f_{x y_1}(x) - f_{x y_2}(x)| \leq |f_{y_1 y_2} \circ f_{x y_1}(x) - y_2| + |f_{x y_2}(x) - y_2|, \tag{4.2}$$

which after substitution in Eq. 4.1 yields:

$$\mathcal{L}^{triangle}_{x y_1 y_2} \leq |f_{x y_1}(x) - y_1| + |f_{y_1 y_2} \circ f_{x y_1}(x) - y_2| + 2|f_{x y_2}(x) - y_2|. \tag{4.3}$$

The upper bound for $\mathcal{L}^{triangle}_{x y_1 y_2}$ in inequality 4.3 can be optimized in lieu of $\mathcal{L}^{triangle}_{x y_1 y_2}$ itself, as they both have the same minimizer.[4] The terms of this bound include either $f_{x y_1}$ or $f_{x y_2}$, but not both, hence we now have a loss separable into functions of $f_{x y_1}$ or $f_{x y_2}$, and they can be optimized independently. The part pertinent to the network $f_{x y_1}$ is:

$$\mathcal{L}^{separate}_{x y_1 y_2} \triangleq |f_{x y_1}(x) - y_1| + |f_{y_1 y_2} \circ f_{x y_1}(x) - y_2|, \tag{4.4}$$

named *separate*, as we reduced the closed triangle objective ${}_{x}\overset{y_1}{\triangle}_{y_2}$ in Eq. 4.1 to two separate path objectives $x \to y_1 \to y_2$ and $x \to y_2$. The first term of Eq. 4.4 enforces the general correctness of predicting y_1, and the second term enforces its consistency with y_2 domain.

Reconfiguration into a "Perceptual Loss"

Training $f_{x y_1}$ using the loss $\mathcal{L}^{separate}_{x y_1 y_2}$ requires a training dataset with multi domain annotations for one input: (x, y_1, y_2). It also relies on availability of a *perfect* function $f_{y_1 y_2}$ for mapping y_1 onto y_2; i.e. it demands $y_2 = f_{y_1 y_2}(y_1)$. We show how these two requirements can be reduced.
Again, from triangle inequality we can derive:

$$|f_{y_1 y_2} \circ f_{x y_1}(x) - y_2| \leq |f_{y_1 y_2} \circ f_{x y_1}(x) - f_{y_1 y_2}(y_1)| + |f_{y_1 y_2}(y_1) - y_2| \tag{4.5}$$

which after substitution in Eq. 4.4 yields:

$$\mathcal{L}^{separate}_{x y_1 y_2} \leq |f_{x y_1}(x) - y_1| + |f_{y_1 y_2} \circ f_{x y_1}(x) - f_{y_1 y_2}(y_1)| + |f_{y_1 y_2}(y_1) - y_2|. \tag{4.6}$$

Similar to the discussion for inequality 4.3, the upper bound in inequality 4.6 can be optimized in lieu of $\mathcal{L}^{separate}_{x y_1 y_2}$ as both have the same minimizer.[5] As the last term is a constant w.r.t. $f_{x y_1}$, the final loss for training $f_{x y_1}$ subject to consistency with domain y_2 is:

$$\mathcal{L}^{perceptual}_{x y_1 y_2} \triangleq |f_{x y_1}(x) - y_1| + |f_{y_1 y_2} \circ f_{x y_1}(x) - f_{y_1 y_2}(y_1)|. \tag{4.7}$$

[4] Both sides of inequality 4.3 are ≥ 0 and $=0$ for the minimizer $f_{x y_1}(x) = y_1$ & $f_{x y_2}(x) = y_2$.

[5] Both sides of inequality 4.6 are ≥ 0 and $=0$ for the minimizer $f_{x y_1}(x) = y_1$. The term $|f_{y_1 y_2}(y_1) - y_2|$ is a constant and ~ 0, as it is exactly the training objective of $f_{y_1 y_2}$. The non-zero residual should be ignored and assumed 0 as the non-zero part is irrelevant to $f_{x y_1}$, but imperfections of $f_{y_1 y_2}$.

The loss $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ no longer includes y_2, hence it admits pair training data (x, y_1) rather than triplet (x, y_1, y_2).[6] Comparing $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ and $\mathcal{L}^{separate}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ shows the modification boiled down to replacing y_2 with $f_{\mathcal{Y}_1\mathcal{Y}_2}(y_1)$. This makes intuitive sense too, as y_2 is the match of y_1 in the $\mathcal{Y}_2$ domain.

Why "perceptual" loss? The process that led to Eq. 4.7 can be generally seen as using the loss $|g \circ f(x) - g(y)|$ instead of $|f(x) - y|$. The latter compares $f(x)$ and y in their explicit space, while the former compares them via the lens of function g. This is often referred to as "perceptual loss" in super-resolution and style transfer literature [103]–where two images are compared in the *representation space* of a network pretrained on ImageNet, rather than in *pixel space*. Similarly, the consistency constraint between the domains $\mathcal{Y}_1$ and $\mathcal{Y}_2$ in Eq. 4.7 (second term) can be viewed as judging the prediction $f_{\mathcal{X}\mathcal{Y}_1}(x)$ against y_1 via the lens of the network $f_{\mathcal{Y}_1\mathcal{Y}_2}$; here $f_{\mathcal{Y}_1\mathcal{Y}_2}$ is a "perceptual loss" for training $f_{\mathcal{X}\mathcal{Y}_1}$. However, unlike the ImageNet-based perceptual loss [103], this function has the specific and interpretable job of enforcing consistency with another task. We also use multiple $f_{\mathcal{Y}_1\mathcal{Y}_i}$s simultaneously which enforces consistency of predicting $\mathcal{Y}_1$ against multiple other domains (Sections 4.3 and 4.3).

Ill-posed tasks and imperfect networks

If $f_{\mathcal{Y}_1\mathcal{Y}_2}$ is a *noisy* estimator, then $f_{\mathcal{Y}_1\mathcal{Y}_2}(y_1) = y_2 + noise$ rather than $f_{\mathcal{Y}_1\mathcal{Y}_2}(y_1) = y_2$. Using a noisy $f_{\mathcal{Y}_1\mathcal{Y}_2}$ in $\mathcal{L}^{separate}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ corrupts the training of $f_{\mathcal{X}\mathcal{Y}_1}$ since the second loss term does not reach 0 if $f_{\mathcal{X}\mathcal{Y}_1}(x)$ correctly outputs y_1. That is in contrast to $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ where both terms have the same global minimum and are always 0 if $f_{\mathcal{X}\mathcal{Y}_1}(x)$ outputs y_1 – even when $f_{\mathcal{Y}_1\mathcal{Y}_2}(y_1) = y_2 + noise$. Thus $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ enables a robust training of $f_{\mathcal{X}\mathcal{Y}_1}(x)$ w.r.t. imperfections in $f_{\mathcal{Y}_1\mathcal{Y}_2}$. This is crucial since neural networks are almost never perfect estimators, e.g., due to lacking an optimal training process for them or potential ill-posedness of the task $y_1 \to y_2$. Further discussion and experiments are available in supplementary material.

Extending Consistency to Multiple Output Domains

The derived $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ loss augments learning of $f_{\mathcal{X}\mathcal{Y}_1}$ with a consistency constraint against *one* domain $\mathcal{Y}_2$. Straightforward extension of the same derivation to enforcing consistency of $f_{\mathcal{X}\mathcal{Y}_1}$ against *multiple* other domains (i.e. when $f_{\mathcal{X}\mathcal{Y}_1}$ is part of multiple simultaneous triangles) yields:

$$\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1 Y} \triangleq |Y| \cdot |f_{\mathcal{X}\mathcal{Y}_1}(x) - y_1| + \sum_{\mathcal{Y}_i \in Y} |f_{\mathcal{Y}_1\mathcal{Y}_i} \circ f_{\mathcal{X}\mathcal{Y}_1}(x) - f_{\mathcal{Y}_1\mathcal{Y}_i}(y_1)| \tag{4.8}$$

where Y is the *set* of domains with which $f_{\mathcal{X}\mathcal{Y}_1}$ must be consistent, and $|Y|$ is the cardinality of Y. Notice that $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1\mathcal{Y}_2}$ is a special case of $\mathcal{L}^{perceptual}_{\mathcal{X}\mathcal{Y}_1 Y}$ where $Y = \{\mathcal{Y}_2\}$. Fig. 4.4 summarizes the derivation of losses for $f_{\mathcal{X}\mathcal{Y}_1}$.

Fig. 4.5 shows qualitative results of learning $f_{\mathcal{X}\mathcal{Y}_1}$ with and without cross-task consistency for a sample query.

[6] Generally for n domains, this formulation allows using datasets of *pairs* among n domains, rather than one n-*tuple* multi annotated dataset.

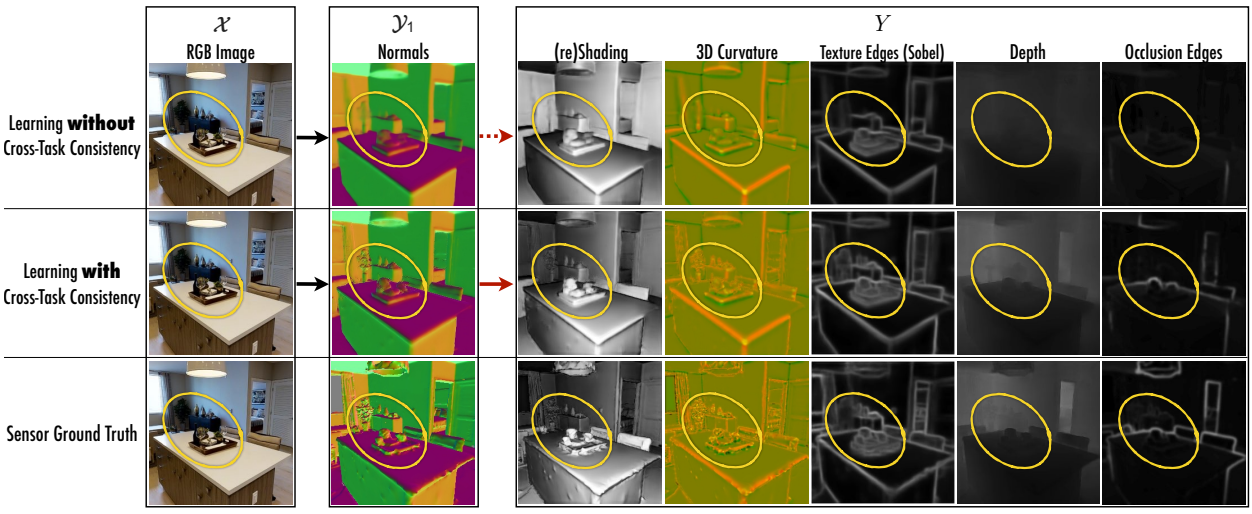


Figure 4.5: **Learning with and without cross-task consistency shown for a sample query**. Using the notation $x \rightarrow y_1 \rightarrow Y$, here x is RGB image, y_1 is surface normals, and five domains in Y are reshading, 3D curvature, texture edges (Sobel filter), depth, and occlusion edges.
Top row shows the results of standard training of $x \rightarrow y_1$. After convergence of training, the predicted normals (y_1) are projected onto other domains (Y) which reveal various inaccuracies. This demonstrates such cross-task projections $y_1 \rightarrow Y$ can provide additional cues to training $x \rightarrow y_1$.
Middle row shows the results of consistent training of $x \rightarrow y_1$ by leveraging $y_1 \rightarrow Y$ in the loss. The predicted normals are notably improved, especially in hard to predict *fine-grained details* (zoom into the yellow markers. Best seen on screen).
Bottom row provides the ground truth. See video examples at visualizations webpage.

Beyond Triangles: Globally Consistent Graphs

The discussion so far provided the loss for the cross-task consistent training of *one* function f_{xy_1} using elementary *triangle* based units. We also assumed the functions $\mathcal{F}_y$ were given apriori. The more general multi-task setup is: given a *large set of domains*, we are interested in learning functions that map the domains onto each other in a *globally cross-task consistent* manner. This objective can be formulated over a graph $\mathcal{G}=(\mathcal{D}, \mathcal{F})$ with nodes representing all of the domains $\mathcal{D}=(x \cup y)$ and edges being neural networks between them $\mathcal{F}=(\mathcal{F}_x \cup \mathcal{F}_y)$; see Fig.4.3(c).

Extension to Arbitrary Paths: The transition from three domains to a large graph $\mathcal{G}$ enables forming more general consistency constraints using *arbitrary-paths*. That is, two *paths* with same endpoint should yield the same results – an example is shown in Fig.4.3(d). The triangle constraint in Fig.4.3(b,c) is a special case of the more general constraint in Fig.4.3(d), if paths with lengths 1 and 2 are picked for the green and blue paths. Extending the derivations done for a triangle in Sec. 4.3 to paths yields:

$$\mathcal{L}^{perceptual}_{xy_1y_2..y_k} = |f_{xy_1}(x) - y_1| + |f_{y_{k-1}y_k} \circ ... \circ f_{y_1y_2} \circ f_{xy_1}(x) - f_{y_{k-1}y_k} \circ ... \circ f_{y_1y_2}(y_1)|, \quad (4.9)$$

which is the loss for training f_{xy_1} using the arbitrary consistency path $x \rightarrow y_1 \rightarrow y_2 ... \rightarrow y_k$ with length k (full derivation provided in supplementary material). Notice that Eq. 4.7 is a special case of

Eq. 4.9 if k=2. Equation 4.9 is particularly useful for incomplete graphs; if the function $y_1 \rightarrow y_k$ is missing, consistency between domains y_1 and y_k can still be enforced via transitivity through other domains using Eq. 4.9.

Also, extending Eq. 4.9 to *multiple simultaneous paths* (as in Eq. 4.8) by summing the path constraints is straightforward.

Global Consistency Objective: We define reaching *global* cross-task consistency for graph $\mathcal{G}$ as satisfying the consistency constraint for *all* feasible paths in $\mathcal{G}$. We can write the global consistency objective for $\mathcal{G}$ as $\mathcal{L}_{\mathcal{G}} = \sum_{p \in \mathcal{P}} \mathcal{L}_p^{perceptual}$, where p represents a path and $\mathcal{P}$ is the set of all feasible paths in $\mathcal{G}$.

Optimizing the objective $\mathcal{L}_{\mathcal{G}}$ directly is intractable as it would require simultaneous training of all networks in $\mathcal{F}$ with a massive number of consistency paths[7]. In Alg.1 we devise a straightforward training schedule for an approximate optimization of $\mathcal{L}_{\mathcal{G}}$. This problem is similar to inference in graphical models, where one is interested in marginal distribution of unobserved nodes given some observed nodes by passing "messages" between them through the graph until convergence. As exact inference is usually intractable for unconstrained graphs, often an approximate message passing algorithm with various heuristics is used.

Algorithm 1: Globally Cross-Task Consistent Learning of Networks $\mathcal{F}$

Result: Trained edges $\mathcal{F}$ of graph $\mathcal{G}$

Train each $f \in \mathcal{F}$ independently. ▷ initialization by standard direct training.
for $k \leftarrow 2$ *to* L **do**
  while *LossConvergence*($\mathcal{F}$) not met **do**
    $f_{ij} \leftarrow$ *SelectNetwork*($\mathcal{F}$) ▷ selects target network to be trained.
    $p \leftarrow$ *SelectPath*($f_{ij}, k, \mathcal{P}$) ▷ selects a feasible consistency path for f_{ij} with maximum length k from $\mathcal{P}$.
    optimize $\mathcal{L}_{ijp}^{perceptual}$ ▷ trains f_{ij} using path constraint p in loss 4.9.
  end
end

Instead of optimizing all terms in $\mathcal{L}_{\mathcal{G}}$, Alg.1 selects one network $f_{ij} \in \mathcal{F}$ to be trained, selects consistency path(s) $p \in \mathcal{P}$ for it, and trains f_{ij} with p for a fixed number of steps using loss 4.9 (or its multi path version if multiple paths selected). This is repeated until all networks in $\mathcal{F}$ satisfy a convergence criterion.

A number of choices for the selection criterion in *SelectNetwork* and *SelectPath* is possible, including round-robin and random selection. While we did not observe a significant difference in the final results, we achieved the best results using *maximal violation* criterion: at each step select the network and path with the largest loss[8]. Also, Alg.1 starts from shorter paths and progressively opens up to longer ones (up to length L) only after shorter paths have converged. This is based on the observation that the benefit of short and long paths in terms of enforcing cross-task consistency overlap, while shorter paths are computationally cheaper[8]. For the same reason, all of the networks are initialized by training using the standard direct loss (Op.1 in Alg.1) before progressively adding consistency terms.

[7] For example, a complete $\mathcal{G}$ with n nodes includes $n(n-1)$ networks and $\sum_{k=2}^{L} \binom{n}{k+1}(k+1)!$ feasible paths, with path length capped at L.

[8] See supplementary material for an experimental comparison.

Finally, Alg.1 does not distinguish between $\mathcal{F}_x$ and $\mathcal{F}_x$ and can be used to train them all in the same pool. This means the selected path p may include networks not fully converged yet. This is not an issue in practice, because, first, all networks are pre-trained with their direct loss (Op.1 in Alg.1) thus they are not wildly far from their convergence point. Second, the perceptual loss formulation makes training f_{ij} robust to imperfections in functions in p (Sec. 4.3). However, as practical applications primarily care about $\mathcal{F}_x$, rather than $\mathcal{F}_y$, one can first train $\mathcal{F}_y$ to convergence using Alg.1, then start the training of $\mathcal{F}_x$ with well trained and converged networks $\mathcal{F}_y$. We do the latter in our experiments.[9] Please see supplementary material for how to normalize and balance the direct and consistency loss terms, as they belong to different domains with distinct numerical properties.

4.4 Consistency Energy

We quantify the amount of cross-task consistency in the system using an energy-based quantity [126] called *Consistency Energy*. For a single query x and domain $\mathcal{Y}_k$, the consistency energy is defined to be the standardized average of pairwise inconsistencies:

$$\text{Energy}_{\mathcal{Y}_k}(x) \triangleq \frac{1}{|\mathcal{Y}|-1} \sum_{\mathcal{Y}_i \in \mathcal{Y}, i \neq k} \frac{|f_{\mathcal{Y}_i \mathcal{Y}_k} \circ f_{\mathcal{X} \mathcal{Y}_i}(x) - f_{\mathcal{X} \mathcal{Y}_k}(x)| - \mu_i}{\sigma_i}, \tag{4.10}$$

where μ_i and σ_i are the average and standard deviation of $|f_{\mathcal{Y}_i \mathcal{Y}_k} \circ f_{\mathcal{X} \mathcal{Y}_i}(x) - f_{\mathcal{X} \mathcal{Y}_k}(x)|$ over the dataset. Eq. 4.10 can be computed per-pixel or per-image by average over its pixels. Intuitively, the energy can be thought of as the amount of variance in predictions in the lower row of Fig. 4.2 – the higher the variance, the higher the inconsistency, and the higher the energy. The consistency energy is an intrinsic quantity of the system and needs no ground truth or supervision.

In Sec. 4.5, we show this quantity turns out to be quite informative as it can indicate the reliability of predictions (useful as a confidence/uncertainty metric) or a shift in the input domain (useful for domain adaptation). This is based on the fact that if the query is from the same data distribution as the training and is unchallenging, all inference paths of a system trained with consistency path constraints work well and yield similar results (as they were trained to); whereas under a distribution shift or for a challenging query, different paths break in different ways resulting in dissimilar predictions, and therefore, creating a higher variance. In other words, usually *correct* predictions are *consistent* while *mistakes* are *inconsistent*. (Plots 4.8b, 4.8c, 4.8d.)

4.5 Experiments

The evaluations are organized to demonstrate the proposed approach yields predictions that are **I.** more *consistent* (Sec.4.5), **II.** more *accurate* (Sec.4.5), and **III.** more *generalizable* to out-of-training-distribution data (Sec.4.5). We also **IV.** quantitatively analyze the *Consistency Energy* and report its utilities (Sec.4.5).

[9] A further cheaper alternative is applying cross-task consistent learning only on $\mathcal{F}_x$ and training $\mathcal{F}_y$ using standard independent training. This is significantly cheaper and more convenient, but still improves $\mathcal{F}_x$ notably.

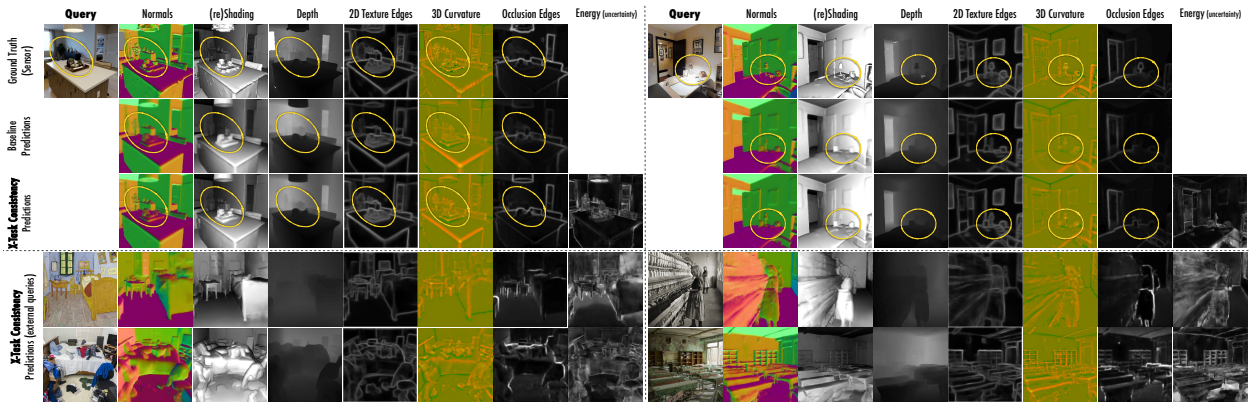


Figure 4.6: **Qualitative results of predicting multiple domains along with the pixel-wise Consistency Energy.** The **top** queries are from the Taskonomy dataset's test set. The results of networks trained with consistency are more accurate, especially in fine-grained regions (zoom into the yellow markers), and more correlated across different tasks. The **bottom** images are external queries (no ground truth available) demonstrating the generalization and robustness of consistency networks to external data. Comparing the energy against a prediction domain (e.g., normals) shows that energy often correlates with error. More examples are provided on the project page, and a live demo for user uploaded images is available at the demo page. **External Queries**: Bedroom in Arles, Van Gogh (1888); Cotton Mill Girl, Lewis Hine (1908); Chernobyl Pripyat Abandoned School (c. 2009). [best seen on screen]

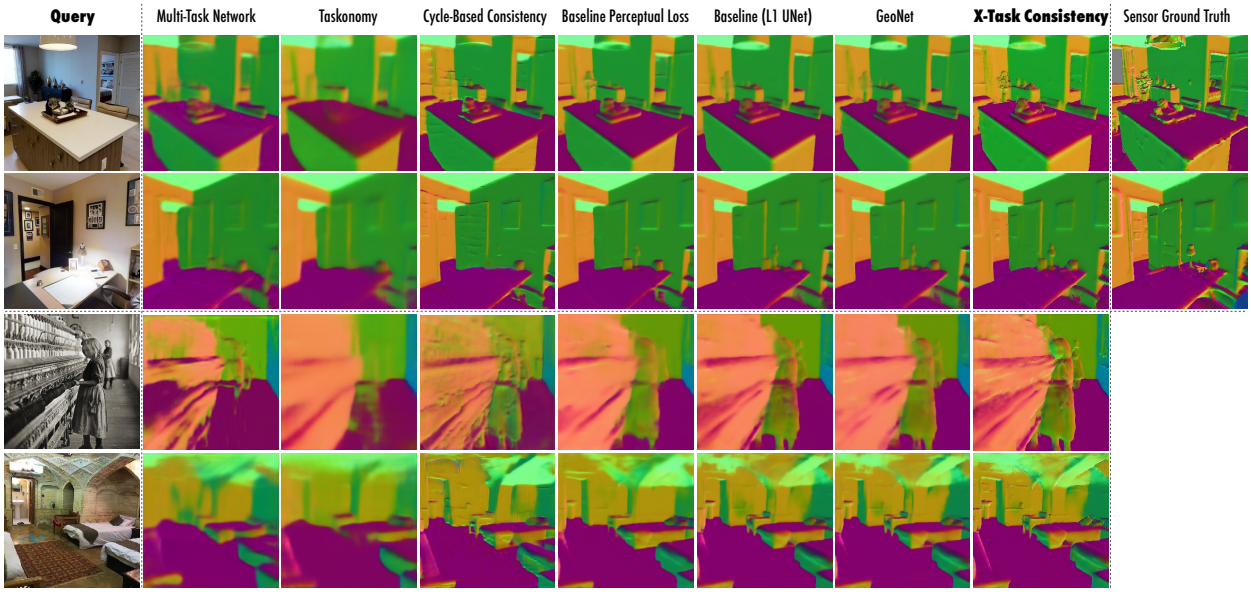


Figure 4.7: **Learning with cross-task consistency vs various baselines** compared over surface normals. Queries are from Taskonomy dataset (**top**) or external data (**bottom**). Similar comparison for other domains and more images are provided on the project page, and a live demo for user uploaded images is available at the demo page. [best seen on screen]

Datasets: We used the following datasets in the evaluations:

Taskonomy [237]: We adopted Taskonomy as our main training dataset. It includes 4 million real images of indoor scenes with multi-task annotations for each image. The experiments were performed using the following 10 domains from the dataset: *RGB images, surface normals, principal curvature, depth (zbuffer), reshading, 3D (occlusion) edges, 2D (Sobel) texture edges, 3D keypoints, 2D keypoints, and semantic segmentation*. The tasks were selected to cover 2D, 3D, and semantic domains and have sensor-based/semantic ground truth. We report results on the test set. Also, as one of the out-of-domain tests, we use a version of Taskonomy images where they undergo distortions (e.g., blurring).

Replica[203] has high resolution 3D ground truth and enables more reliable evaluations of fine-grained details. We *test* on 1227 images from Replica (no training), besides Taskonomy test data.

CocoDoom [135] contains synthetic images from the *Doom* video game. We use it as one of the out-of-training-distribution datasets.

ApolloScape [93] contains real images of outdoor driving scenes. We use it as another out-of-training-distribution dataset.

NYU [193]: We also evaluated on NYUv2. The findings are similar to those on Taskonomy and Replica (in supplementary material).

Architecture & Training Details: We used a UNet [178] backbone architecture. We benchmarked alternatives, e.g., ResNet [82], and found UNets to yield superior pixel-wise predictions. All networks in $\mathcal{F}_x$ and $\mathcal{F}_y$ have a similar architecture. The networks have 6 down and 6 up sampling blocks and were trained using AMSGrad [174] and Group Norm [222] with learning rate 3×10^{-5}, weight decay 2×10^{-6}, and batch size 32. Input and output images were linearly scaled to the range $[0, 1]$ and resized down to 256×256. We used ℓ_1 as the norm in all losses and set the max path length L=3. We experimented with different loss normalization methods and achieved the best results when the loss terms are weighted negative proportional to their respective gradient magnitude (details in supplementary material).

Baselines: The main baseline categories are described below. To prevent confounding factors, our method and all baselines were implemented using the *same UNet network* when feasible and were *re-trained on Taskonomy dataset*.

Baseline UNet (standard independent learning) is the main baseline. It is identical to consistency models in all senses, except being trained with only the direct loss and no consistency terms.

Multi-task learning: A network with one shared encoder and multiple decoders each dedicated to a task, similar to [116]. This baseline shows if consistency across tasks would emerge by sharing a representation without explicit consistency constraints.

Cycle-based consistency, e.g.[249], is a way of enforcing consistency between two domains *assuming a bijection* between them. This assumption is violated between many domains (e.g. RGB↔3D, as texture cannot be recovered from 3D). This baseline is a special case of the triangle in Fig.4.3(b) by setting y_2=x.

Baseline perceptual loss network uses frozen random (Gaussian weight) networks as $\mathcal{F}_y$, rather than training them to be cross-task functions. This baseline would show if the improvements were owed to the priors in the architecture of constraint networks, rather than them executing cross-task consistency constraints.

GAN-based image translation: We used Pix2Pix [96], which is conditional GAN based framework [144].

Method \ Setup	Replica Dataset									Taskonomy Dataset															
	Normals			**Depth**			**reShading**			**Normals**					**Depth**					**reShading**					**Semantic Segm.**
	Perceptual Err.		Direct	Perceptual Err.		Direct	Perceptual Err.		Direct	Perceptual Err.				Direct	Perceptual Err.				Direct	Perceptual Err.				Direct	Direct
	Depth	reShade	ℓ_1 Err.	Norm.	reShade	ℓ_1 Err.	Norm.	Depth	ℓ_1 Err.	Depth	reShade	Curv.	Edge(2D)	ℓ_1 Err.	Norm.	reShade	Curv.	Edge(2D)	ℓ_1 Err.	Norm.	Depth	Curv.	Edge(2D)	ℓ_1 Err.	X-Entropy (↓)
Blind Guess	4.75	33.31	16.02	22.23	19.94	4.81	15.74	5.14	16.45	7.39	38.11	3.91	12.05	17.77	22.37	27.27	7.96	12.77	7.07	19.96	7.14	3.53	12.62	24.85	
Taskonomy Networks	3.73	11.07	6.55	18.06	15.39	3.72	8.70	3.85	11.43	7.19	22.68	3.68	10.70	7.54	18.82	20.83	6.65	14.10	4.55	11.72	4.69	3.54	11.19	16.58	
Multi-Task	5.58	22.11	6.03	15.30	16.14	2.44	7.24	3.36	10.32	8.78	27.32	3.65	10.16	7.07	17.18	19.55	7.54	13.67	2.81	9.19	3.54	3.56	10.75	11.61	
GeoNet (original)	6.23	19.34	7.48	13.88	14.03	4.01	×	×	×	7.71	27.35	**3.32**	**9.09**	9.58	15.44	18.73	4.03	10.78	4.07	×	×	×	×	×	×
Cycle Consistency	5.65	22.39	7.13							8.81	30.33	3.84	10.26	8.68											
Baseline Perceptual Loss	4.88	15.34	4.99							8.59	23.98	3.41	10.01	6.17											
Pix2Pix	4.52	19.03	7.70							8.12	26.23	3.83	10.33	9.40											
Baseline UNet (ℓ_1)	4.69	13.15	4.96	10.47	12.99	1.99	6.90	2.74	**9.55**	8.17	20.94	3.41	9.98	**5.95**	13.62	15.68	7.31	12.61	**2.27**	9.58	3.38	3.78	10.85	**10.45**	0.246
GeoNet (updated)	4.62	12.79	**4.70**	10.47	12.75	1.83	×	×	×	8.18	20.84	3.40	9.99	**5.91**	13.77	15.76	7.52	12.67	**2.26**	×	×	×	×	×	×
X-Task Consistency	**2.07**	**9.99**	**4.80**	**7.01**	**11.21**	**1.63**	**5.50**	**1.96**	**9.22**	**4.32**	**12.15**	**3.29**	9.50	6.08	**9.46**	**12.66**	**3.61**	**9.82**	**2.29**	**7.13**	**2.51**	**3.28**	**9.38**	**10.52**	**0.237**
0.25% Data: Baseline	5.65	21.76	7.61							8.86	26.91	3.78	10.31	**8.17**											
0.25% Data: Consistency	**2.41**	**12.26**	**7.28**							**5.07**	**15.96**	**3.74**	**9.93**	9.19											

Table 4.1: **Quantitative Evaluation of Cross-Task Consistent Learning vs Baselines.** Results are reported on Replica and Taskonomy Datasets for four prediction tasks (normals, depth, reshading, pixel-wise semantic labeling) using 'Direct' and 'Perceptual' error metrics. The Perceptual metrics evaluate the target prediction in another domain (e.g., the *leftmost* column evaluates the *depth* inferred out of the predicted *normals*). **Bold** marks the best-performing method. If more than one value is bold, their performances were statistically indistinguishable from the best, according to 2-sample paired t-test $\alpha = 0.01$. Learning with consistency led to improvements with large margins in most columns. (In all tables, ℓ norm values are multiplied by 100 for readability. Methods that cannot be run for a given target are denoted by '×'.)

Blind guess: A query-agnostic statistically informed guess computed from data for each domain (visuals in supplementary). It shows what can be learned from general dataset regularities. [237]

GeoNet [166] is a task-specific consistency method analytically curated for depth and normals. This baseline shows how closely the task-specific consistency methods based on known analytical relationships perform vs the proposed generic data-driven method. The "original" and "updated" variants represent original authors' released networks and our re-implemented and re-trained version.

Consistency of Predictions

Fig. 4.8a (blue) shows the amount of inconsistency in test set predictions (Consistency Energy) successfully decreases over the course of training. The convergence point of the network trained with consistency constraints is well below baseline independent learning (orange) and multi-task learning (green)–which shows consistency among predictions *does not naturally emerge* in either case without explicit constraining. Plots of individual loss terms similarly show minimizing the direct term does not lead to automatic minimization of consistency terms (provided in supplementary).

Accuracy of Predictions

Figures 4.6 and 4.7 compare the prediction results of networks trained with cross-task consistency against the baselines in different domains. The improvements are considerable particularly around the difficult *fine-grained details*.

Quantitative evaluations are provided in Tab. 4.1 for Replica dataset and Taskonomy datasets on depth, normal, reshading, and pixel-wise semantic prediction tasks. Learning with consistency led to large improvements in most of the setups. As most of the pixels in an image belong to easy to predict regions governed by the room layout (e.g., ceiling, walls), the standard pixel-wise error metrics (e.g., ℓ_1) are dominated by them and consequently insensitive to fine-grained changes. Thus,

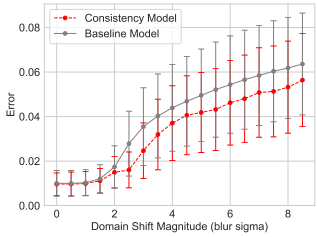


Figure 4.9: **Error with Increasing (Smooth) Domain Shift.** The network trained with consistency is more robust to the shift.

Novel Domain	Error (Post-Adaption) # images	Consistency	Baseline	Error (Pre-Adaptation) Consistency	Baseline
Gaussian blur (Taskonomy)	128	17.4 (+14.7%)	20.4	46.2 (+12.8%)	53.0
	16	22.3 (+8.6%)	24.4		
CocoDoom	128	18.5 (+19.2%)	22.9	54.3 (+15.8%)	64.5
	16	27.1 (+24.5%)	35.9		
ApolloScape	8	40.5 (+11.9%)	46.0	55.8 (+5.5%)	59.1

Table 4.2: **Domain generalization and adaptation** on CocoDoom, ApolloScape, and Taskonomy blur data. Networks trained with consistency show better generalization to new domains and a faster adaptation with little data. (relative improvement in parentheses)

besides standard *Direct* metrics, we report *Perceptual* error metric (e.g., *normal→curvature*) that evaluate the same prediction, but with a non-uniform attention to pixel properties.[10] Each perceptual error provides a different angle, and the optimal results would have a low error for *all* metrics.

The corresponding *Standard Error* for the reported numbers are provided in supplementary material, which show the trends are statistically significant. Tab. 4.1 also includes evaluation of the networks when trained with little data (0.25% subset of Taskonomy dataset), which shows the consistency constraints are useful under low-data regime as well.

We adopted normals as the canonical task for more extensive evaluations, due to its practical value and abundance of baselines. The conclusions remained the same regardless.

Using Consistency with Unsupervised Tasks: Unsupervised tasks can provide consistency constraints, too. Examples of such tasks are 2D Edges and 2D Keypoints (SURF[22]), which are included in our dictionary. Such tasks have fixed operators that can be applied on any image to produce their respective domains without any additional supervision. Interestingly, we found enforcing consistency with these domains is still useful for gaining better results (see supplementary material for the experiment). The ability to utilize unsupervised tasks further extends the applicability of our method to single/few task datasets.

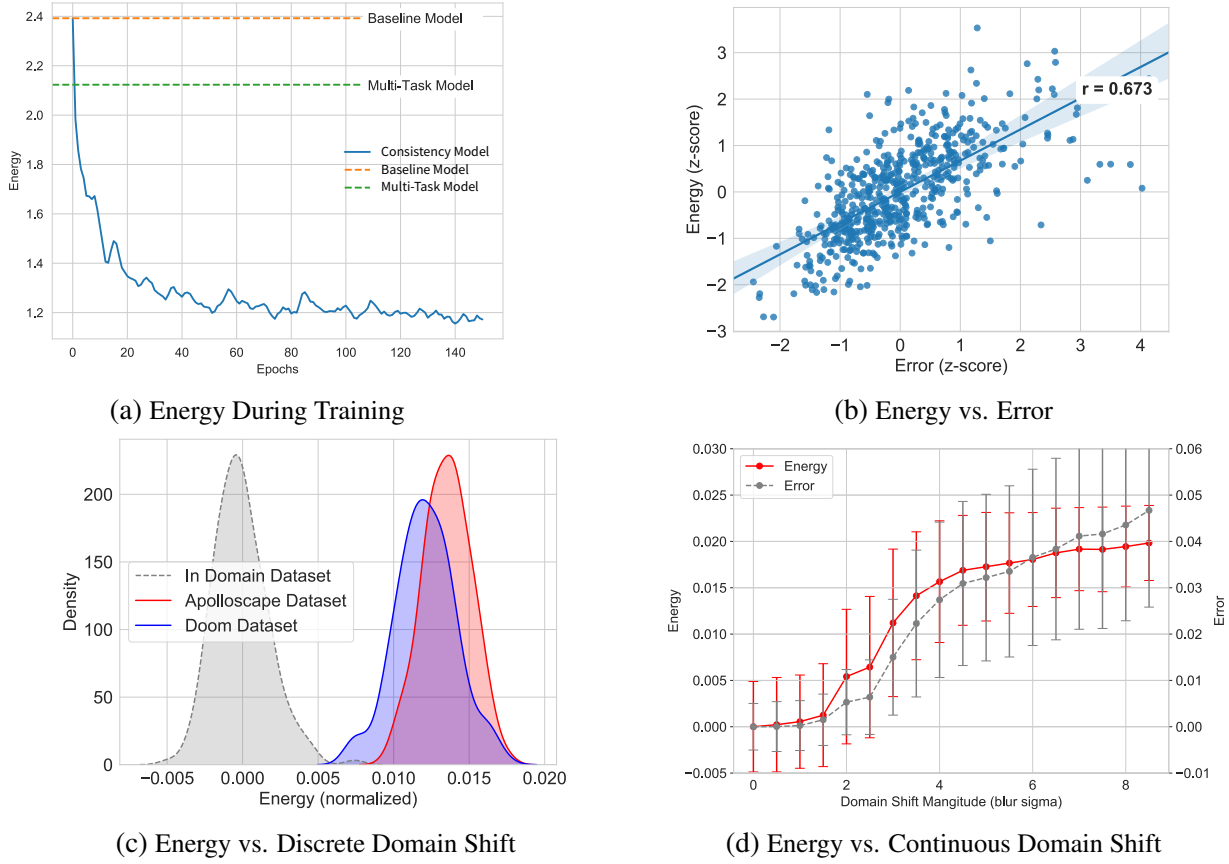
Utilities of Consistency Energy

Below we quantitatively analyze the Consistency Energy. The energy is shown (per-pixel) for sample queries in Fig. 4.6.

Consistency Energy as a Confidence Metric (Energy vs Error): Plot 4.8b shows the energy of predictions has a strong positive correlation with the error computed using ground truth (Pearson corr. 0.67). This suggests the energy can be adopted for confidence quantification and handling uncertainty. This experiment was done on Taskonomy test set thus images had no domain shift from the training data.

Consistency Energy as a Domain Shift Detector: Plot 4.8c shows the energy distribution of in-distribution (Taskonomy) and out-of-distribution datasets (ApolloScape, CocoDoom). Out-

[10]For example, evaluation of normals via the *normal→curvature* metric is akin to paying more attention to where normals change, hence reducing the domination of flat regions, such as walls, in the numbers.



(a) Energy During Training

(b) Energy vs. Error

(c) Energy vs. Discrete Domain Shift

(d) Energy vs. Continuous Domain Shift

Figure 4.8: **Analyses of Consistency Energy.** The energy goes down during training (a), and on each image is correlated with actual prediciton error (b). It can be used to estimate when the network is failing due to domain shift: both discrete shifts (c) and continuous shifts (d).

of-distribution datapoints have notably higher energy values, which suggests that energy can be used to detect anomalous samples or domain shifts. Using the per-image energy value to detect out-of-distribution images achieved ROC-AUC=0.95; the out-of-distribution detection method OC-NN [33] scored 0.51.

Plot 4.8d shows the same concept as 4.8c (energy vs domain shift), but when the shift away from the training data is smooth. The shift was done by applying a progressively stronger Gaussian blur with kernel size 6 on Taskonomy test images. The plot also shows the error computed using ground truth which has a pattern similar to the energy.

We find the reported utilities noteworthy as handling uncertainty, domains shifts, and measuring prediction confidence in neutral networks are open topics of research [156, 77] with critical values in, e.g., active learning [188], real-world decision making [115], and robotics [164].

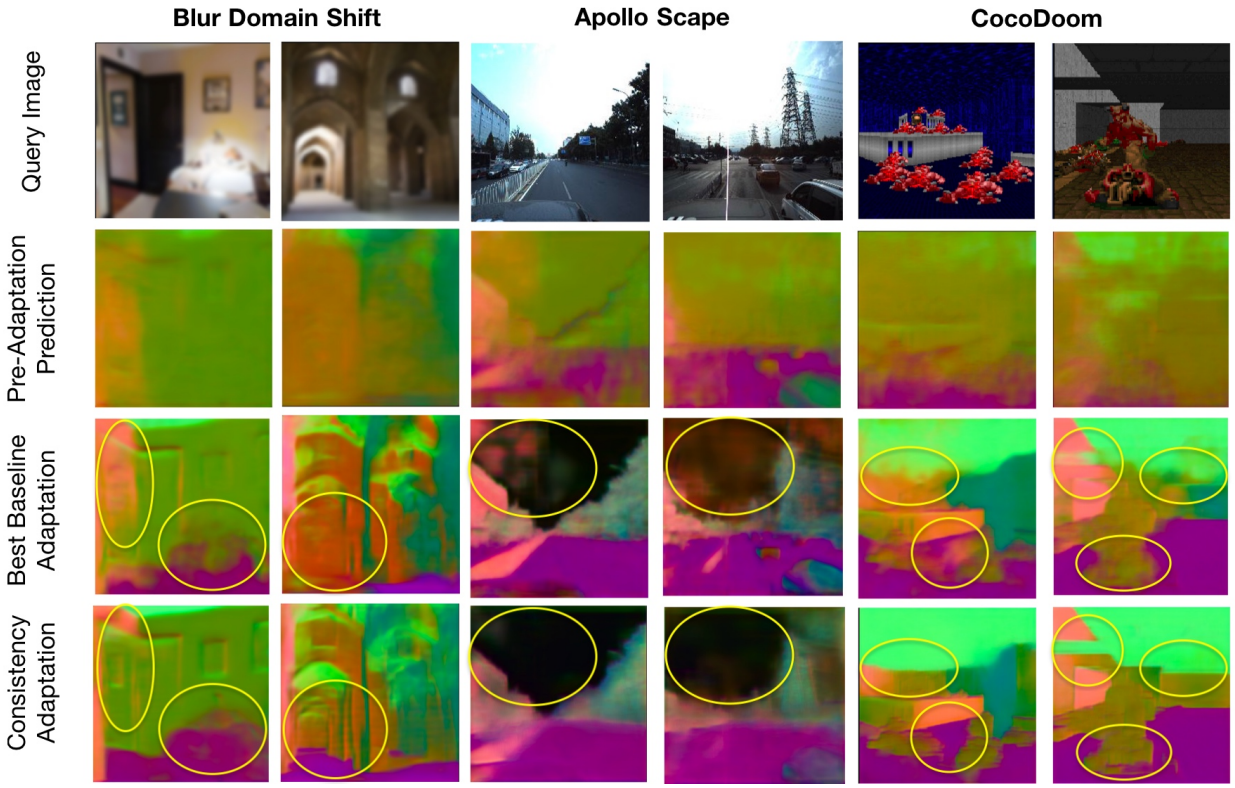


Figure 4.10: **Domain adaptation** results shown for three target domains (ApolloScape [93], CocoDoom [135], Gaussian-blur Taskonomy [237]). Networks trained with consistency show better adaptation with little data.

Generalization & Adaptation to New Domains

To study: **I.** how well the networks generalize to new domains without any adaptation and quantify their resilience, and **II.** how efficiently they can adapt to a new domain given a few training examples by fine-tuning, we test the networks trained on Taskonomy dataset on various new domains. The experiment were conducted on smooth (blurring [102]) and discrete (Doom [135], ApolloScape [93]) shifts. For (**II**), we use a small number (16-128) of images from the new domain to fine-tune the networks with and without consistency constraints. The original training data (Taskonomy) is retained during fine-tuning so prevent the networks from forgetting the original domain [129].

Models trained with consistency constraints generally show more robustness against domain shifts (see Fig. 4.9 and pre-adaptation numbers in Table 4.2) and a better adaptation with little data (see post-adaptation numbers in Table 4.2 and Fig. 4.10). The challenging external queries shown in Figures 4.6&4.7&4.1 similarly denote a good generalization.

Supplementary Material: We defer additional discussions and experiments, particularly analyzing different aspects of the optimization, stability analysis of the experimental trends, and proving qualitative results at scale to the supplementary material and the project page.

4.6 Conclusion and Limitations

We presented a general and data-driven framework for augmenting standard supervised learning with cross-task consistency. The evaluations showed learning with cross-task consistency fits the data better yielding more accurate predictions and leads to models with improved generalization. The Consistency Energy was found to be an informative intrinsic quantity with utilities toward confidence estimation and domain shift detection. Below we briefly discuss some of the limitations and assumptions:

Path Ensembles: We used the various inference paths only as a way of enforcing consistency. Aggregation of *multiple* (comparably weak) inference paths into a *single strong* estimator (e.g., in a manner similar to boosting) is a promising direction that this paper did not address. Performing the aggregation in a probabilistic manner seems viable, as we found the errors of different paths are sufficiently uncorrelated, suggesting possibility of assembling a strong estimator.

Unlabeled/Unpaired Data: The current framework requires paired training data. Extending the concept to unlabeled/unpaired data, e.g., as in [249], appears feasible and remains open for future work.

Categorical/Low-Dimensional Tasks: We primarily experimented with pixel-wise tasks. Classification tasks, and generally tasks with low-dimensional outputs, will be interesting to experiment with, especially given the more severely ill-posed cross-task relationships they induce.

Optimization Limits: The improvements gained by incorporating consistency are bounded by the success of available optimization techniques, as addition of consistency constrains at times makes the optimization job harder. Also, implementing cross-task functions as neural networks makes them subject to certain **output artifacts** similar to those seen in image synthesis with neural networks.

Adversarial Robustness: Lastly, if learning with cross-task consistency indeed reduces the tendency of neural networks to learn surface statistics [102] (Sec. 4.1), studying its implications in defence against adversarial attacks will be worthwhile.

Energy Analyses: We performed post-hoc analyses on the Consistency Energy. More concrete understanding of the properties of the energy and potentially using it actively for network modification, e.g, in unsupervised domain adaptation, requires further focused studies.

Chapter 5

Scaling Datasets to Train Robust Representations

In previous chapters, our focus was primarily on indoor spaces, as we had access to both comprehensive multi-task datasets and effective simulators in that context. The multi-task datasets allowed us to train different representation models using the same input pixels, and thereby isolate the effects of pretraining objectives on subsequent transfer learning for downstream behaviors. This approach worked for the studies we conducted, as the behaviors were *also* assessed in indoor spaces, ensuring in-domain evaluation. Our research led to intriguing discoveries about the relationship between representations and downstream behavior, with results remaining robust across variations in data size, simulator, robot morphology, and learning algorithms.

One key finding was the significant utility of surface normals for manipulation tasks, where contact points are crucial. Surprisingly, there are few image datasets with surface normal labels, and the best monocular surface normal estimators currently available demonstrate limited robustness and accuracy when applied to inputs other than indoor scenes.

In this chapter, we build upon our findings regarding the usefulness of certain pretraining objectives (e.g., surface normals) and shift our focus to developing robust and accurate models trained for those objectives. Specifically, this chapter concentrates on creating large-scale and diverse vision datasets by leveraging 3D assets constructed from robot experience or available on the internet. As more 3D data is uploaded to the web and simulators improve, we can utilize this 3D data to generate new image-based datasets for tasks like surface normals. We develop a pipeline for generating such datasets, parametrically creating views and labels that can be used to train robust models for surface normals and other applications.

The resulting surface normal models are highly effective, and since this pipeline parametrically generates data, it also offers a way to experiment with the settings crucial for creating robust models. For instance, we discovered that randomizing the field-of-view in the training data was essential.

This chapter is based on joint work with Ainaz Eftekhar, Roman Bachmann, Jitendra Malik, Amir Zamir [59], and is presented much as it appeared in the ICCV 2021 proceedings.

5.1 Introduction

This paper introduces a pipeline to bridge the gap between comprehensive 3D scans and static vision datasets. Specifically, we implement and provide a platform that takes as input one of the following:

- a textured mesh,
- a mesh with images from an actual camera/sensor,
- a 3D pointcloud and aligned RGB images,

and generates a multi-task dataset with as many cameras and images as desired to densely cover the space. For each image, there are 21 different default mid-level cues, shown in Fig. 3.1. The software makes use of Blender [44], a powerful physics-based 3D rendering engine to create the labels, and exposes complete control over the sampling and generation process. With the proliferation of reasonably-priced 3D sensors (e.g. Kinect, Matterport, and the newest iPhone), we anticipate an increase in such 3D-annotated data.

In order to establish the soundness for training computer vision models, we used our pipeline to annotate several existing 3D scans and produce a medium-size starter dataset of mid-level cues. Samples of the data and different cues are shown in Fig. 5.4. Standard models trained on this starter dataset achieve state-of-the-art performance for several standard computer vision tasks. For surface normal estimation, a standard UNet [179] model trained on this starter dataset yields human-level surface normal estimation performance on the in-the-wild dataset OASIS [38], even though the model never saw OASIS data during training. For depth estimation, our DPT-Hybrid [171] is comparable to or outperforms state-of-the-art models such as MiDaS DPT-Hybrid [172, 171]. The qualitative performance of these networks (shown in Figs. 5.5, 5.6) is often better than the numbers suggest, especially for fine-grained details.

We further provide an ecosystem of tools and documentation around this platform. Our project website contains links to a Docker containing the annotator and all necessary libraries, PyTorch [158] dataloaders to efficiently load the generated data, pretrained models, scripts to generate videos in addition to images, and other utilities.

We argue that these results should not be interpreted narrowly. The core idea of the platform is that the “sectors of the ambient [light-field] array are not to be confused with temporary *samples* of the array" (J. J. Gibson [70]). That is, static images only represent single samples of the entire 360-degree panoramic light-field environment surrounding an agent. How an agent or model samples and represents this environment will affect its performance on downstream tasks. The proposed platform in this paper is designed to reduce the technological barriers for research into the effect of data sampling practices and into the interrelationships between data distribution, data representation, models, and training algorithms. We discuss directions here and analyze a few illustrative examples in the final section of the paper.

First, the pipeline proposed in this paper provides a possible pathway to understand such sampling effects. That is, the rendering pipeline offers complete control over (heretofore) fixed design choices such as camera intrinsics, scene lighting, object-centeredness [165], the level of “photographer’s bias” [16], data domain, and so on. This makes it possible to run intervention

studies (e.g. A/B tests), without collecting and validating a new dataset or relying on a post-hoc analysis. As a consequence, this provides an avenue for a computer vision "dataset design guide".

Second, vision is about much more than semantic recognition, but our datasets are biased towards that as the core problem. The best-studied, most diverse and largest dataset (>10M images) generally contains some form of textual/class labels [49, 204] and only RGB images. On the other hand, datasets for most non-classification tasks remain tiny by modern standards. For example, the indoor scene dataset NYU [194], still used for training some state-of-the-art depth estimation models [231], contains only 795 training images—all taken with a single camera. The pipeline presents a way to generate datasets of comparable quality for non-recognition tasks.

Third, the generated data allows "matched-pair experimental design" that simplifies study into the *interrelationships* of different tasks, since the pipeline produces labels for every sample. In particular, it helps to avoid issues like the following: suppose a model trained for object classification on ImageNet transfers to COCO [130] better than a model trained for depth estimation on NYU [194]–is that due to the data domain, the training task, the diversity of camera intrinsics, or something else?

Existing matched-pair datasets usually focus on a single domain (indoor scenes [238, 194, 14, 202], driving [55, 46], block-worlds [104], etc.) and contain few cues [46, 194, 14, 202]. The provided starter dataset may be a better candidate for this research than these existing datasets, since it contains over 14.5 million images from different domains (more than the full ImageNet database), contains many different cues (e.g. for depth, surface normals, curvature, panoptic segmentation, and so on), and models trained on this dataset reach excellent performance for several tasks and existing benchmarks. We demonstrate the value of such matched-pairs data in Sec. 5.5,

Though our pipeline is designed to facilitate understanding the principles of dataset design, vision beyond recognition, the interrelationships between data, tasks, and models, this paper does not extensively pursue those questions themselves. It provides a few analyses, but these are merely intended as illustrative examples. Instead, the paper introduces tooling designed to facilitate such research as 3D data becomes more widely available and the capture technology improves. On our website, we provide a documented, open-sourced, and Dockerized annotator pipeline with a convenient CLI, runnable examples, a live demo, the starter dataset, pretrained models, PyTorch dataloaders, and code for the paper (including annotator and models).

5.2 Related Work

In this section we examine related datasets and other approaches. A thorough review would take more space than we have, so we restrict our attention to only the most relevant groupings.

Static 3D Datasets. The past few years have witnessed an uptick in the number of mesh-based datasets, thanks largely to the availability of reasonably-priced 3D scanners. Each dataset in the current crop, though, usually consists of scenes in a restricted domain. Prominent examples of indoor building datasets include Stanford Building Dataset (S3DIS) [12], Matterport3D [34], Taskonomy [238], Replica [202], 2D-3D-Semantic [13], Habitat-Matterport [137], and Hypersim [176]. Other datasets contain primarily outdoor scenes, usually driving–such as CARLA [55], GTA5 [175]–

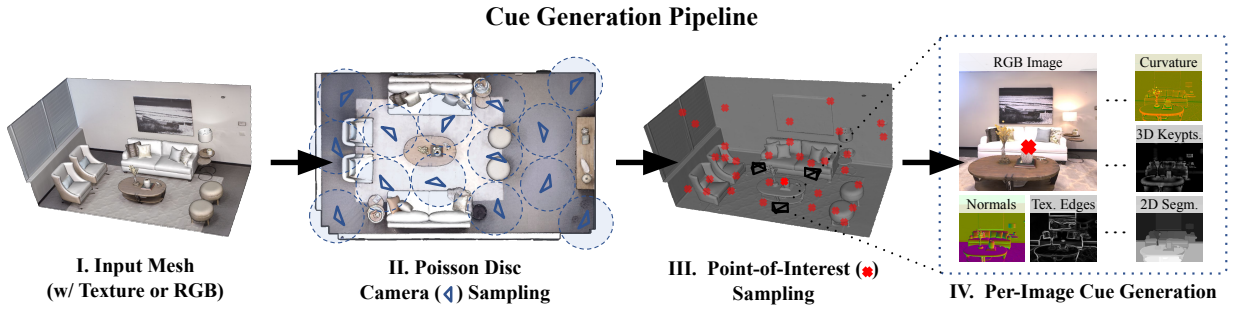


Figure 5.1: **Overview of the generation pipeline.** (**I**) Given a textured mesh (or other options discussed in Sec. 5.3), our pipeline (**II**) generates dense camera locations, (**III**) generates points-of-interest subject to multi-view constraints and (**IV**) produces 21 different mid-level cues for each (shown in Fig. 3.1).

or other narrow domains such as the aptly-named Tanks and Temples [112] dataset. Models trained on such scene-level views often do not generalize to object-centric views (see Fig. 5.6), but existing datasets with high-resolution object meshes do not include 2D images samples [95, 29].

Other recent datasets aim to link diverse monocular 2D images and corresponding 3D meshes, but take the reverse approach of this paper by using hand-annotation to create meshes from single-view in-the-wild RGB samples [38, 37]. This labeling process is expensive and time-consuming, and crucially does not allow regenerating the image dataset. In Sec. 5.4, we consider our pipeline vs. OASIS, one of the largest and most diverse of these benchmarks, and demonstrate that models trained on our starter dataset already match human-level performance on OASIS—outperforming the same architectures models trained on OASIS itself.

Vision-Focused Simulators. Like our platform, simulators typically take a textured mesh as the representation of the scene and aim to produce realistic sensory inputs [137, 223]. While spiritually similar to the pipeline proposed in this paper, the current generation of simulators is designed first and foremost to train embodied agents. They prioritize rendering speed and real-time mechanics at the cost of photorealism and cue diversity [107, 146]. Extending such simulators to handle additional cues or to parametrically render out vision datasets often requires writing new components of the simulator codebase (usually in C++, CUDA, or OpenGL), a surmountable but unpleasant barrier to entry. In contrast, our platform extends Blender which "supports the entirety of the 3D pipeline" [45] and provides Python bindings that will be intuitive to most vision researchers, and we implement many of these cues and sampling methods out-of-the-box. In short, we provide a bridge between simulators and static vision datasets.

Multi-Task Datasets. Vision-based multi-task learning (MTL), like computer vision in general, shows a general bias towards recognition. MTL datasets often take different shades of classification as the core problem of interest [121, 216, 136]. In particular, MTL literature often focuses on binary attribute classification in specialized domains, such as Caltech-UCSD Birds [218] or CelebA [133]. Visual MTL datasets that contain non-recognition tasks often contain only a single domain or a few tasks (NYU [194], CityScapes [46] or Taskonomy [238]). Sometimes, MTL papers take mix datasets for a "single" task and consider each dataset as a different task [131, 172, 123, 171].

In general, the multi-task learning literature has not converged on a standardized definition of the setting or dataset. Recent work has demonstrated that MTL methods developed on existing datasets seem to specialize to their respective developement set and do not perform well on large, realistic datasets, or on other tasks [212, 213, 239]. This underscores the importance of developing *realistic training setting and datasets* that generalizes to real-world scenarios.

Data Augmentation + Domain Randomization. Data augmentation is a way to modify the data or training regimen so that the trained model exhibits desirable invariances (or equivariances). During training, *any* transformation of sensor inputs that determines a unique (possibly identity) transformation on the label can be used as "augmented" data. For example, simple 2D augmentations such as 2D affine transformations, crops, and color changes that leave the labels unchanged are the common in computer vision [36, 75], since they can be used even when datasets lack 3D geometry information. In robotics and reinforcement learning where 3D simulators are more standard, data augmentation was introduced as "domain randomization" [209], and common augmentations include texture and background randomization on the scene mesh. Recently, [51] introduced a Blender-based approach for doing domain randomization and creating static datasets of RGB, depth, and surface normals from SunCG [205].

Our pipeline makes all these augmentations available for static computer vision datasets: not just flips/crops/texture randomization, but also dense viewpoints, multi-view consistency, Euclidean transforms, lens flare, etc.). We implement and examine depth-of-field augmentation in Sec. 5.5.

Auto Labeling is an umbrella term for a group of data labeling procedures that harness structure in the data as constraints in order to prune or propagate labels and save annotation labor. This is accomplished primarily by I) pre-trained models as noisy annotator (e.g. [11, 90, 187]), and/or II) aggregating and filtering annotations based on known constraints (e.g. backprojection error, bundle adjustment, temporal smoothness, or [94, 101, 14]). Our pipeline has connections to auto labeling in the sense that we make use of the strong structure present in 3D scanned data to compute and propagate labels across images and reduce the load of (automatically or manually) labeling each image.

5.3 Pipeline Overview

We call our pipeline `Omnidata` as it aspires to encapsulate comprehensive scene information ("omni") in the generated "data". **Try a live example here** to get acquainted with the pipeline. The example uses the CLI and a YAML-like config file to generate images from a textured mesh in Replica [202].

Inputs: The annotator operates upon the following inputs:

- Untextured Mesh (.obj or .ply)
- *Either:* Mesh Texture or Aligned RGB Images
- *Optional:* Pre-Generated Camera Pose File

A 3D pointcloud can be used as well: simply mesh the pointcloud using a standard mesher like COLMAP [184]. An example of meshing and using a 3D pointcloud with the annotator, as well as a more complete description of inputs are available in the supplementary.

Outputs: The pipeline generates 21 mid-level cues in the initial release. All labels are available for all generated images (or videos). Fig. 3.1 provides a visual summary of the different types of outputs. A detailed description of the default mid-level cues and additional outputs provided by the `Omnidata` annotator is included in the supplementary.

Sampling and Generation

In this section we provide a high-level outline of the generation and rendering process (see Fig. 5.1), deferring full details to the supplementary.

First, the annotator generates camera locations (Fig. 5.1 II) and points-of-interest (Fig. 5.1 III) along the mesh.

Second, for each camera and each point-of-interest, it creates a view from that camera fixated on the point (three fixated views are depicted in the lower part of Fig. 5.2).

Third, for each space-point-view triplet, the annotator renders (Fig. 5.3) all the mid-level cues (Fig. 5.1 IV).

Each step is elaborated next.

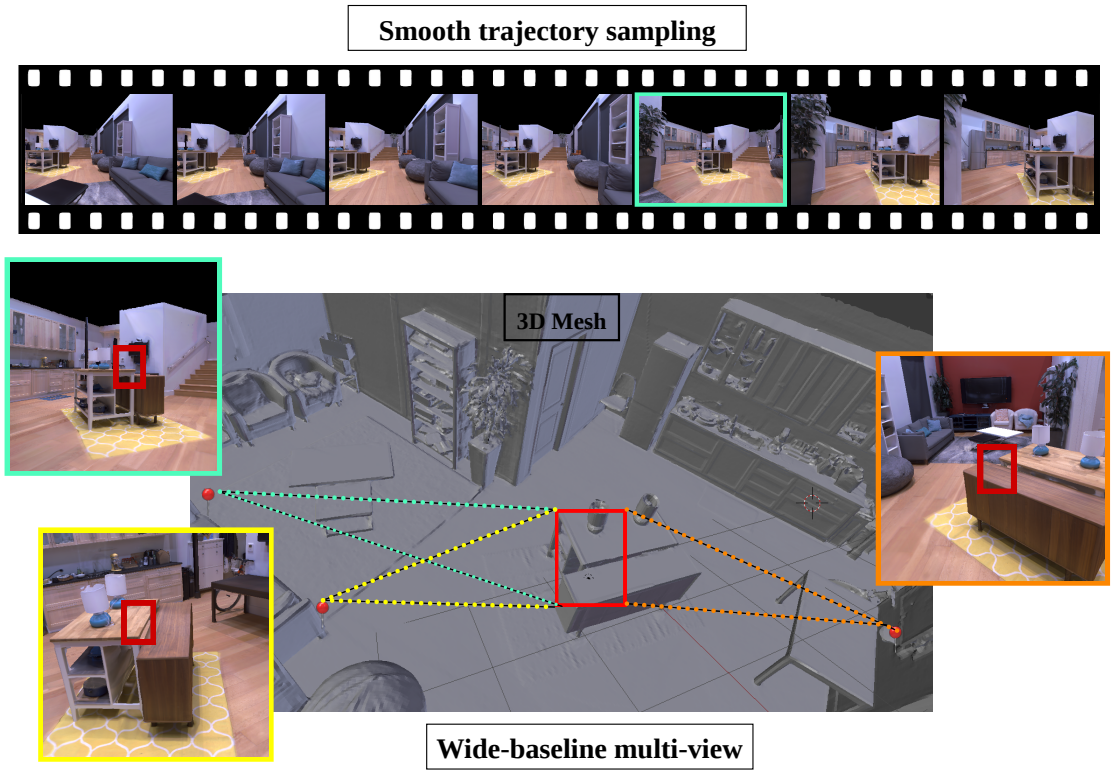


Figure 5.2: **Wide- and narrow-baseline dense view sampling.** Each point-of-interest can be viewed by a guaranteed minimum number of cameras. We also provide an option for creating denser views with narrower baselines (e.g. similar to consecutive video frames) that is crucial for inverse rendering methods..

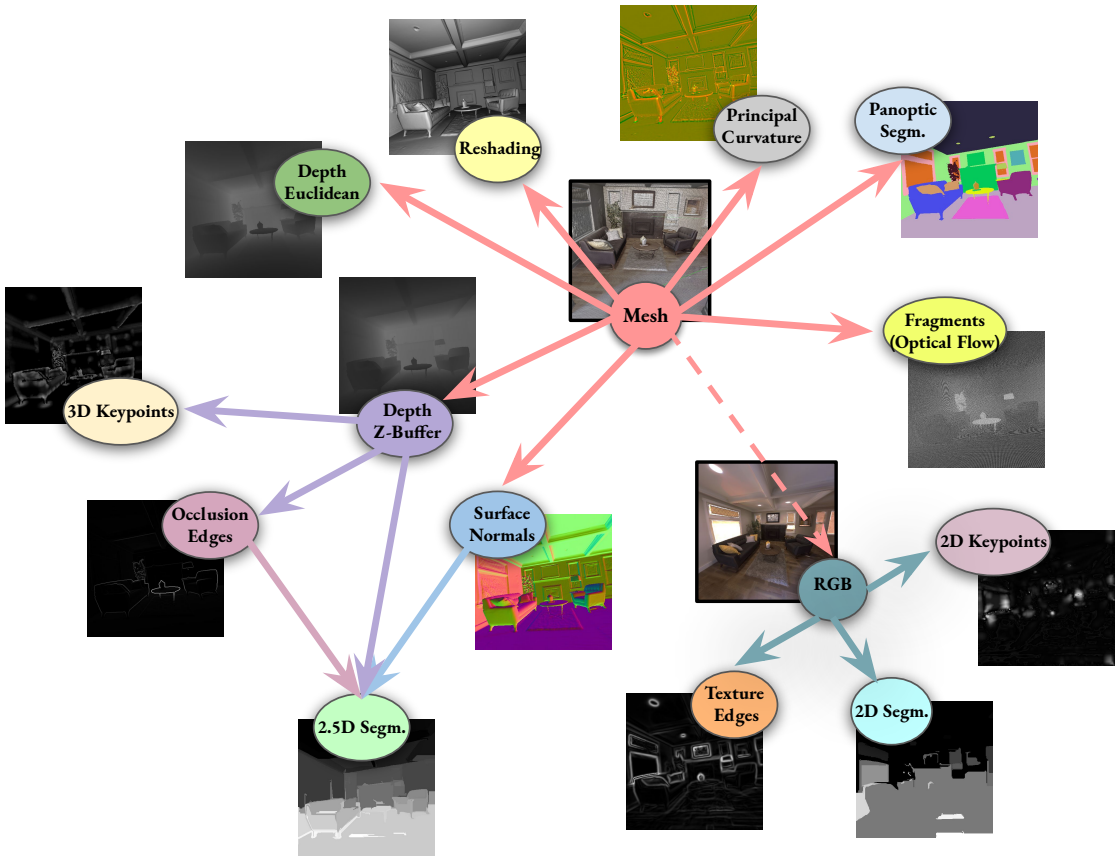


Figure 5.3: **DAG of processing pipeline.** The pipeline uses some of the mid-level cues to produce others. The ordering of this process (for image-like cues) is shown by the DAG.

Camera and Point Sampling: Camera locations can be provided (if the mesh comes with aligned RGB), or as in Fig. 5.1 II, the annotator generates cameras in each space so that cameras are not inside or overlapping with the mesh (default: cameras generated via Poisson-disc sampling to cover the space). Points-of-interest are then sampled from the mesh with a user-specified sampling strategy (default: uniform sampling of each mesh face and then uniform sampling on that face). Cameras and points are then filtered so that each camera sees at least one point and each point is seen by at least some user-specified minimum number of cameras (default: 3).

View Sampling: The annotator provides two default methods for generating views of each point. The first method (wide-baseline) generates images while the second, (smooth-trajectory mode) generates videos.

- *Wide-baseline multi-view:* A view is saved for each space-camera-point combination where there is an unobstructed line-of-sight between the camera center and the point-of-interest. The camera is fixated on the point-of-interest, as shown in Fig. 5.2, bottom.
- *Smooth trajectory sampling:* For each point of interest, a subset of cameras with a fixated view of the point are selected, and a smooth cubic-spline trajectory is interpolated between the cameras. Views of the point are generated for cameras at regular intervals along this trajectory (see Fig. 5.2, top).

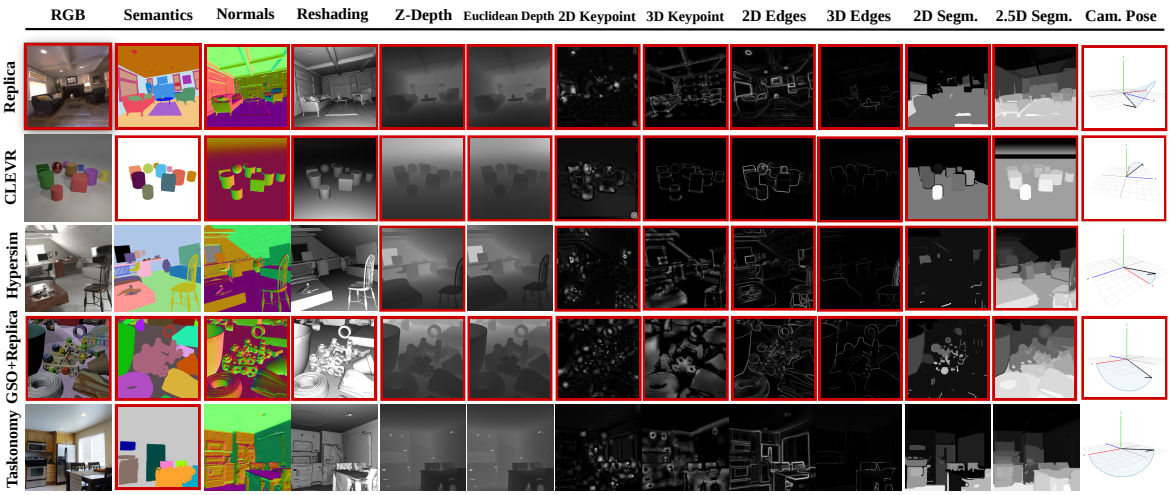


Figure 5.4: **Mid-level cues provided for the starter set.** 12 out of 21 mid-level cues visualized for each component dataset in the starter set, which contains both scenes and objects. Images with red borders indicate cues that were not included in the original data. Fig. 3.1 visualizes all 21 cues.

Rendering mid-level cues: Since no single piece of software was able to provide all mid-level cues, we created an interconnected pipeline connecting several different pieces of software that are all freely available and open-source. We tried to primarily use Blender (a 3D creation suite), since it has an active user and maintenance community, excellent documentation, and python bindings for almost everything. Used by professional animators and artists, it is is generally well-optimized. The overall pipeline is fairly complex, so we defer a full description to the supplementary. The order of cue generation is shown in Fig. 5.3. The full code is available on our website.
Performance: The annotator generates labels in any resolution. Each space+point+view+cue label in the starter dataset (512×512) typically takes 1-4 seconds on server or desktop CPUs and can be parallelized over many machines.

Ecosystem Tools

To simplify adoption, the following tools are available on our website and the associated GitHub repository:

Pipeline code and documentation.
Docker containing the annotator and properly linked software (Blender [45], compatible Python versions, MeshLab [42], etc.).
Dataloaders in PyTorch for correctly and efficiently loading the resulting dataset
Starter dataset of 14.5 million images with associated labels for each task
Convenience utilities for downloading and manipulating data and *automatically filtering* misaligned meshes (description and sensitivity analysis in the supplementary).
Pretrained models and code, including the first publicly available implementation of MiDaS [172] training code.

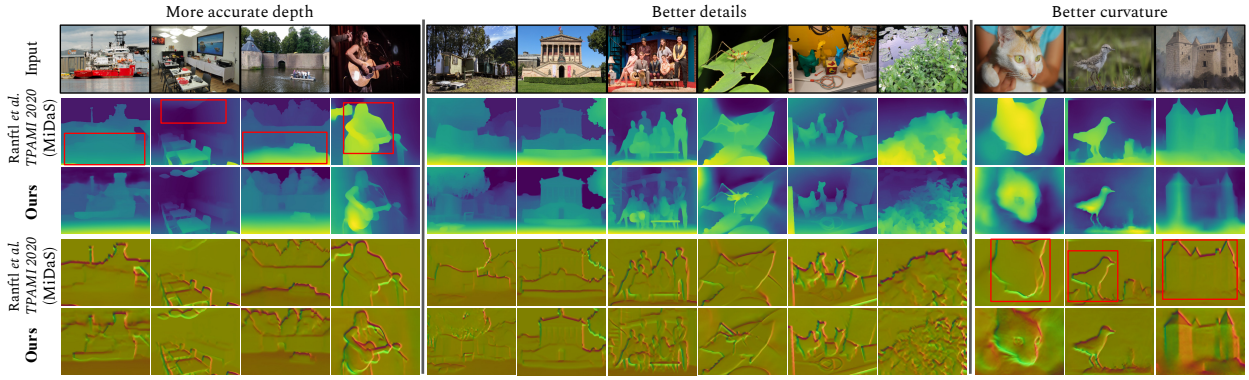


Figure 5.5: **Qualitative comparison with MiDaS on zero-shot OASIS depth estimation.** The last 2 rows show the surface normals extracted from the depth predictions. Our model predicts more accurate depth (left), and also outperforms in recovering the fine-grained details (middle). As shown by the extracted surface normal in the last 3 columns (right), our depth predictions better reflect the curvature and true shape of the objects, while the same regions appear flat in the predictions by the MiDaS model. The red rectangles highlight the regions useful for comparison [best viewed zoomed in].

5.4 Starter Dataset Overview

We provide a relatively large starter dataset of data annotated with the `Omnidata` annotator. The dataset comprises roughly **14.5 million** images from scenes that are both scene- and object-centric. Fig. 5.4 shows sample images from the starter dataset along with 12 of the 21 mid-level cues provided. Cues that are not present in the original dataset are indicated with a red border. We evaluate this starter dataset on existing benchmarks in Sec. 5.4. Note that the dataset could be straightforwardly extended to other existing outdoor and driving datasets such as GTA5 [175], CARLA [55], or Tanks and Temples [113].

Datasets Included

The starter data was created from 7 mesh-based datasets:
Indoor scene datasets: Replica [202], HyperSim [176], Taskonomy [238], Habitat-Matterport (HM3D)
Aerial/outdoor datasets: BlendedMVG [228]
Diagnostic/Structured datasets: CLEVR [104]
Object-centric datsets: To provide object-centric views in addition to scene-centric ones, we create a dataset of Google Scanned Objects [95] scattered around buildings from the Replica [202] dataset (similar to how ObjectNet [19] diversified images for image classification). We used the Habitat [137] environment to generate physically plausible scenes, and generated different densities of objects. Examples of images are shown in Fig. 5.4, and a full description of the generation process is available in the supplementary.

Dataset Statistics

The starter dataset contains **14,601,449** images from **2,414** spaces. Views are both scene- and object-centric, and they are labeled with each modality listed in Fig.3.1. Camera field-of-view is sampled from a truncated normal distribution between $30°$ and $125°$ with mean $77.5°$, and camera roll is uniform in $[-10°, 10°]$. Tab. 5.1 contains data broken down to sub-datasets.

	Images			Spaces			Points
Dataset	Train	Val	Test	Train	Val	Test	
CLEVR	60,000	6,000	6,000	1	0	0	72,000
Replica	56,783	23,725	23,889	10	4	4	4,150
Replica + GSO	107,404	43,450	42,665	10	4	4	31,167
Hypersim	59,543	7,386	7,690	365	46	46	74,619
Taskonomy	3,416,314	538,567	629,581	379	75	79	684,052
BlendedMVG	79,023	16,787	16,766	341	74	73	112,576
Habitat-Matterport	8,470,855	1,061,021	-	800	100	-	564,328
Total (no CLEVR)	12,189,922	1,690,936	720,591	1,905	303	206	1,434,892

Table 5.1: **Component dataset statistics.** Breakdown of train/val/test split sizes in each of the components of the starter datset.

Soundness for Existing Computer Vision

We demonstrate that the generated dataset is capable of training standard, modern vision systems to state-of-the-art performance on existing benchmarks. Once we have established that the models can be trusted, we further provide a few transfer experiments to quantify how related the different component datasets are.

We show that the models trained on a 5-dataset portion of the starter dataset (4M images) for depth and surface normal estimation have state-of-the-art performance on the in-the-wild OASIS benchmark. To demonstrate the effectiveness of the pipeline for semantic tasks, we show that the predictions from a network trained for panoptic segmentation on a smaller 3-dataset portion (1M images) are of similar quality to models trained on COCO [130]. Full experimental details and more results are available in the supplementary.

Method	Test Data	L1 Error (↓)	$\delta > 1.25$ (↓)	$\delta > 1.25^2$ (↓)	$\delta > 1.25^3$ (↓)
XTC [236]	OASIS [38]	1.180	85.28	71.86	60.22
MiDaSv3 [171]		0.8057	82.03	67.25	55.35
Omnidata		**0.7901**	**81.00**	**65.22**	**52.93**
XTC [236]	NYU [194]	0.5279	70.41	49.90	36.28
MiDaSv3 [171]		0.3838	63.84	41.65	28.97
Omnidata		**0.2878**	**51.73**	**30.98**	**20.86**

Table 5.2: **Zero-shot depth estimation.** On NYU and OASIS, a DPT-Hybrid trained on the Omnidata starter dataset is comparable or better then the same model trained on existing depth datasets.

Method	Training Data	Anglular Error°		% Within t°			Relative Normal	
		Mean	Median	11.25°	22.5°	30°	AUC_o	AUC_p
Hourglass [39]	OASIS [38]	**23.91**	18.16	**31.23**	59.45	**71.77**	0.5913	0.5786
Hourglass [39]	SNOW [37]	31.35	26.97	13.98	40.20	56.03	0.5329	0.5016
Hourglass [39]	NYU [193]	35.32	29.21	14.23	37.72	51.31	0.5467	0.5132
PBRS [241]	NYU [193]	38.29	33.16	11.59	32.14	45.00	0.5669	0.5253
UNet [179]	SunCG [196]	35.42	28.70	12.31	38.51	52.15	0.5871	0.5318
UNet [179]	**Omnidata**	24.87	**18.04**	31.02	**59.53**	71.37	**0.6692**	**0.6758**
Human (Approx.)	-	17.27	12.92	44.36	76.16	85.24	0.8826	0.6514

Table 5.3: **Zero-shot surface normal estimation on OASIS.** A UNet trained on the Omnidata starter dataset matched or outperformed models trained on OASIS itself, and it matched human-level AUC_p. Notice that the first row is not zero-shot since it's trained on OASIS.

Monocular depth estimation: The current best approach for depth estimation is to aggregate multiple smaller datasets and train with scale- and shift-invariant losses [172, 171] to handle the different unknown depth ranges and scales. As of this writing, the DPT-based [171] models from "MiDaS v3.0" [171] define the state-of-the-art, especially on NYU [194]. We adopt a similar setting to MiDaS v3.0, but train on a 5-dataset portion of our starter dataset instead of their 10-dataset mix[1].

As in [171], we evaluate zero-shot cross-dataset transfer with test predictions and GT aligned in scale and shift in inverse-depth space. Tab. 5.2 shows that the DPT-Hybrid trained on our starter dataset outperforms MiDaS DPT-Hybrid on both the test set of NYU [194] and the validation split of OASIS (the test GT is not available). The error metrics use $\delta = max(\frac{d}{d^*}, \frac{d^*}{d})$ where d and d^* are aligned depth and ground truth. Our model better recovers the fine-grained details and true shape of the objects—this is especially clear in the surface normals extracted from the predictions (last 2 rows of Fig. 5.5). Full details, code, and more qualitative results are available on our website.

[1]MiDaS v3.0 also uses MTAN [131] for dataset balancing, and though in Sec. 5.5 we examine MTAN (it indeed helped on our dataset), we used here a naive sampling strategy in order to be consistent with the majority of the other models in this paper.

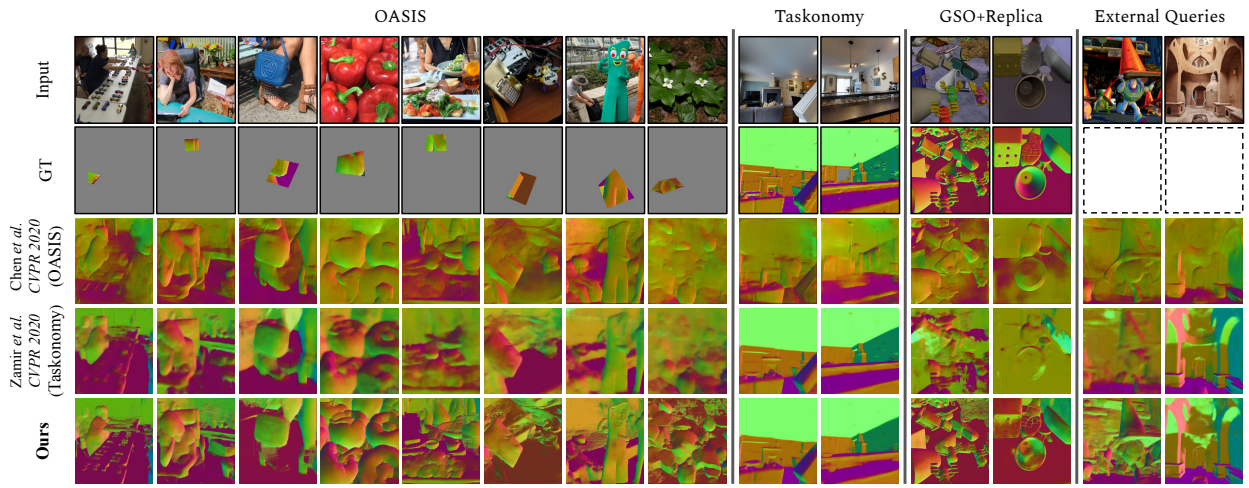


Figure 5.6: **Qualitative results of zero-shot surface normal estimation.** The 3 models are trained on OASIS[38], Full Taskonomy[236, 238], and our starter set. Queries are from 3 different datasets (OASIS, Taskonomy, GSO+Replica) in addition to some external queries in the last 2 columns (no ground truth available) which show the generalization of the models to external data [best viewed zoomed in].

Surface normal estimation: Similar to the existing models on the surface normal track of OASIS, we train a vanilla UNet [179] architecture (6 down/6 up, similar to [236]) with angular & L1 losses, light 2D data augmentation, and input resolutions between 256 and 512. We use Adam [109] with LR 10^{-4} & weight decay 2×10^{-6}. The results in Tab. 5.3 indicate that our model matched human-level performance on OASIS AUC_p. On most of the remaining metrics, it outperformed related models trained on other datasets (including OASIS itself) and models with architectures specifically designed for normals estimation (PBRS). Fig. 5.6 shows that our model qualitatively performs *much* better on selected images than is indicated by the numbers, which may be because the standard metrics do not align with perceptual quality as "uninteresting" areas (walls, floors) dominate the score [38]. Further details and results are available in the supplementary.

Panoptic segmentation: To demonstrate the pipeline's ability to train models for non-geometric tasks, we train a PanopticFPN [110] on a 3-dataset subset of our starter dataset. Fig. 5.7 shows that on in-the-wild images of indoor buildings, the resulting model is of similar quality to one trained on COCO [130] (an extensive manually labeled dataset). Quantitative results, full experimental details, and code are available on our website.

Figure 5.7: **Qualitative results of panoptic segmentation with PanopticFPNs [110] trained on COCO [130] and `Omnidata`.** The `Omnidata` model trained jointly on Taskonomy, Replica, and Hypersim shows good out-of-distribution performance on indoor scenes without people.

Dataset Relatedness

To estimate how the components of the starter dataset are related, we use zero-shot cross-dataset transfer performance for surface normal and panoptic segmentation models trained on different components. Tab. 5.4 shows that each single model performs well on its corresponding test set, but typically generalizes poorly. The models trained on larger splits perform better overall (see supplementary). The model trained on the largest set achieved the best average performance (harmonic mean 25.8% and 30.3% better than best single-dataset models for surface normal estimation and panoptic segmentation). The ranking of transfers depended on the task, which might be due to the sparse panoptic labels on Taskonomy (from the followup paper [11]), but we believe the dependency is true in general.

	Surface normal estimation: L1 Error (↓)						Panoptic Quality (PQ) (↑)			
Train/Test	Taskonomy	Replica	Hypersim	Replica+GSO	BlendedMVG	h. mean	Taskonomy*	Replica	Hypersim	h. mean
Taskonomy*	**4.85**	7.76	8.69	13.89	15.55	8.53	8.39	3.95	11.67	6.55
Replica	9.36	**3.98**	11.78	10.28	15.02	8.24	1.01	**41.97**	4.50	2.43
Hypersim	7.28	7.57	**6.72**	11.34	12.94	8.56	**9.35**	14.08	25.39	13.80
Replica+GSO	13.88	4.94	15.05	**5.17**	14.03	8.26	-	-	-	-
BlendedMVG	17.1	14.23	16.93	14.87	**8.85**	13.58	-	-	-	-
`Omnidata`	5.32	4.24	6.53	6.45	11.53	**6.11**	9.14	41.24	**30.16**	**17.98**

Table 5.4: **Inter-dataset domain transfer performance for surface normal estimation and panoptic segmentation.** Models trained on each individual dataset and `Omnidata` are evaluated on test splits of the starter set. The harmonic mean across datasets is shown in the last column. (* PQ on *things* classes only, as Taskonomy does not feature *stuff* labels.)

5.5 Illustrative Data-Focused Analyses

Now that we have established that the annotator produces datasets capable of training reliable models, what analyses can we do with such datasets? We survey a few examples here, but they are not intended to be comprehensive (Sec. 5.1).

New 3D Data Augmentations

Data augmentation is used to address shortcomings in model performance and robustness. For example, models trained only on images captured with narrow apertures (e.g. NYU or Taskonomy) tend to perform poorly on images taken with a wide aperture (i.e. strong depth-of-field), and augmenting with 2D Gaussian blur is used to improve model performance on unfocused portions of images. The approach is common enough that 2D blur was included in the Common Corruptions benchmark [85]. Because the full scene geometry is available for our starter dataset, it is possible to do the *data augmentation in 3D* (image refocusing) instead of 2D (flat blurring). Fig. 5.8 shows an example of what 3D "image refocusing" augmentation on our dataset looks like. In the supplementary, we show that models trained for surface normal estimation using only 3D augmentation were more robust to both 2D blurring and 3D refocusing than those trained with 2D augmentation.

Figure 5.8: **Image refocusing augmentation on Taskonomy.** Portions of the image that are in focus are highlighted in red [best viewed zoomed in].

Mid-Level Cues as Inputs: Are They Useful?

Is there an advantage in using multiple sensors or non-RGB representations of the environment? Instead of predicting mid-level cues as the downstream task (i.e. multi-task learning), multiple

cues could also be used as inputs (if relevant sensors are available) or specified as intermediate representation (with the labels being used as supervision only during training, i.e. PADNet [225]).

Tab. 5.5 demonstrates that using these additional cues in the latter 2 ways can improve performance on the original test set and also on unseen data. In this experiment, we train HRNet-18 [205] backbones for semantic segmentation using a single component dataset (10 spaces from Replica) and evaluate them on Replica, Hypersim and Taskonomy (tiny split). Relative to using only RGB inputs and the semantic segmentation labels, cross-entropy performance improves across the board when treating the cues as sensors (23%, 34% and 30%) or using them as intermediate representations (13%, 17%, and 19%). Adding more cues seems to help. Full experimental settings in the supplementary.

Future work could further analyze how the effectiveness of these different methods change with dataset size, which cues to use, how many additional images a mid-level cue is worth, and on the relative importance of getting more data from the same scene vs. adding data from new scenes.

Input/Supervised Domains	GT Mid-Level			Predicted Mid-Level		
	Cross-Entropy (↓)			Cross-Entropy (↓)		
	Repl.	H.Sim	Task.	Repl.	H.Sim	Task.
RGB	0.61	5.87	7.55	0.61	5.87	7.55
(All Above) + Normals	0.47	4.47	6.12	0.61	5.44	7.12
(All Above) + 3D Edges	**0.46**	4.47	6.75	0.54	5.06	6.49
(All Above) + (2D Edges, Z-Depth, 3D Keypts)	**0.46**	**3.86**	**6.04**	**0.53**	**4.9**	**6.13**

Table 5.5: **Utility of mid-level cues.** The table shows semantic segmentation results using models trained on Replica. The models (except for "RGB") received (either predicted or GT) mid-level cues in their input in addition to the RGB. The results show they notably benefited from the mid-level cues.

Systematic Evaluation of Multi-Task Learning

Recent work [212] shows that existing MTL techniques for computer vision appear to be specialized to their development setting, and in general they do not outperform single-task or shared-encoder approaches on novel datasets or tasks. We extend those results for additional tasks (3D Keypoints) and add numbers on our dataset as a comparison point. Specifically, we follow [212] and train models for a fixed set of tasks (semantic segmentation, 3D keypoints, depth z-buffer, and occlusion edges) using different MTL methods (Tab. 5.6). On a 3-dataset split of the starter dataset, some methods naturally perform better and others do worse. One might hope that the ordering of these methods would be the same on different tasks (semantic segmentation vs 3D Keypoints), or at least when training for those same tasks on a different dataset (NYU [194], CityScapes [46], or Taskonomy [238]). Yet, Tab. 5.6 shows that MTL methods display no clear ranking in either case (i.e. Spearman's ρ was always indistinguishable from 0). Ignoring the lack of significance, the cross-dataset correlation was still weak ($\rho < 0.45$), and methods performance was actually *anti-correlated* across tasks (ρ=-0.4), suggesting that the models are indeed specialized to specific tasks. This anti-correlation was true even when controlling for dataset.

Given that current MTL approaches do not outperform single-task baselines, predicting different mid-level cues poses a challenging setting for MTL. The `Omnidata` pipeline provides an avenue to create large and diverse multi-task mid-level benchmarks that could more systematically and reliably evaluate progress in multi-task learning.

Method	Semantic Segmentation								3D Keypoints			
	Ours		NYU [194]		CityScapes. [211]		Taskonomy [211]		Ours		Taskonomy [211]	
	IoU (↑)	Rank	IoU (↑)	Rank	IoU (↑)	Rank	IoU (↑)	Rank	L1 (↓)	Rank	L1 (↓)	Rank
Single task	85.12	1	90.69	2	65.2	1	43.5	4	0.0439	4	0.23	1
MTL baseline	81.82	3	90.63	3	61.5	4	47.8	1	0.0429	3	0.34	2
MTAN [131]	83.00	2	91.11	1	62.8	3	43.8	3	0.0426	1	0.4	3
Cross-stitch [145]	80.69	4	90.33	4	65.1	2	44.0	2	0.0427	2	0.50	4
Spearman's ρ	Within task.: ρ=0.43. Between segm.-3D keypts.: ρ=-0.4								Within task: ρ=0.2.			

Table 5.6: **Multi-task training methods do not show a clear ordering.** Within-task, rankings between different methods were indistinguishable from random orderings (i.e. ρ=0). Between tasks, rankings on Sem. Seg. were anti-correlated with rank on the 3D Keypts (ρ=-0.4). Both conclusions were strengthened after controlling for training setups.

5.6 Conclusion and Limitations

This paper introduces a pipeline to create steerable datasets from comprehensive scans of the environment. The resulting multi-task datasets can be large and diverse, and realistic enough that models trained on the data perform well in the real world. To demonstrate this, we annotated an example dataset and used it to train a few standard vision methods to state-of-the-art performance on multiple computer vision tasks. We believe this is capability is useful on its own, especially since it acts as a bridge between real-world 3D scans, simulators, and static vision datasets.

Our main intention for this tool is to better study properties of vision datasets, and their interaction with models and tasks. Crucially, the fact that the pipeline can be used to train strong models in real-world settings gives us hope that findings stemming from this pipeline here might hold true more generally. In particular, we believe that this "steerable dataset" method could bear fruit in fundamental lines of research such as how data sampling strategies and choice of cue/sensor impact representations and model reliability.

To close, we discuss some of important limitations of this pipeline and possibilities for future lines of work.

1. **Studying steerability.** The Omnidata pipeline provides a method to create steerable datasets, but we did not present any analysis of the effects of tuning the different steering 'knobs'. I.e. our starter dataset used a fixed choice of generation settings and we did no tuning on that initial choice. Clearly, such choices do have important effects on the dataset (e.g. see our online demo) and on the resulting models [211, 210, 17, 224]). We believe rich insights lie in this direction, which is why we created this pipeline. This paper only provides a few sporadic experiments to illustrate the general idea, it does not represent a systematic study.

2. **Limited capture information.** The 3D scans used in this paper come from the output of standard structure-from-motion methods that stitch together many overlapping images from RGB and depth sensors. These scans are represented as meshes (with textures or aligned RGB images), but this representation leaves out important information about the scene. For example, the materials lack reflectance models (e.g. BRDF) and there is no information about scene lighting. Moreover, scans usually have limited reconstruction accuracy (e.g. commonly up to 2cm error in Taskonomy), which affects both the texture quality and the quality of the generated labels. Better sensing technology (e.g. light-field cameras, higher-resolution depth sensors), as well as algorithmic improvements (e.g. NeRF, below) can add more dimensions of control and reduce the gap between the resampled and real cues/images.

3. **How to represent the 'complete capture'.** The Omnidata pipeline uses 3D meshes to represent the scene, and samples images using that representation. Other representations, such as using light-field cameras and NeRF [143] could be used as implicit representations of the scene and similarly used for resampling the scene. The surprising effectiveness of NeRF makes this direction quite compelling.

4. **Limited number of mid-level cues.** The initial release of the Omnidata annotator provides 21 mid-level cues. Like most tasks in computer vision, the current mid-level cues are based more on human intuition than on demonstrably predictive theories of vision. As computer vision and vision science make new advancements, these can be integrated to the sampling pipeline as long as the required information is present in the capture information (e.g. new cues and augmentations).

Chapter 6

Conclusion

This thesis has made several original contributions towards the development and application of pretrained visual representations for Embodied AI. By addressing the challenges of learning efficiency, skill generalization, and robustness to new settings, the work presented in Chapters 2, 3, and 4 offers valuable insights into the types of pretraining objectives that are most effective in this domain. Additionally, the method for leveraging 3D data from the internet and prior robot experiences presented in Chapter 5 contributes to the creation of large, diverse datasets necessary for producing accurate and robust models.

Despite these advances, there are important challenges that must still be addressed before robots are capable of autonomous behavior in home robotics settings. These challenges include the development of multi-view representations, multisensory representations, action trajectory datasets, and long-horizon planning.

Multi-view representations: Our world is inherently 3D (or even 4D), yet the majority of representations in this thesis are "monocular" and process each image independently. The training objectives are primarily 2D (image space) or 2.5D (depth images). How can we represent multiple views of the same scene, either captured simultaneously by multiple cameras or over time by the same camera? In the coming years, significant progress is anticipated as both algorithms and datasets improve.

Chapter 5 introduces a large-scale multi-view dataset suitable for training these enhanced architectures. Given the ongoing development of wearable AR/VR headsets, it is expected that many more extensive multi-view datasets will emerge. On the algorithmic front, two promising developments include the transformer [214, 54] architecture, which flexibly parses inputs into variable-length sets of tokens for inference, and Neural Radiance Fields (NeRF)[143], which combines scene-specific memory (e.g., spherical harmonic position embeddings or a multiresolution hash table[151]) with a decoder shareable across scenes [232]. These innovations offer a powerful way to combine generalizable multi-view learning with short-term 3D memory.

Multisensory representations: The representations in this paper are predominantly *visual*. However, animals possess multiple senses, and many lack vision entirely. Some species, like cave-dwelling Mexican tetras, lose their vision within a few generations. Yet, I'm not aware of any animals that have lost their sense of touch. Developing touch representations and integrating multiple

senses (e.g., touch, vision, and proprioception) is crucial, as animals often use multiple senses to perform behaviors [148]. Recent self-supervised techniques and transformer architectures provide an elegant way to integrate multiple modalities and views [168, 167]. However, comprehensive multimodal datasets are still needed.

Action trajectory datasets: Sensing is always in service of downstream decisions, but current representation learning seldom considers downstream goals or actions. How can we learn representations better suited for decision-making and action execution? Large "trajectory" datasets of agents acting in environments are likely necessary for successful large-scale pretraining in Embodied AI. Few such datasets exist, but they typically fall into three categories:

1. Collecting numerous on-robot demonstrations (e.g., [27])—often expert trajectories from human teleoperation.

2. Gathering extensive data of humans in real-world situations (e.g., Ego4D [74]) and mapping those actions to robot actions in a common space.

3. Simulating vast amounts of data using a world model and computing "expert" trajectories optimal for that model.

A robust solution likely requires all three. In particular, human trajectories seem a natural and easy way to demonstrate desired robot behaviors.

However, one big lesson from the successes of deep learning has been in the importance of diverse and broad pretraining data. Though there have been several high-profile attempts to create such large-scale trajectory datasets, from multiple groups–all existing trajectory datasets are exceedingly narrow. They use either a handful of objects (on the order of 5-100), in a handful of settings (usually 2-10 buildings), and the robot placement and work surface is usually flat and tightly controlled. As a result, most pretraining in robotics fails to generalize to new scenes.

So the near future, I expect that we will see great progress from combining large-scale scene datasets with large-scale object datasets and generating expert trajectories in simulation. Then using that as pretraining (either with behavior cloning or subsequent fine-tuning). This could also be combined with human-guided expert trajectories in simulation, which might also be easier to collect than similar data from the real-world.

Long-horizon planning: This thesis focuses more on low-level "atomic" skills than long-term planning. Low-level control is often challenging, while long-term planning can be surprisingly straightforward (Moravec's paradox). Remarkable progress in large language models (LLMs) [28] and their application to robot planning [3] makes me optimistic about the ability for robots to do medium- and long-term planning in the near future. And since most current work in this area does not leverage pretrained sensory representations, improving sensory representations and low-level skills could provide significant advancements in the ability to successfully execute long-term behavior.

Bibliography

[1] E. H. Adelson and A. P. Pentland. "The perception of shading and reflectance". In: *Perception as Baysian Inference* (1996), pp. 409–423.

[2] Pulkit Agrawal et al. "Learning to Poke by Poking: Experiential Learning of Intuitive Physics". In: *CoRR* abs/1606.07419 (2016). arXiv: 1606.07419. URL: http://arxiv.org/abs/1606.07419.

[3] Michael Ahn et al. "Do as i can, not as i say: Grounding language in robotic affordances". In: *arXiv preprint arXiv:2204.01691* (2022).

[4] Ilge Akkaya et al. "Solving rubik's cube with a robot hand". In: *arXiv preprint arXiv:1910.07113* (2019).

[5] Dario Amodei et al. "Concrete Problems in AI Safety". In: *CoRR* abs/1606.06565 (2016). arXiv: 1606.06565. URL: http://arxiv.org/abs/1606.06565.

[6] Peter Anderson et al. "On Evaluation of Embodied Navigation Agents". In: *CoRR* abs/1807.06757 (2018). arXiv: 1807.06757. URL: http://arxiv.org/abs/1807.06757.

[7] Peter Anderson et al. "On evaluation of embodied navigation agents". In: *arXiv preprint arXiv:1807.06757* (2018).

[8] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[9] OpenAI: Marcin Andrychowicz et al. "Learning dexterous in-hand manipulation". In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[10] Brenna D. Argall et al. "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469–483. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2008.10.024. URL: https://www.sciencedirect.com/science/article/pii/S0921889008001772.

[11] Iro Armeni et al. "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[12] Iro Armeni et al. "3d semantic parsing of large-scale indoor spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1534–1543.

[13] Iro Armeni et al. "Joint 2D-3D-Semantic Data for Indoor Scene Understanding". In: *arXiv preprint arXiv:1702.01105* (2017).

[14] Iro Armeni et al. “Joint 2D-3D-Semantic Data for Indoor Scene Understanding”. In: *CoRR* abs/1702.01105 (2017). arXiv: 1702.01105. URL: http://arxiv.org/abs/1702.01105.

[15] Mikel Artetxe and Holger Schwenk. “Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond”. In: *CoRR* abs/1812.10464 (2018). arXiv: 1812.10464. URL: http://arxiv.org/abs/1812.10464.

[16] Aharon Azulay and Yair Weiss. “Why do deep convolutional networks generalize so poorly to small image transformations?” In: *arXiv preprint arXiv:1805.12177* (2018).

[17] Aharon Azulay and Yair Weiss. “Why do deep convolutional networks generalize so poorly to small image transformations?” In: *CoRR* abs/1805.12177 (2018). arXiv: 1805.12177. URL: http://arxiv.org/abs/1805.12177.

[18] Somil Bansal et al. “Combining Optimal Control and Learning for Visual Navigation in Novel Environments”. In: *CoRR* abs/1903.02531 (2019). arXiv: 1903.02531. URL: http://arxiv.org/abs/1903.02531.

[19] Andrei Barbu et al. “Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models”. In: (2019).

[20] J. T. Barron and J. Malik. “Shape, Illumination, and Reflectance from Shading”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.8 (Aug. 2015), pp. 1670–1687. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2377712.

[21] Jonathan Baxter. “A model of inductive bias learning”. In: *Journal of artificial intelligence research* 12 (2000), pp. 149–198.

[22] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[23] J. C. Bazin et al. “Globally optimal line clustering and vanishing point estimation in Manhattan world”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 638–645. DOI: 10.1109/CVPR.2012.6247731.

[24] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[25] Yoav Benjamini and Yosef Hochberg. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1 (1995), pp. 289–300. ISSN: 00359246. URL: http://www.jstor.org/stable/2346101.

[26] Richard W. Brislin. “Back-Translation for Cross-Cultural Research”. In: *Journal of Cross-Cultural Psychology* 1.3 (1970), pp. 185–216. DOI: 10.1177/135910457000100301. eprint: https://doi.org/10.1177/135910457000100301. URL: https://doi.org/10.1177/135910457000100301.

[27] Anthony Brohan et al. “Rt-1: Robotics transformer for real-world control at scale”. In: *arXiv preprint arXiv:2212.06817* (2022).

[28] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[29] Berk Calli et al. “Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols”. In: *arXiv preprint arXiv:1502.03143* (2015).

[30] John Canny. “A computational approach to edge detection”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[31] Zhe Cao et al. “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *arXiv preprint arXiv:1812.08008* (2018).

[32] Marcela Carvalho et al. “Deep Depth from Defocus: how can defocus blur improve 3D estimation using dense neural networks?” In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[33] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. “Anomaly detection using one-class neural networks”. In: *arXiv preprint arXiv:1802.06360* (2018).

[34] Angel Chang et al. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *International Conference on 3D Vision (3DV)* (2017).

[35] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[36] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[37] Weifeng Chen, Donglai Xiang, and Jia Deng. “Surface Normals in the Wild”. In: *CoRR* abs/1704.02956 (2017). arXiv: 1704.02956. URL: http://arxiv.org/abs/1704.02956.

[38] Weifeng Chen et al. *OASIS: A Large-Scale Dataset for Single Image 3D in the Wild*. 2020. arXiv: 2007.13215 [cs.CV].

[39] Weifeng Chen et al. “Single-Image Depth Perception in the Wild”. In: *CoRR* abs/1604.03901 (2016). arXiv: 1604.03901. URL: http://arxiv.org/abs/1604.03901.

[40] Bryan Chen* et al. “Robust Policies via Mid-Level Visual Representations: An Experimental Study in Manipulation and Navigation”. In: *4th Annual Conference on Robot Learning, CoRL 2020*. Proceedings of Machine Learning Research. PMLR, 2020. URL: https://arxiv.org/abs/2011.06698.

[41] Yunjey Choi et al. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8789–8797.

[42] Paolo Cignoni et al. “Meshlab: an open-source mesh processing tool.” In: *Eurographics Italian chapter conference*. Vol. 2008. Salerno, Italy. 2008, pp. 129–136.

[43] Felipe Codevilla et al. “On Offline Evaluation of Vision-based Driving Models”. In: *CoRR* abs/1809.04843 (2018). arXiv: 1809.04843. URL: http://arxiv.org/abs/1809.04843.

[44] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[45] BO Community. “Blender–a 3D modelling and rendering package.” In: (2018).

[46] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[47] Luca Cosmo et al. “Consistent partial matching of shape collections via sparse modeling”. In: *Computer Graphics Forum*. Vol. 36. 1. Wiley Online Library. 2017, pp. 209–221.

[48] James M. Coughlan and Alan L Yuille. “The Manhattan World Assumption: Regularities in Scene Statistics which Enable Bayesian Inference”. In: *Advances in Neural Information Processing Systems 13*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 845–851. URL: http://papers.nips.cc/paper/1804-the-manhattan-world-assumption-regularities-in-scene-statistics-which-enable-bayesian-inference.pdf.

[49] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[50] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[51] Maximilian Denninger et al. “BlenderProc”. In: *arXiv preprint arXiv:1911.01911* (2019).

[52] Carl Doersch, Abhinav Gupta, and Alexei A Efros. “Unsupervised visual representation learning by context prediction”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1422–1430.

[53] Alexey Dosovitskiy and Vladlen Koltun. “Learning to Act by Predicting the Future”. In: *CoRR* abs/1611.01779 (2016). arXiv: 1611.01779. URL: http://arxiv.org/abs/1611.01779.

[54] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR* (2021).

[55] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[56] Ralf Dragon and Luc Van Gool. “Ground plane estimation using a hidden markov model”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2014, pp. 4026–4033.

[57] Debidatta Dwibedi et al. “Temporal Cycle-Consistency Learning”. In: *CoRR* abs/1904.07846 (2019). arXiv: 1904.07846. URL: http://arxiv.org/abs/1904.07846.

[58] Sergey Edunov et al. “Understanding Back-Translation at Scale”. In: *CoRR* abs/1808.09381 (2018). arXiv: 1808.09381. URL: http://arxiv.org/abs/1808.09381.

[59] Ainaz Eftekhar et al. “Omnidata: A Scalable Pipeline for Making Multi-Task Mid-Level Vision Datasets From 3D Scans”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10786–10796.

[60] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. 2nd. Springer Publishing Company, Incorporated, 2015. ISBN: 3662448734.

[61] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth Map Prediction from a Single Image using a Multi-Scale Deep Network”. In: *CoRR* abs/1406.2283 (2014). arXiv: 1406.2283. URL: http://arxiv.org/abs/1406.2283.

[62] Chelsea Finn et al. “Deep spatial autoencoders for visuomotor learning”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 512–519.

[63] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. “Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation”. In: *arXiv preprint arXiv:1806.08756* (2018).

[64] Michel. Foucault and Alan Sheridan. *Discipline and punish : the birth of the prison / Michel Foucault ; translated from the French by Alan Sheridan*. English. Penguin Harmondsworth, 1979, [xi], 333 p., 8 p. of plates : ISBN: 0140551972.

[65] Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *CoRR* abs/1802.09477 (2018). arXiv: 1802.09477. URL: http://arxiv.org/abs/1802.09477.

[66] Ravi Garg et al. “Unsupervised cnn for single view depth estimation: Geometry to the rescue”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 740–756.

[67] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[68] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias/Variance Dilemma”. In: *Neural Computation* 4.1 (Jan. 1992), pp. 1–58. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.1.1.

[69] Georgios Georgakis et al. “End-to-end learning of keypoint detector and descriptor for pose invariant 3D matching”. In: *CoRR* abs/1802.07869 (2018). arXiv: 1802.07869. URL: http://arxiv.org/abs/1802.07869.

[70] J. Gibson. “The Senses Considered As Perceptual Systems”. In: 1966.

[71] R. Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2014.81.

[72] Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[73] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. “Unsupervised monocular depth estimation with left-right consistency”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 270–279.

[74] Kristen Grauman et al. “Ego4D: Around the World in 3,000 Hours of Egocentric Video”. In: *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*. 2022.

[75] Jean-Bastien Grill et al. *Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning*. 2020. arXiv: 2006.07733 [cs.LG].

[76] V. Guillemin and A. Pollack. *Differential Topology*. Mathematics Series. Prentice-Hall, 1974. ISBN: 9780132126052. URL: https://books.google.com/books?id=CmbwAAAAMAAJ.

[77] Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[78] Abhinav Gupta et al. “Robot learning in homes: Improving generalization and reducing dataset bias”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9094–9104.

[79] Shir Gur and Lior Wolf. “Single image depth estimation trained via depth from defocus cues”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7683–7692.

[80] Inc. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2016. URL: http://www.gurobi.com.

[81] Nicklas Hansen et al. “Self-Supervised Policy Adaptation during Deployment”. In: *arXiv preprint arXiv:2007.04309* (2020).

[82] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[83] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2019. arXiv: 1911.05722 [cs.CV].

[84] Olivier J. Hénaff et al. “Data-Efficient Image Recognition with Contrastive Predictive Coding”. In: *CoRR* abs/1905.09272 (2019). arXiv: 1905.09272. URL: http://arxiv.org/abs/1905.09272.

[85] Dan Hendrycks and Thomas Dietterich. “Benchmarking neural network robustness to common corruptions and perturbations”. In: *arXiv preprint arXiv:1903.12261* (2019).

[86] Aaron Hertzmann et al. “Image analogies”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 327–340.

[87] Steven Hickson et al. “Floors are Flat: Leveraging Semantics for Real-Time Surface Normal Prediction”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[88] Irina Higgins et al. “DARLA: Improving Zero-Shot Transfer in Reinforcement Learning”. In: *arXiv e-prints*, arXiv:1707.08475 (July 2017), arXiv:1707.08475. arXiv: 1707.08475 [stat.ML].

[89] G. E. Hinton and R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: http://science.sciencemag.org/content/313/5786/504.full.pdf. URL: http://science.sciencemag.org/content/313/5786/504.

[90] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[91] Judy Hoffman et al. “CyCADA: Cycle-Consistent Adversarial Domain Adaptation”. In: *CoRR* abs/1711.03213 (2017). arXiv: 1711.03213. URL: http://arxiv.org/abs/1711.03213.

[92] Qi-Xing Huang and Leonidas Guibas. “Consistent Shape Maps via Semidefinite Programming”. In: *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. SGP ’13. Genova, Italy: Eurographics Association, 2013, pp. 177–186. DOI: 10.1111/cgf.12184. URL: http://dx.doi.org/10.1111/cgf.12184.

[93] Xinyu Huang et al. “The ApolloScape Dataset for Autonomous Driving”. In: *CoRR* abs/1803.06184 (2018). arXiv: 1803.06184. URL: http://arxiv.org/abs/1803.06184.

[94] Junhwa Hur and Stefan Roth. “MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation”. In: *CoRR* abs/1708.05355 (2017). arXiv: 1708.05355. URL: http://arxiv.org/abs/1708.05355.

[95] *Ignition App*. URL: https://app.ignitionrobotics.org/GoogleResearch/fuel/collections/Google%20Scanned%20Objects.

[96] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CoRR* abs/1611.07004 (2016). arXiv: 1611.07004. URL: http://arxiv.org/abs/1611.07004.

[97] Stephen James, Marc Freese, and Andrew J. Davison. “PyRep: Bringing V-REP to Deep Robot Learning”. In: *CoRR* abs/1906.11176 (2019). arXiv: 1906.11176. URL: http://arxiv.org/abs/1906.11176.

[98] Stephen James et al. “RLBench: The Robot Learning Benchmark & Learning Environment”. In: (Sept. 26, 2019). URL: https://arxiv.org/abs/1909.12271v1 (visited on 07/21/2020).

[99] Stephen James et al. “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12627–12637.

[100] Redhwan Jamiruddin et al. “RGB-Depth SLAM Review”. In: *CoRR* abs/1805.07696 (2018). arXiv: 1805.07696. URL: http://arxiv.org/abs/1805.07696.

[101] Zequn Jie et al. “Left-Right Comparative Recurrent Model for Stereo Matching”. In: *CoRR* abs/1804.00796 (2018). arXiv: 1804.00796. URL: http://arxiv.org/abs/1804.00796.

[102] Jason Jo and Yoshua Bengio. “Measuring the tendency of CNNs to learn surface statistical regularities”. In: *arXiv preprint arXiv:1711.11561* (2017).

[103] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155.

[104] Justin Johnson et al. “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2901–2910.

[105] Michael I. Jordan and David E. Rumelhart. “Forward Models: Supervised Learning with a Distal Teacher”. In: *Cognitive Science* 16 (1992), pp. 307–354.

[106] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://doi.org/10.1038/s41586-021-03819-2.

[107] Michal Kempka et al. “ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning”. In: *arXiv preprint arXiv:1605.02097* (2016).

[108] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[109] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[110] Alexander Kirillov et al. “Panoptic Feature Pyramid Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 6392–6401.

[111] Alexander Kirillov et al. “Panoptic Segmentation”. In: *CoRR* abs/1801.00868 (2018). arXiv: 1801.00868. URL: http://arxiv.org/abs/1801.00868.

[112] Arno Knapitsch et al. “Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction”. In: *ACM Transactions on Graphics* 36.4 (2017).

[113] Arno Knapitsch et al. “Tanks and temples: Benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–13.

[114] Jan Knopp et al. “Hough Transform and 3D SURF for Robust Three Dimensional Classification”. In: *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 589–602. ISBN: 978-3-642-15567-3. DOI: 10.1007/978-3-642-15567-3_43. URL: https://doi.org/10.1007/978-3-642-15567-3_43.

[115] Mykel J. Kochenderfer et al. *Decision Making Under Uncertainty: Theory and Application*. 1st. The MIT Press, 2015. ISBN: 0262029251, 9780262029254.

[116] Iasonas Kokkinos. “UberNet: Training a ’Universal’ Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory”. In: *CoRR* abs/1609.02132 (2016). arXiv: 1609.02132. URL: http://arxiv.org/abs/1609.02132.

[117] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 0262013193, 9780262013192.

[118] Hui Kong, J. Y. Audibert, and J. Ponce. “Vanishing point detection for road detection”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 96–103. DOI: 10.1109/CVPR.2009.5206787.

[119] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[120] Uday Kusupati et al. “Normal Assisted Stereo Depth Estimation”. In: *arXiv preprint arXiv:1911.10444* (2019).

[121] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “The Omniglot challenge: a 3-year progress report”. In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 97–104.

[122] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6402–6413.

[123] John Lambert et al. “MSeg: a composite dataset for multi-domain semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2879–2888.

[124] Guillaume Lample and Alexis Conneau. “Cross-lingual Language Model Pretraining”. In: *CoRR* abs/1901.07291 (2019). arXiv: 1901.07291. URL: http://arxiv.org/abs/1901.07291.

[125] Nevena Lazic et al. “Data center cooling using model-predictive control”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 3814–3823.

[126] Yann LeCun, Sumit Chopra, and Raia Hadsell. “A tutorial on energy-based learning”. In: (2006).

[127] Sergey Levine et al. “End-to-End Training of Deep Visuomotor Policies”. In: *CoRR* abs/1504.00702 (2015). arXiv: 1504.00702. URL: http://arxiv.org/abs/1504.00702.

[128] Sergey Levine et al. “End-to-End Training of Deep Visuomotor Policies”. In: *CoRR* abs/1504.00702 (2015). arXiv: 1504.00702. URL: http://arxiv.org/abs/1504.00702.

[129] Zhizhong Li and Derek Hoiem. “Learning without Forgetting”. In: *CoRR* abs/1606.09282 (2016). arXiv: 1606.09282. URL: http://arxiv.org/abs/1606.09282.

[130] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[131] Shikun Liu, Edward Johns, and Andrew J Davison. “End-to-End Multi-task Learning with Attention”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1871–1880.

[132] Ying Liu et al. “Deep reinforcement learning for dynamic treatment regimes on medical registry data”. In: *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE. 2017, pp. 380–385.

[133] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[134] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[135] Aravindh Mahendran et al. “ResearchDoom and CocoDoom: Learning Computer Vision with Games”. In: *CoRR* abs/1610.02431 (2016). arXiv: 1610.02431. URL: http://arxiv.org/abs/1610.02431.

[136] Davide Maltoni and Vincenzo Lomonaco. “Continuous Learning in Single-Incremental-Task Scenarios”. In: *CoRR* abs/1806.08568 (2018). arXiv: 1806.08568. URL: http://arxiv.org/abs/1806.08568.

[137] Manolis Savva* et al. “Habitat: A Platform for Embodied AI Research”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[138] Lucas Manuelli et al. *Keypoints into the Future: Self-Supervised Correspondence in Model-Based Reinforcement Learning*. 2020. arXiv: 2009.05085 [cs.RO].

[139] Lucas Manuelli et al. “kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation”. In: *CoRR* abs/1903.06684 (2019). arXiv: 1903.06684. URL: http://arxiv.org/abs/1903.06684.

[140] Loic Matthey et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *ICLR 2017*. 2017.

[141] A. Mian, M. Bennamoun, and R. Owens. “On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes”. In: *International Journal of Computer Vision* 89.2 (Sept. 2010), pp. 348–361. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0296-z. URL: https://doi.org/10.1007/s11263-009-0296-z.

[142] O. Miksik. “Rapid vanishing point estimation for general road detection”. In: *2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 4844–4849. DOI: 10.1109/ICRA.2012.6225206.

[143] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[144] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[145] Ishan Misra et al. “Cross-stitch networks for multi-task learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3994–4003.

[146] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. URL: http://dx.doi.org/10.1038/nature14236.

[147] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[148] Henrik Mouritsen. “Long-distance navigation and magnetoreception in migratory animals”. In: *Nature* 558.7708 (2018), pp. 50–59. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0176-1. URL: https://doi.org/10.1038/s41586-018-0176-1.

[149] Arsalan Mousavian et al. “Visual Representations for Semantic Target Driven Navigation”. In: *CoRR* abs/1805.06066 (2018). arXiv: 1805.06066. URL: http://arxiv.org/abs/1805.06066.

[150] Matthias Müller et al. “Driving Policy Transfer via Modularity and Abstraction”. In: *CoRR* abs/1804.09364 (2018). arXiv: 1804.09364. URL: http://arxiv.org/abs/1804.09364.

[151] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

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

[153] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[154] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[155] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[156] Yaniv Ovadia et al. *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*. 2019. arXiv: 1906.02530 [stat.ML].

[157] Maks Ovsjanikov et al. "Functional Maps: A Flexible Representation of Maps Between Shapes". In: *ACM Trans. Graph.* 31.4 (July 2012), 30:1–30:11. ISSN: 0730-0301. DOI: 10.1145/2185520.2185526. URL: http://doi.acm.org/10.1145/2185520.2185526.

[158] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[159] Deepak Pathak et al. "Curiosity-driven Exploration by Self-supervised Prediction". In: *CoRR* abs/1705.05363 (2017). arXiv: 1705.05363. URL: http://arxiv.org/abs/1705.05363.

[160] Jonathan W. Peirce. "Understanding mid-level representations in visual processing". In: *Journal of Vision* 15.7 (June 2015), pp. 5–5. ISSN: 1534-7362. DOI: 10.1167/15.7.5. eprint: https://jov.arvojournals.org/arvo/content_public/journal/jov/934215/i1534-7362-15-7-5.pdf. URL: https://dx.doi.org/10.1167/15.7.5.

[161] Lerrel Pinto et al. "Asymmetric actor critic for image-based robot learning". In: *arXiv preprint arXiv:1710.06542* (2017).

[162] Lerrel Pinto et al. "The curious robot: Learning visual representations via physical interactions". In: *European Conference on Computer Vision*. Springer. 2016, pp. 3–18.

[163] Matthias Plappert et al. "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research". In: *CoRR* abs/1802.09464 (2018). arXiv: 1802.09464. URL: http://arxiv.org/abs/1802.09464.

[164] F Proctor, Marek Franaszek, and J Michaloski. "Tolerances and uncertainty in robotic systems". In: *ASME 2017 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers Digital Collection. 2017.

[165] Senthil Purushwalkam Shiva Prakash and Abhinav Gupta. "Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases". In: *Advances in Neural Information Processing Systems* 33 (2020).

[166] Xiaojuan Qi et al. "Geonet: Geometric neural network for joint depth and surface normal estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 283–291.

[167] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: http://proceedings.mlr.press/v139/radford21a.html.

[168] Ilija Radosavovic et al. "Real-World Robot Learning with Masked Visual Pre-training". In: *CoRL* (2022).

[169] Antonin Raffin et al. "S-RL Toolbox: Environments, Datasets and Evaluation Metrics for State Representation Learning". In: *arXiv preprint arXiv:1809.09369* (2018).

[170] Antonin Raffin et al. "S-RL Toolbox: Environments, Datasets and Evaluation Metrics for State Representation Learning". In: *arXiv preprint arXiv:1809.09369* (2018).

[171] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. "Vision Transformers for Dense Prediction". In: *ArXiv preprint* (2021).

[172] René Ranftl et al. "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer". In: *arXiv preprint arXiv:1907.01341* (2019).

[173] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. *Learning multiple visual domains with residual adapters*. 2017. arXiv: 1705.08045 [cs.CV].

[174] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond". In: *arXiv preprint arXiv:1904.09237* (2019).

[175] Stephan R. Richter et al. "Playing for Data: Ground Truth from Computer Games". In: *European Conference on Computer Vision (ECCV)*. Ed. by Bastian Leibe et al. Vol. 9906. LNCS. Springer International Publishing, 2016, pp. 102–118.

[176] Mike Roberts and Nathan Paczan. *Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding*. arXiv 2020.

[177] E. Rohmer, S. P. N. Singh, and M. Freese. "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. www.coppeliarobotics.com. 2013.

[178] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[179] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[180] Fereshteh Sadeghi and Sergey Levine. "Cad2rl: Real single-image flight without a single real image". In: *arXiv preprint arXiv:1611.04201* (2016).

[181] Manolis Savva* et al. "Habitat: A Platform for Embodied AI Research". In: *arXiv preprint arXiv:1904.01201* (2019).

[182] Alexander Sax et al. *Learning to Navigate Using Mid-Level Visual Priors*. 2019. arXiv: 1912.11121 [cs.CV].

[183] Steffen Schneider et al. "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, 2019, pp. 3465–3469. DOI: 10.21437/Interspeech.2019-1873. URL: https://doi.org/10.21437/Interspeech.2019-1873.

[184] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[185] John Schulman et al. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

[186] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[187] H. Scudder. "Probability of error of some adaptive pattern-recognition machines". In: *IEEE Transactions on Information Theory* 11.3 (1965), pp. 363–371. DOI: 10.1109/TIT.1965.1053799.

[188] Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach". In: *arXiv preprint arXiv:1708.00489* (2017).

[189] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[190] Evan Shelhamer et al. "Loss is its own Reward: Self-Supervision for Reinforcement Learning". In: *CoRR* abs/1612.07307 (2016). arXiv: 1612.07307. URL: http://arxiv.org/abs/1612.07307.

[191] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8 (Aug. 2000), pp. 888–905. ISSN: 0162-8828. DOI: 10.1109/34.868688. URL: http://dx.doi.org/10.1109/34.868688.

[192] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 354023957X.

[193] Nathan Silberman et al. "Indoor Segmentation and Support Inference from RGBD Images". In: *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760. 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.

[194] Nathan Silberman et al. "Indoor segmentation and support inference from rgbd images". In: *European conference on computer vision*. Springer. 2012, pp. 746–760.

[195] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[196] Shuran Song et al. "Semantic Scene Completion from a Single Depth Image". In: *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[197] Pratul P Srinivasan et al. "Aperture supervision for monocular depth estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6393–6401.

[198] Trevor Standley et al. "Which Tasks Should Be Learned Together in Multi-task Learning?" In: *arXiv e-prints*, arXiv:1905.07553 (May 2019), arXiv:1905.07553. arXiv: 1905.07553 [cs.CV].

[199] Bastian Steder et al. "NARF: 3D Range Image Features for Object Recognition". In: ().

[200] Bastian Steder et al. "NARF: 3D range image features for object recognition". In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 44. 2010.

[201] James Stewart. *Essential calculus: Early transcendentals*. Cengage Learning, 2012.

[202] Julian Straub et al. "The Replica Dataset: A Digital Replica of Indoor Spaces". In: *arXiv preprint arXiv:1906.05797* (2019).

[203] Julian Straub et al. "The Replica dataset: A digital replica of indoor spaces". In: *arXiv preprint arXiv:1906.05797* (2019).

[204] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 843–852.

[205] Ke Sun et al. "Deep high-resolution representation learning for human pose estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5693–5703.

[206] Yu Sun et al. "Test-time training for out-of-distribution generalization". In: *arXiv preprint arXiv:1909.13231* (2019).

[207] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262193981.

[208] N Tinbergen. *The study of instinct.* New York, NY, US, 1951.

[209] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[210] Antonio Torralba and Alexei A. Efros. "Unbiased look at dataset bias". In: *CVPR 2011*. 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[211] Simon Vandenhende, Bert De Brabandere, and Luc Van Gool. "Branched Multi-Task Networks: Deciding What Layers To Share". In: *CoRR* abs/1904.02920 (2019). arXiv: 1904.02920. URL: http://arxiv.org/abs/1904.02920.

[212] Simon Vandenhende et al. "Branched multi-task networks: deciding what layers to share". In: *arXiv preprint arXiv:1904.02920* (2019).

[213] Simon Vandenhende et al. "Multi-task learning for dense prediction tasks: A survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[214] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[215] Pascal Vincent et al. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 1096–1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294. URL: http://doi.acm.org/10.1145/1390156.1390294.

[216] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *CoRR* abs/1606.04080 (2016). arXiv: 1606.04080. URL: http://arxiv.org/abs/1606.04080.

[217] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. "Learning Correspondence from the Cycle-Consistency of Time". In: *CoRR* abs/1903.07593 (2019). arXiv: 1903.07593. URL: http://arxiv.org/abs/1903.07593.

[218] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[219] Max Wertheimer. "Laws of Organization in Perceptual Forms". In: *Psycologische Forschung* 4 (1923), pp. 301–350.

[220] Norbert Wiener. *Cybernetics; or control and communication in the animal and the machine.* Oxford, England: John Wiley, 1948, p. 194.

[221] Yilin Wu et al. "Learning to manipulate deformable objects without demonstrations". In: *arXiv preprint arXiv:1910.13439* (2019).

[222] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.

[223] Fei Xia et al. "Gibson Env: Real-World Perception for Embodied Agents". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[224] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A benchmark for 3D object detection in the wild". In: *IEEE Winter Conference on Applications of Computer Vision*. 2014, pp. 75–82. DOI: 10.1109/WACV.2014.6836101.

[225] Dan Xu et al. "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 675–684.

[226] Wilson Yan et al. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *arXiv preprint arXiv:2003.05436* (2020).

[227] Wei Yang et al. "Visual Semantic Navigation using Scene Priors". In: *CoRR* abs/1810.06543 (2018). arXiv: 1810.06543. URL: http://arxiv.org/abs/1810.06543.

[228] Yao Yao et al. "BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks". In: *Computer Vision and Pattern Recognition (CVPR)* (2020).

[229] Lin Yen-Chen et al. "Learning to See before Learning to Act: Visual Pre-training for Manipulation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020. URL: https://yenchenlin.me/vision2action/.

[230] Wei Yin et al. "Enforcing geometric constraints of virtual normal for depth prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 5684–5693.

[231] Zhichao Yin and Jianping Shi. "Geonet: Unsupervised learning of dense depth, optical flow and camera pose". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1983–1992.

[232] Alex Yu et al. "pixelNeRF: Neural Radiance Fields from One or Few Images". In: *CVPR*. 2021.

[233] Zehao Yu et al. "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[234] Xiaomin Yue et al. "Curvature-processing network in macaque visual cortex". In: *Proceedings of the National Academy of Sciences* 111.33 (2014), E3467–E3475. DOI: 10.1073/pnas.1412616111. eprint: http://www.pnas.org/content/111/33/E3467.full.pdf. URL: http://www.pnas.org/content/111/33/E3467.abstract.

[235] A. Zaharescu et al. "Surface feature detection and description with applications to mesh matching". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 373–380. DOI: 10.1109/CVPR.2009.5206748.

[236] Amir Zamir et al. “Robust Learning Through Cross-Task Consistency”. In: *arXiv* (2020).

[237] Amir R Zamir et al. “Taskonomy: Disentangling task transfer learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3712–3722.

[238] Amir R. Zamir et al. “Taskonomy: Disentangling Task Transfer Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2018.

[239] Jeffrey O Zhang et al. *Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks*. 2019. arXiv: 1912.13503 [cs.LG].

[240] Lilian Zhang et al. “Vanishing Point Estimation and Line Classification in a Manhattan World with a Unifying Camera Model”. In: *International Journal of Computer Vision* 117.2 (Apr. 2016), pp. 111–130. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0854-5. URL: https://doi.org/10.1007/s11263-015-0854-5.

[241] Yinda Zhang et al. “Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks”. In: *CoRR* abs/1612.07429 (2016). arXiv: 1612.07429. URL: http://arxiv.org/abs/1612.07429.

[242] Yinda Zhang et al. “Physically-based rendering for indoor scene understanding using convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5287–5295.

[243] Yu Zhang and Qiang Yang. “A Survey on Multi-Task Learning”. In: *CoRR* abs/1707.08114 (2017). arXiv: 1707.08114. URL: http://arxiv.org/abs/1707.08114.

[244] Zaiwei Zhang et al. “Path-Invariant Map Networks”. In: *CoRR* abs/1812.11647 (2018). arXiv: 1812.11647. URL: http://arxiv.org/abs/1812.11647.

[245] Y. Zhong. “Intrinsic shape signatures: A shape descriptor for 3D object recognition”. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. Sept. 2009, pp. 689–696. DOI: 10.1109/ICCVW.2009.5457637.

[246] Bolei Zhou et al. “Learning Deep Features for Scene Recognition using Places Database”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 487–495. URL: http://papers.nips.cc/paper/5349-learning-deep-features-for-scene-recognition-using-places-database.pdf.

[247] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. “Does computer vision matter for action?” In: *arXiv e-prints*, arXiv:1905.12887 (May 2019), arXiv:1905.12887. arXiv: 1905.12887 [cs.CV].

[248] Tinghui Zhou et al. “Learning Dense Correspondence via 3D-guided Cycle Consistency”. In: *CoRR* abs/1604.05383 (2016). arXiv: 1604.05383. URL: http://arxiv.org/abs/1604.05383.

[249] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *CoRR* abs/1703.10593 (2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

[250] Z. Zhu et al. “Ten-fold Improvement in Visual Odometry Using Landmark Matching”. In: *2007 IEEE 11th International Conference on Computer Vision*. Oct. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409062.

[251] Zihan Zhu et al. *NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM*. 2023. arXiv: 2302.03594 [cs.CV].

[252] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. “Df-net: Unsupervised joint learning of depth and flow using cross-task consistency”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 36–53.

Appendix A

Chapter 2 Supplementary Material

A.1 Detailed Methodology

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 [146] 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 [146].

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) [100], 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 [152, 190, 105], curiosity [159], DARLA [88], and ImageNet pretraining [119].

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.

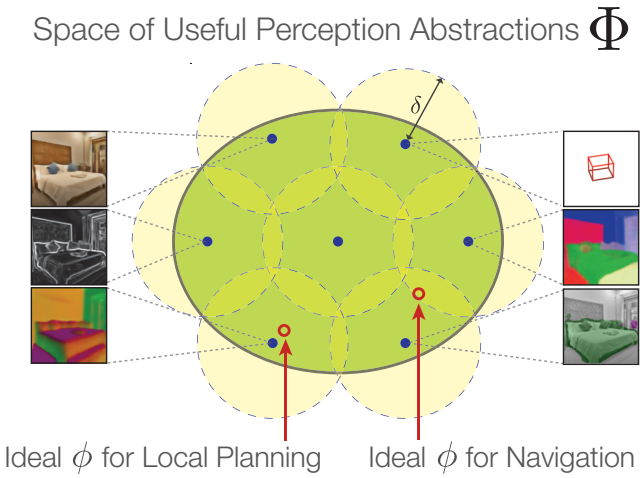


Figure A.1: **Geometry of the feature set.** We select a covering set of features that minimizes the worst-case distance between the subset and the *ideal* task feature. By *Hypothesis III*, no single feature will suffice and a set is required. okay.

The question now becomes how to find the best compact set, shown in Figure A.1. 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 A.1.

The task taxonomy method [238] 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 [238], we define operators returning target and sources of an edge:

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

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

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 [238])
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:

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

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

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

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

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:

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

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 [80].

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.

Downstream Active Tasks

This section contains detailed descriptions of our 3 locomotion-based active tasks. Visual depictions are shown in Fig. A.2, and (environment-specific numbers are provided in supplementary material).

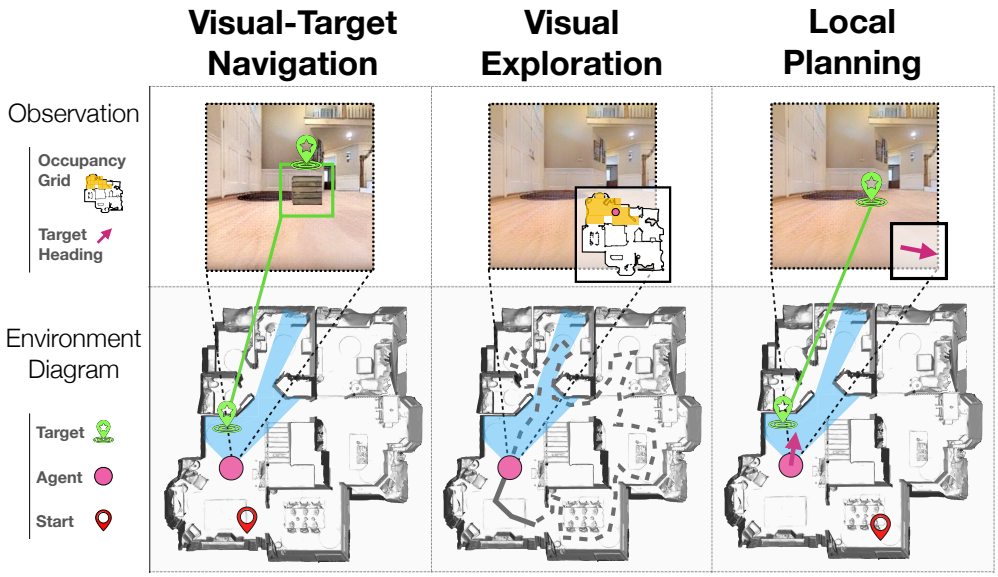


Figure A.2: **Visual descriptions of the selected active tasks.** For each task, an example state of the environment is shown in the bottom row. The agent is shown as a pink dot, and its field of view is shown in light blue. Starting locaitons are shown with the red marker and a target, if it exists 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.

Dictionary of Mid-Level Vision Objectives

Figure A.3 contains a complete list of our studied vision objectives. A detailed description of each vision task can be found the Taskonomy [238] supplementary material (Section 14). We visualize

some tasks in Figure A.3 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

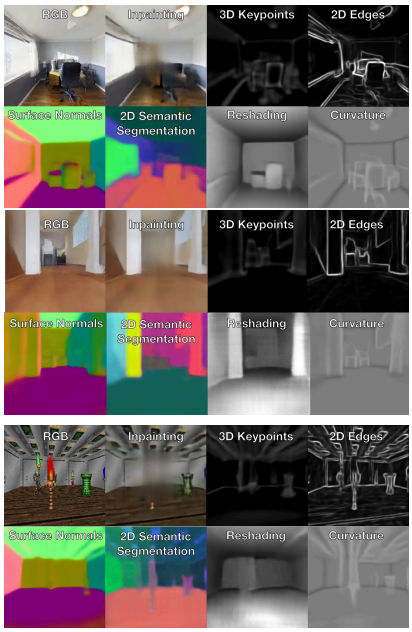


Figure A.3: **Feature Bank with Sample Frames.** [Left] The mid-level features we used for all experiments. [Right] Frames and their respective mid-level features for Habitat, Gibson, and Doom for the top, middle, bottom frames respectively.

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*:

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

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 2.5 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 [7]. 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:
>
> $$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)}.$$

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:

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

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 [185]) 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:

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

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:

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

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

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 [186]. 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.

A.2 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.

Performance

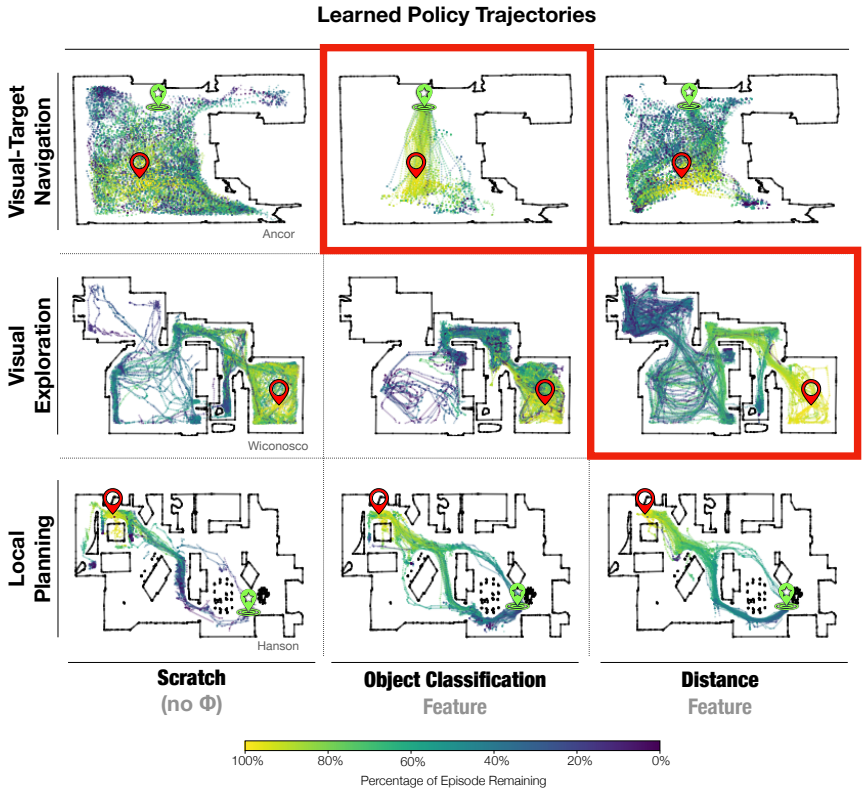


Figure A.4: **Visualizations of the agents' trajectory.** While all approaches have reasonable trajectories for local planning, only certain features have desired trajectories for visual-target navigation and visual exploration.

Are the differences in the reward values meaningful? Do they leed to different behaviors? In Figure A.4, 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 A.5 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 [25]. 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 A.2).

Generalization

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

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 A.6 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.

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 A.7.

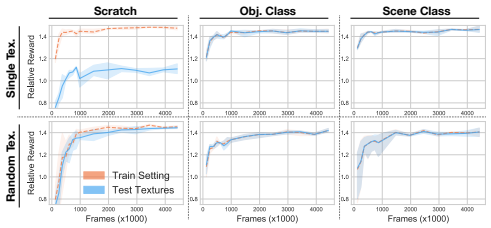


Figure A.7: **Generalization in Doom Textures** In ViZDoom, feature-based agents (right two columns) generalize to new textures even when not exposed to texture variation in training (top row), while agents trained from scratch suffer a significant drop in performance (left, top).

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.

Feature Rankings in different tasks

In Figure A.8, 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

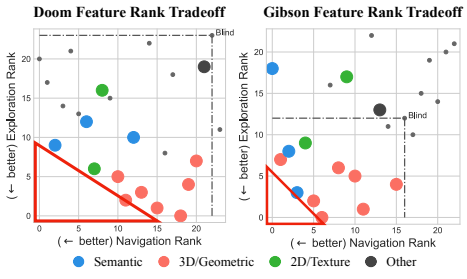


Figure A.8: **Feature Ranks on Exploration and Navigation** Scatterplots of the rank of fea-

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.

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 A.9: **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 A.9.

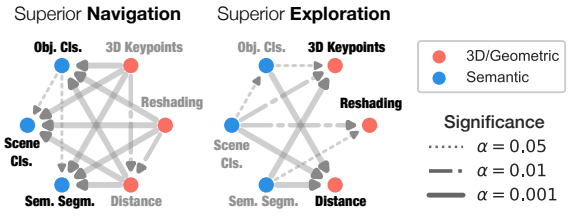


Figure A.10: **Rank Reversal Significance Graphs** Results from pairwise tests evaluating 3 semantic and 3 geometric features on each task. For each task graph, arrows point towards the better-performing feature. Lack of an arrow indicates the performance difference was not statistically significant. Heavier arrowheads denote more significant results (lower α-level). The essentially complete bipartite structure in the graphs shows that

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 A.10, 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.

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 A.11 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.

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 A.1: **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.

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 A.1 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 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.

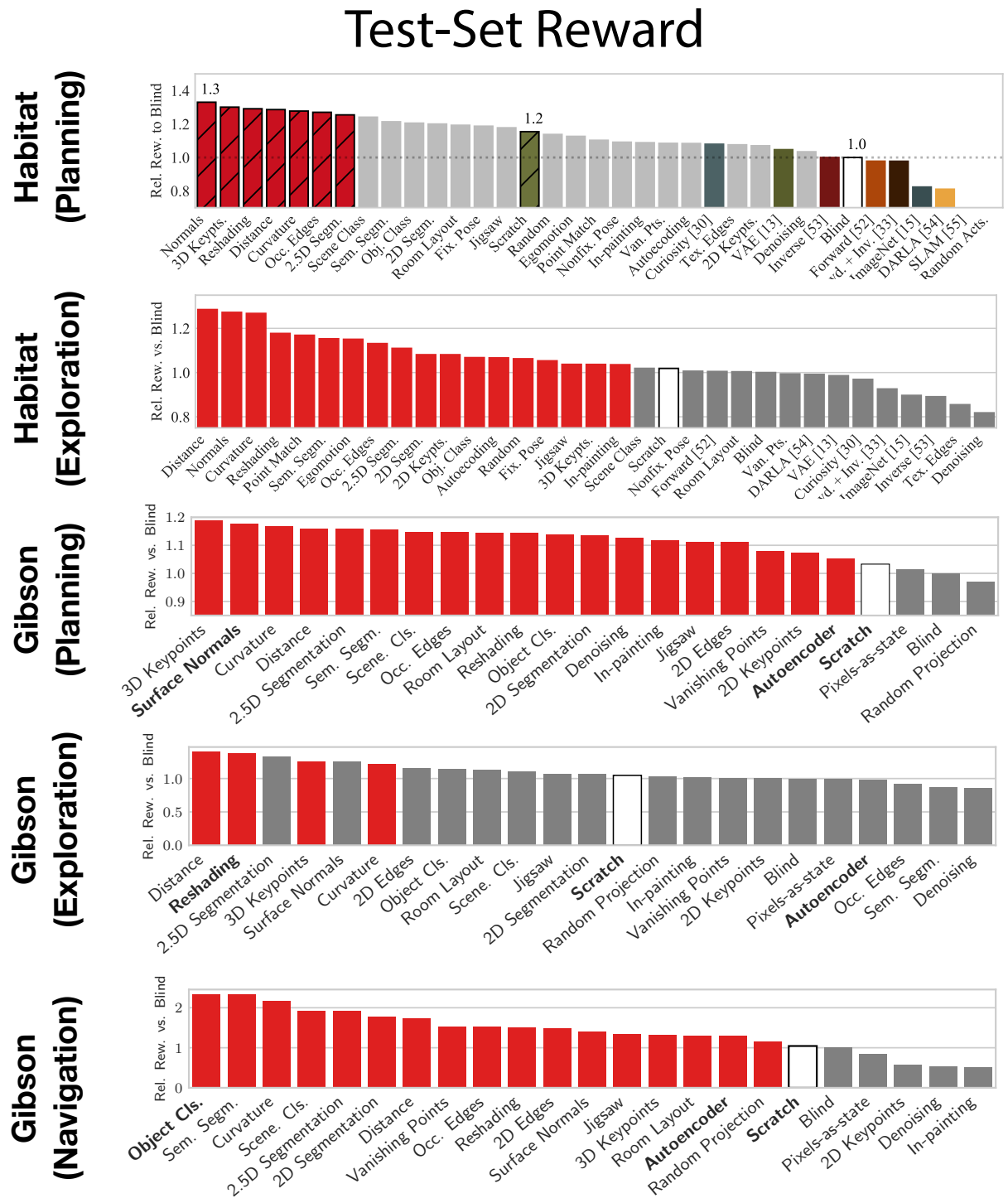


Figure A.5: **Performance.** The reward relative to blind for all methods. Agents significantly better than *scratch* are shown in red.

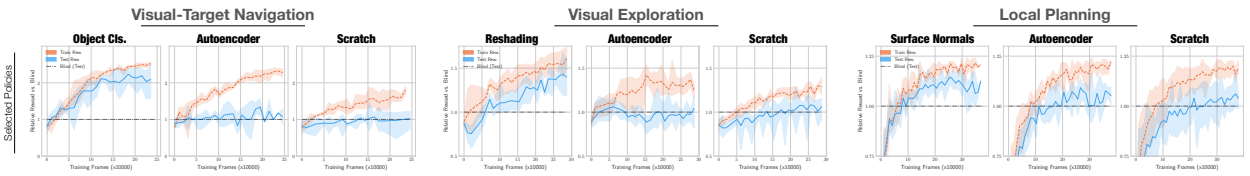


Figure A.6: **Mid-level feature generalization.** The plots above show training and test performance of *scratch* vs. some selected features throughout training. For all tasks there is a significant gap between train/test performance for *scratch*, and a much smaller one for the best feature. This underscores the importance of separating the train and test environment in RL.

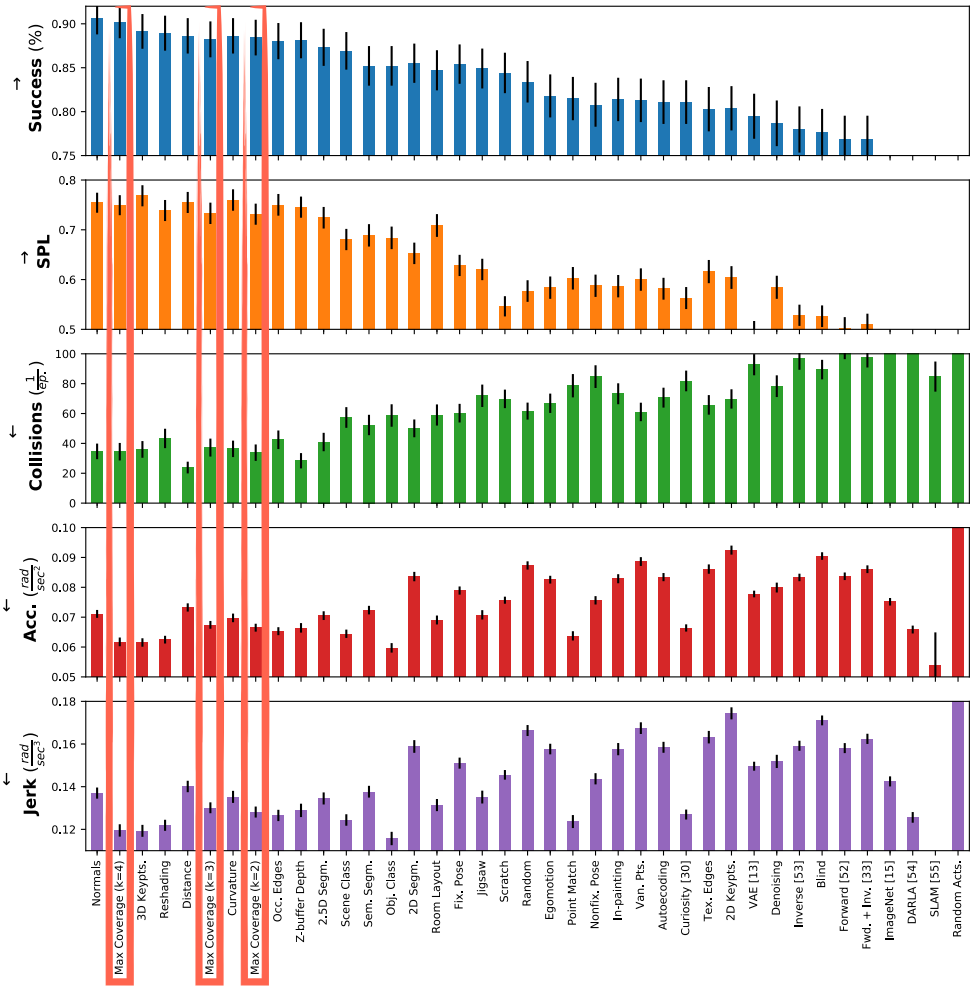


Figure A.11: **Max-coverage feature sets exhibit strong performance and desirable behavior on Local Planning in Habitat.** The shows performance on the Habitat/Local Planning test set along a variety of dimensions. Features are ordered according to *test-set reward*. Max-coverage policies exhibit a strong combination of desirable properties, suggesting that they confer the benefits of mid-level vision.

Appendix B

Chapter 3 Supplementary Material

Supplementary Material

The supplementary material provides additional materials that support the main paper. Specifically we include:

I. An overview video of the project (Section B.1)
II. Videos of sim-to-real test episodes from physical onboard cameras (Section B.2)
III. Code (Section B.3)
IV. Additional experiments with shaped rewards (Section B.4)
V. Complete sim-to-real episode-level results (Section B.5)
VI. Complete descriptions of manipulation tasks (Section B.6)
VII. Complete descriptions of navigation tasks (Section B.7)
VIII. Complete description of train/test splits (Section B.8)
IX. Full list of vision objectives and samples of networks evaluated in our environments (Section B.9)
X. Complete descriptions of sim-to-real setup (Section B.10)
XI. Policy training setup (architectures, hyperparams, etc.) (Section B.11)
XII. Complete description of baselines (Section B.12)
XIII. Train/Test curves for experiments in the main paper (Section B.13)

B.1 Overview Video Clip

The supplementary material includes an overview video clip which provides *various results from the proposed study* including a description of the main hypotheses and comparisons to controls.The clip also shows some sample episode of the sim-to-real generalization experiments, which we find insightful. **We strongly recommend watching the video clip**.

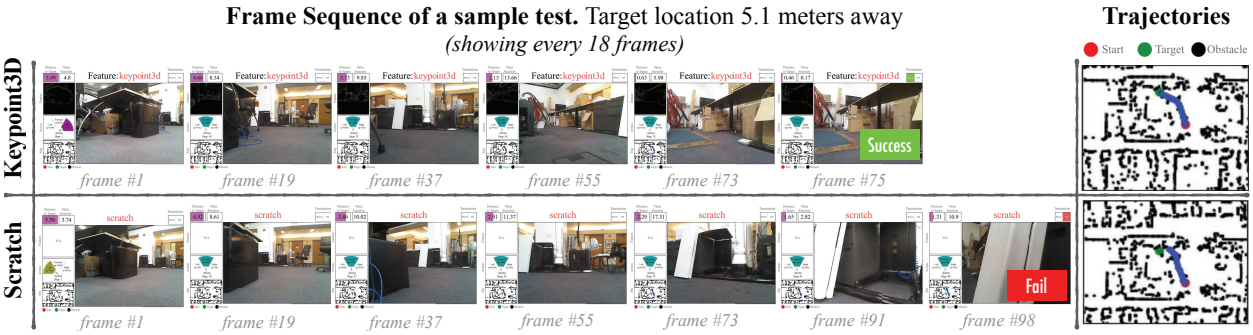


Figure B.1: **Frame sequence of a sample test episode.** Selected frames from keypoint3D (**top**) and scratch (**bottom**) experiments. **Mid-level versus no features:** Even when the target is far away and occluded from the agent, mid-level features are able to successfully complete the task - here is an example of using 3D keypoints. The absence of such features leads most of the cases to failed attempts. **We include more than 100 execution videos in the supplementary main video and in the folder `sim_to_real/test_episode_videos`.**

B.2 Videos of sim-to-real test episodes from physical onboard cameras

We ran 594 evaluation episodes in the sin-to-real experiments comprising 13 hours of runtime. In the folder `sim_to_real/test_episode_videos`, we've included videos from onboard RGB cameras during a representative sample of the test episodes. These videos include a dashboard showing useful metrics at each point in time such as distance to goal, and episode success. Frames sequences from two videos videos are shown in Figure B.1.

B.3 Code

We provide all of our code through a Github repository available https://github.com/alexsax/robust-policies-via-midlevel-vision

B.4 Experiments with Shaped Rewards

Using sparse rewards and HER, we were unable to get the agent trained from scratch to learn useful behaviors in our environments, even after running multiple hyperparameter searches. HER [8] required using shaped rewards to train from pixels, and we were also able to train an agent only by using a dense, shaped reward. Agents trained using mid-level vision outperformed agents trained from scratch even when the pixel-based agents were trained using a dense reward, but dense reward also improved performance for mid-level based agents, shown in Figure B.8.

B.5 Complete sim-to-real episode-level results

These are provided in the file `sim_to_real/test_episodes.csv`.

B.6 Descriptions of Manipulation Tasks

We provide full descriptions of the task setups within RLBench.

Reward: Sparse reward is given by 0 when the cube is within an $\epsilon = 4$cm distance of the target, and -1 otherwise. Dense/shaped reward is given by the negative euclidean distance between the target and the end effector / object, with a positive reward bonus of 0.1 when within the ϵ distance. In total, we tried rewards of the form $-|g - s_{object}|_2^p, -|g - s_{object}|_1^p, p \in \{1, 2\}$ and a brief search over reward bonuses. [8] finds that more advanced shaped rewards considering future states does not help in learning tasks.

Gripper: For Reach and Push, the gripper is in a fixed position. For Pick and Place, it is controlled by the agent's actions. A positive value is considered a close action if not already closed, and vice versa a negative value an open action.

Extra Task Details:

Reach: The target sphere location is randomized in a cube. The agent begins in a fixed position within the cube.

Push: The cube starts in a fixed location on the table. The target sphere location is randomized in a square around the cube. The agent begins in a fixed position above the cube.

Pick and Place: The cube location is randomized in a square on the table. The target sphere location is randomized in a square at a fixed z position above the table. To aid exploration as in [8], the agent begins half of the episodes gripping the cube. We were also able to train mid-level policies by starting half of the episodes with the target sphere on the table, but present the results using the aforementioned setup.

B.7 Description of Navigation Tasks

The task under consideration is PointGoal [7] (referred to as Local Planning in [182]). In the PointGoal task, the agent must navigate to a specified (fixed) target location coordinate. The target location is specified as a vector from the agent to the target. The agent receives positive reward for Euclidean distance to goal at each timestep, and receives negative reward for collisions and time spent. During training, the episode terminates successfully when the agent comes within $0.5m$ of the goal location and fails if the agent doesn't reach the goal in 400 actions. In real-world testing, the episode additionally terminates if the agent collides with a wall or obstacle.

Action space Discretized to $A = \{$*forward 0.25m*, *pivot right* $10°$, *pivot left* $10°\}$. To abstract away difficulties and imperfections in simulating physical dynamics, Gibson assumes no lag dynamics in either the controller or camera; thus, the sense-action loop for the agent consists of 1)

policy receives RGB image from Gibson; 2) policy produces an action; 3) embodied agent teleports to resulting pose; 4) policy receives a new RGB image. We refer the reader to [223] for a detailed discussion of collision handling.

Observation space of the agent consists of Taskonomy [238] encoder representations (with shape $16 \times 16 \times 8$) which output the mid-level features as well as the vector to target location. The vector to target location is encoded as $\cos$ and $\sin$ of the relative heading of the agent to the target, and the magnitude of the distance. To match the shape of the Taskonomy encoders, the target vector is tiled to have shape $16 \times 16 \times 3$.

B.8 Train and Test Splits

Manipulation

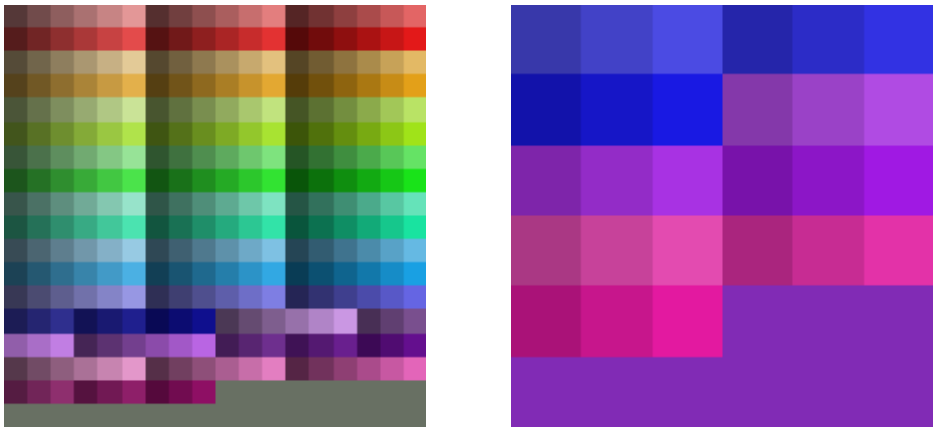


Figure B.2: **Train/test split of textures for domain randomization. Left:** Colors used for textures in the *domain randomization* treatment group. Replaced table, floor, and background textures. **Right:** Textures used in the testing environment for all policies.

We create our train/test set through applying color textures to the floor, walls, and table. To generate colors, we generate HSV tuples with equally spaced values across each dimension, throwing away values for visibility. We take 90% of these values along each dimension for train, and hold out the remaining 10% for test. This formed the train and test for policies trained with domain randomization, whereas for policies trained with the regular environment, the test set was given by the domain randomized train set.

Sim-to-real

The sim-to-real policies are trained in simulation in a single building and tested in two unseen physical buildings. For complete information, please see Section B.10.

B.9 Mid-Level Vision Objectives

We trained our agents using the following mid-level objectives:

All mid-level vision objectives:

Mid-Level Vision Objectives ($\widetilde{\Phi}$)
Autoencoding
Denoising
Depth Estimation, Z buffer
Edge Detection, 2D
Edge Detection, 3D
Image Segmentation
Surface Normal Estimation

Except for *Image Segmentation* we used the tasks as defined in [238]. Image segmentation is similar to instance segmentation, but in our case each instance of an item `floor, table, arm, gripper, object` are sorted into their own label. The `object` class includes target spheres as well as cubes.

For denoising, we use inputs with added Gaussian noise independent per pixel sampled from $\mathcal{N}(0, 0.05)$.

Dataset Generation: We generate a dataset to train the mid-level vision objectives by taking random actions in various environment setups. The environments we train on are Reach and Push as above, as well as Pick + Place with agent starting above and near the block. For each of these setups, we generate with and without domain randomization as well as with 2 camera angles. We take uniformly random actions for 25 resets and 150 steps per reset. For the procedural objects, we train another dataset which is comprised of 45 objects each with 50 resets and 2 camera angels. We adapted the code from [170].

Training / Fine Tuning: For final performance figures, we initialize from Taskonomy [238], and for the checkpointing figure we initialize from scratch. We use the Adam optimizer with learning rate 3e-4, with batch size 32, and number of epochs 150 and 500 respectively, taking the model with the best validation loss. We use L1 Loss for all takss except for Image Segmentation, where we use Cross Entropy Loss. For the procedural objects, we fine tuned the mid-level normal policy from the final performance figures on just the procedural object dataset, using learning rate 3e-4 with batch size 32 and 500 epochs.

We show (non-cherry-picked) results of the final networks evaluated in our environments in Figure B.3.

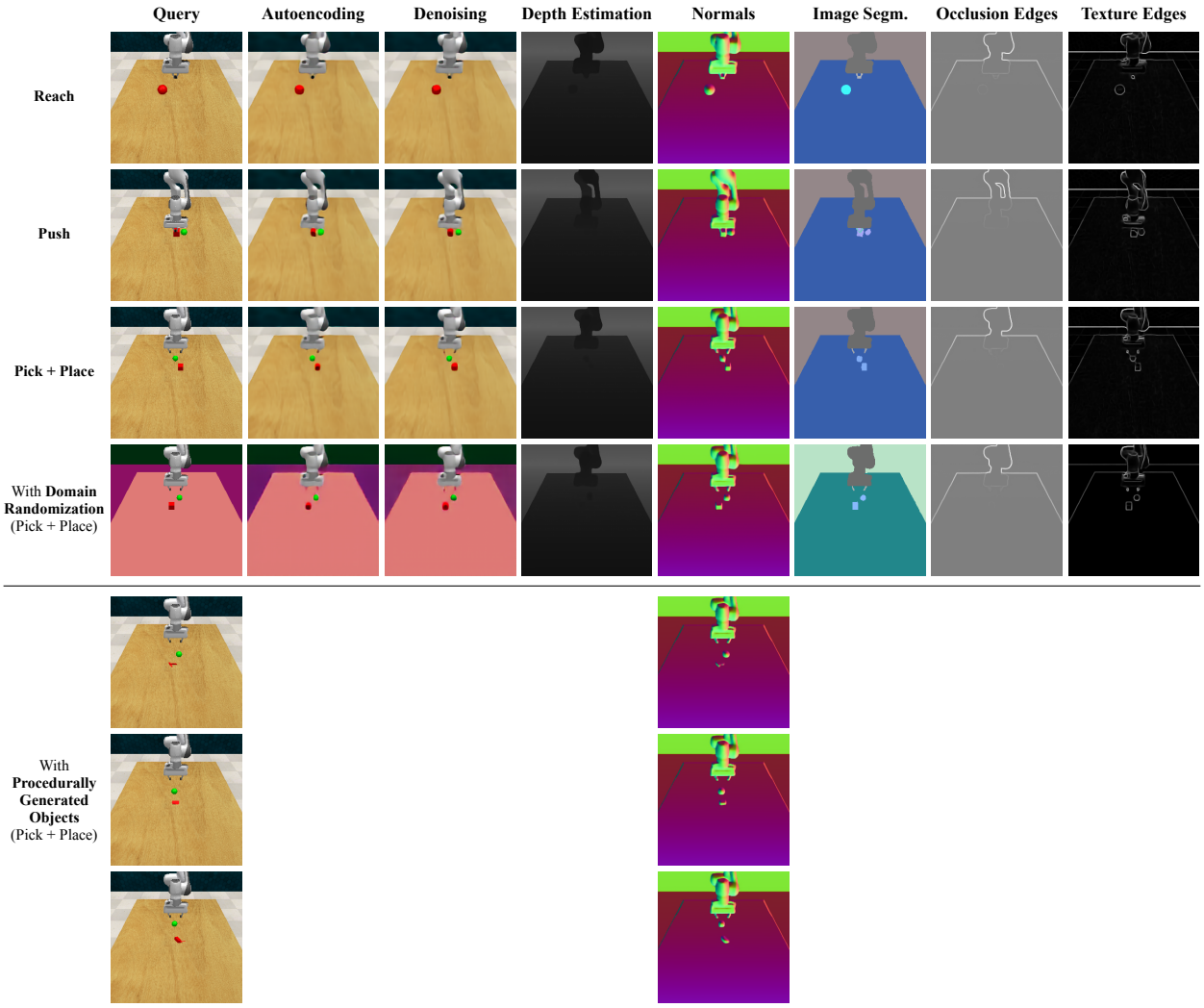


Figure B.3: **Mid-level neural networks evaluated in our environments.**

B.10 Sim-to-Real Setup

For the sim-to-real setup we test the generalization of the pretrained "Local Planning" (PointGoal) policies from [182]. The policies are trained in the Gibson simulation environment [223], which employs virtualized spaces to render images to the agent that capture much of the inherent visual and semantic complexity of the real world. The training environment, called Beechwood, is a large house with $154.9m^2$ of navigable space.

The policies are AtariNet [146] models trained with Proximal Policy Optimization [186] and Generalized Advantage Estimation [185]. A thorough hyperparameter search was conducted for the *scratch* baseline, with the best performing hyperparameters frozen and used for all policies; this method favors the *scratch* baseline, making any gains over *scratch* all the more significant.

Hardware Setup

We used a Turtlebot with Kobuki base. The Turtlebot is a reliable mobile base platform, with easy interaction using the Robot Operating System (ROS) framework. The on-board computer is quite minimal, and doesn't come with sufficient computing resources to run the policies; we therefore network Turtlebot with a server equipped with an NVIDIA RTX 2080 Ti GPU. We additionally augment the platform with a WiFi AC wireless adaptor and router to minimize communication latency between Turtlebot and the server. The round-trip latency between the main computer and Turtlebot is optimized to roughly .2s per action, resulting in near-continuous action execution.

Our camera is a Microsoft Kinect, which has a 43° vertical field of view and outputs color images as $640 \times 480 \times 3$ tensors. We only use the RGB output of the camera and discard depth.

We discretize the Turtlebot's continuous action space into $A = \{$*forward*, *pivot right*, *pivot left*$\}$ to match that of the policies; each action corresponds to a velocity command to the Turtlebot base, resulting in roughly 10cm movement along robot x-axis and heading changes of -10° and +10° for each of the three actions respectively.

For calculation of vector from agent to target, we utilize the Turtlebot's on-board odometry as ground truth; in particular, we don't use external localization such as a GPS, beacon, or SLAM [153] module. We find that the odometry exhibits acceptable performance for our scenarios, with average error of $0.25m$ across all runs.

Our controller communicates with the main computer with a client/server pattern. When the controller receives a motion command, it executes an action for .6 seconds and stops. The next received image is then sent back to the main computer. This pattern is used to ensure images acted on by the policy are the final result of executing the previous action, and so provides close parity with the sense-action-loop training dynamics experienced by the policy in simulation.

Real-World Testing Scenarios

We test our policies on 24 scenarios consisting of start-goal location pairs across two office buildings with notably diverse interior scenes, seen in Figure B.4. Each building comes with pre-existing 3D scans and metric meshes, along with camera positions used for scanning. The scans are only

used for planning the study, and the agent does not receive any prior information from them. Scenarios were generated in each building by 1) selecting start locations uniformly at random from the list of camera positions constrained to public areas; 2) sampling a target distance d from a Gaussian with mean $5m$ and standard deviation $4m$; 3) choosing the camera position that has distance from start location closest to the sampled distance; 4) uniformly sampling the initial orientation from $[-\pi, \pi]$. Following this procedure, we generated four starting locations for Building 1 and three starting locations for Building 2, each with three goal locations for a total of 12 scenarios in Building 1 and 9 scenarios in Building 2. We generated a total of 594 evaluation runs across all scenarios and policies, for a total of 13 hours of continuous execution time. A full list of scenarios and their geometric characteristics can be found in the supplementary material file `sim_to_real/test_episode_results.csv`.

Scenarios vary significantly in length, complexity, and visual characteristics. One metric used to evaluate the difficulty or non-linearity of a physical navigation scenario is *navigation complexity* [223], defined as the ratio of geodesic (i.e. optimal path) distance to Euclidean distance. Average geodesic distance among scenarios is $4.92m$ (variance $4.16m^2$) in Building 1, $6.42m$ (variance $0.86m^2$) in Building 2, and $5.24m$ (variance $3.65m^2$) overall, while average navigation complexity among scenarios is 1.18 (variance 0.036) in Building 1, 1.06 (variance 0.001) in Building 2, and 1.15 (variance 0.03) overall. Visual differences between buildings include floor patterns and coloration, lighting sources & conditions, furniture styles, clutter makeup and distribution, and wall material (e.g. glass or drywall).

B.11 Policy Learning Setup

Architecture: Our architecture follows [182] closely:

Pretrained Feature encoder: For all tasks, we modified a ResNet-50 encoder with no average-pooling and replace the last stride 2 convolution with stride 1. This gives us an output shape of 16 x 16 x 2048. We use a 3 x 3 convolution to transform the output to a shape of 16 x 16 x 8.

Feature readout network: The feature readout network maps the feature encoding to a final representation. It consists of one convolutional (Conv) layer and two fully-connected (FC) layers:

- Conv, 32 channel, 4x4 kernel, stride 4
- FC, ouptut size = 1024
- FC, ouptut size = 512

We train the readout networks on the vision objectives end to end, and at the end of training freeze the encoder output for use as inputs to the downstream policy networks.

Atari-net network: The atari-net network [146] consists of three convolutional layers; we modify it slightly as follows:

- Conv, 32 channel, 4x4 kernel, stride 2

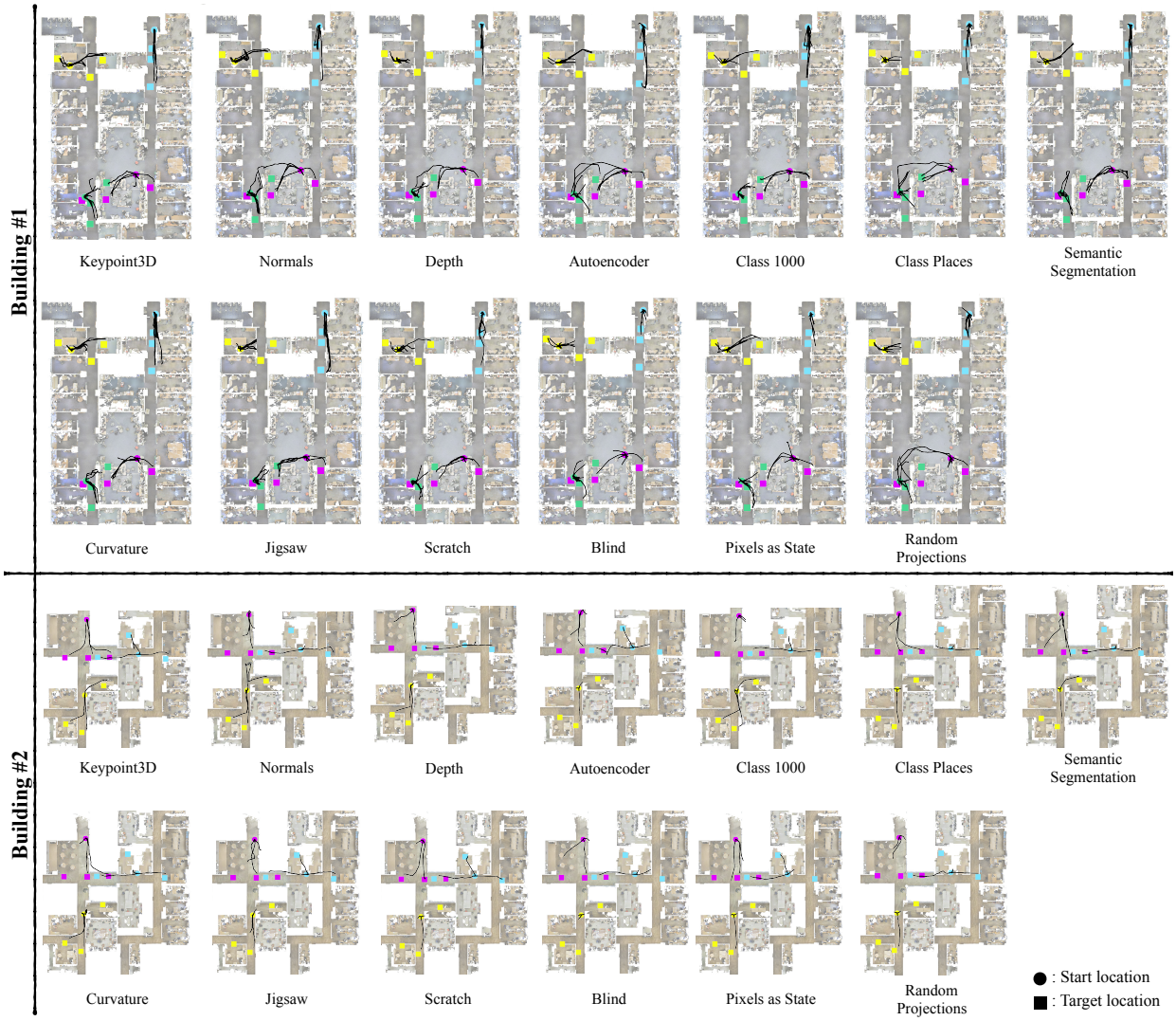


Figure B.4: **Recorded Trajectories** of all experiments for the 2 text buildings. We ran a total of 594 evaluations in both buildings, for a variety of start-target locations and features. The recorded trajectories are visualized per feature and start-target pair.

- Conv, 64 channel, 4x4 kernel, stride 1
- Conv, 64 channel, 3x3 kernel, stride 1
- FC, ouptut size = 1024
- FC, output size = 512

Putting it together: For mid-level policies, we use images of size 256 x 256 x 3 as input for the feature encoder, and freeze the output for use as the representation for downstream policy networks. We use the Atari-net network with slight modifications for learning from pixels on images of size 64 x 64 x 3.

Other methods: Another approach (such as in [247] and others) is to directly use the output of the upstream computer vision tasks instead of the latent representations. In addition to the latents, we ran experiments with the full output as well as the ground truth of the vision tasks, and found no difference in the performance; this was the case in [182] as well. Here, we chose the latents approach due to its uniform shape across tasks and smaller size.

Hyperparameters: We conducted a grid search for the best hyperparameters for learning from scratch for Reach, and then used those same hyperparameters for mid-level policies across all tasks, aside from a few noted differences: first, for Reach, noise for exploration was epsilon-greedy with $\epsilon = 0.3$, while for Push and Pick and Place we used Gaussian noise decaying from $\sigma = 0.3$ to 0.1 over 10^6 timesteps. For learning rate we used 1e-5 for both actor and critic when learning from pixels, and 1e-4 for both actor and critic when learning mid-level policies. We find that mid-level policies are still successful using a learning rate of 1e-5, but we are unable to successfully train policies from scratch using a learning rate of 1e-4. The rest of the hyperparameters are the same: Batch size of 128, discount factor 0.95, target policy noise 0.2, policy delay of 2, replay buffer size of 1e6, rollouts of size 100 per epoch, gradient updates of size 100 per epoch, rollouts of size 50. We use the Adam optimizer [109] and no frame stacking.

HER: To help with exploration for training from sparse rewards, we implement a visual form of HER [8], where instead of concatenating different goals to a state/proprioception observation, we re-render the observation using different goals. For goal resampling, we use the "final" method with k=4.

Training Summary

We provide a full description of the main agents we trained on Reach, Push, and Pick + Place. We focused on training agents with dense rewards as well as with sparse rewards in conjunction with HER. Policies trained using state observations were successful for all settings. Policies trained from scratch were only successful on Reach with dense reward, failing on all tasks with sparse reward. Successful mid-level policies were trained on Reach and Push with dense reward, as well as on all tasks with sparse reward.

B.12 Full Descriptions of Baselines

We use the following baselines, which are the same as [182]. They are summarized in [182] and reproduced here:

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 [146] network.

Blind: 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. This gives us an idea of how much performance can be learned from nonvisual biases, correlations, and structure of the environment.

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.

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.

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 [146].

In addition, we test:

State: In this setting the agent has access to the complete environment state: joint positions, the goal centroid and, if applicable, the object center. Since objects are always the same, this should be sufficient.

B.13 Train and Test Curves

Train/test curves for mid-level policies trained with sparse reward

Train/test curves for policies trained with sparse rewards. We omit the curves for *scratch* as they failed to train in all environments.

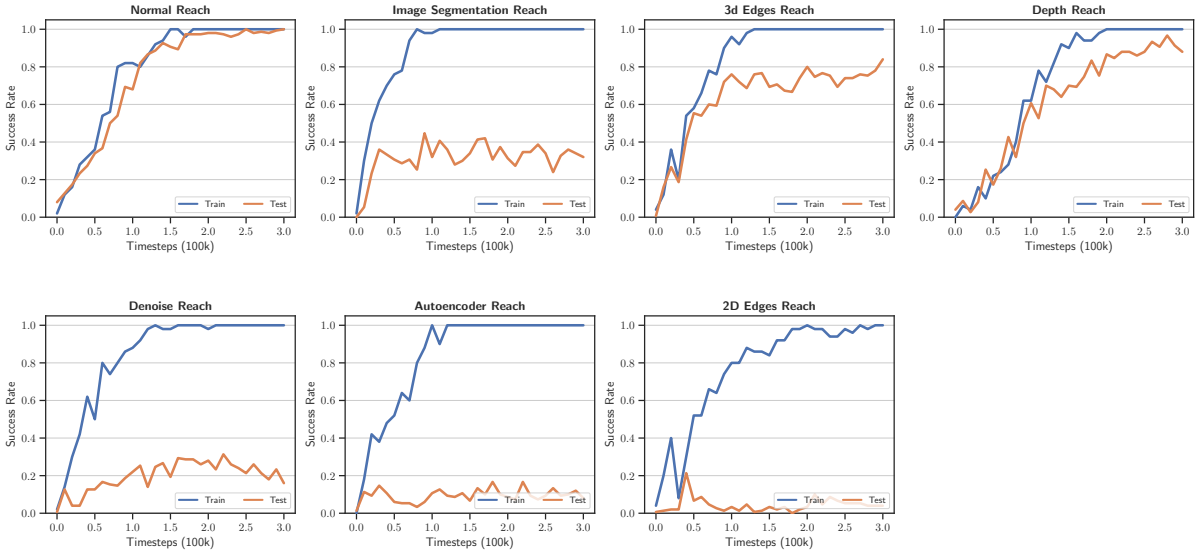


Figure B.5: Reach policies: reward vs. number of training iterations

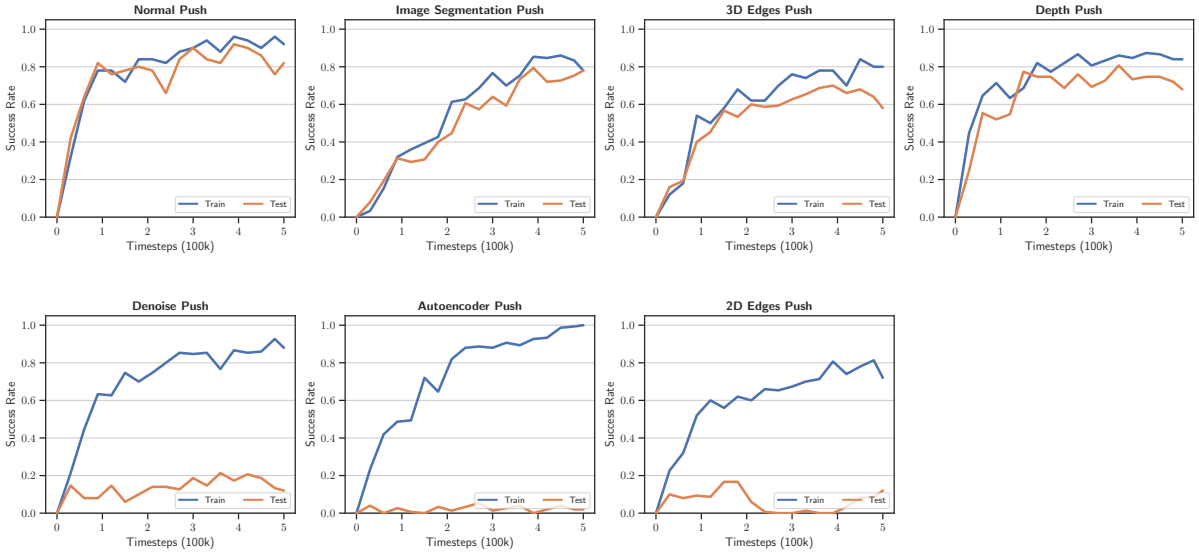


Figure B.6: Push policies: reward vs. number of training iterations

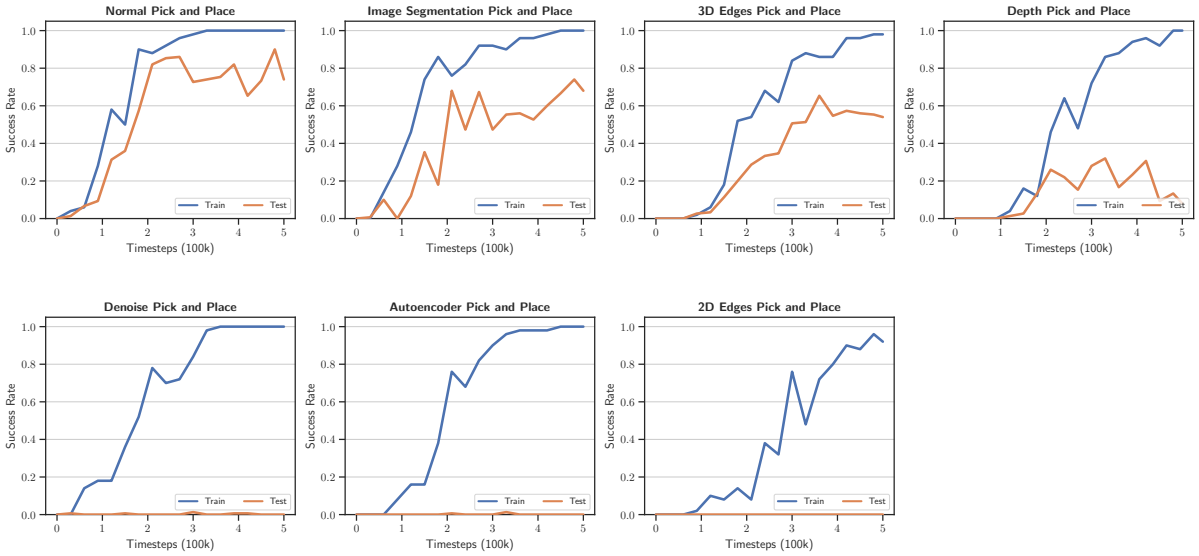


Figure B.7: Pick and place policies: reward vs. number of training iterations

Train/test curves for policies trained with dense reward

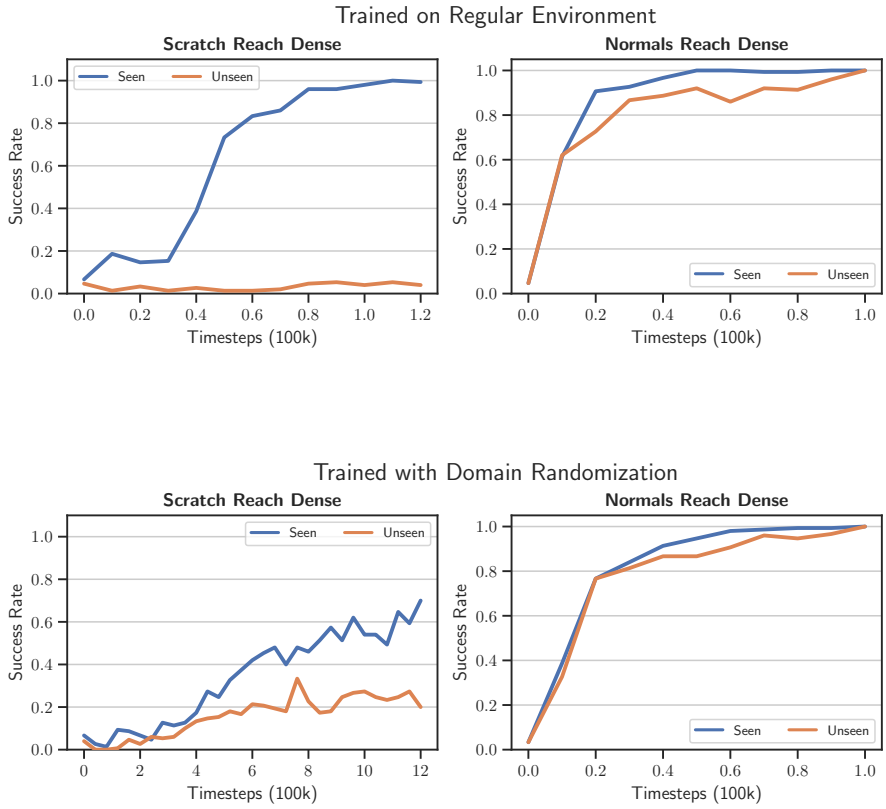


Figure B.8: **Train/test curves for policies trained with dense reward.** The reward engineering improved performance and was necessary to get pixels to train at all. However, we found the approach scaled poorly as harder tasks required prohibitive amounts of engineering.

Train/test curves for policies trained with additional objects

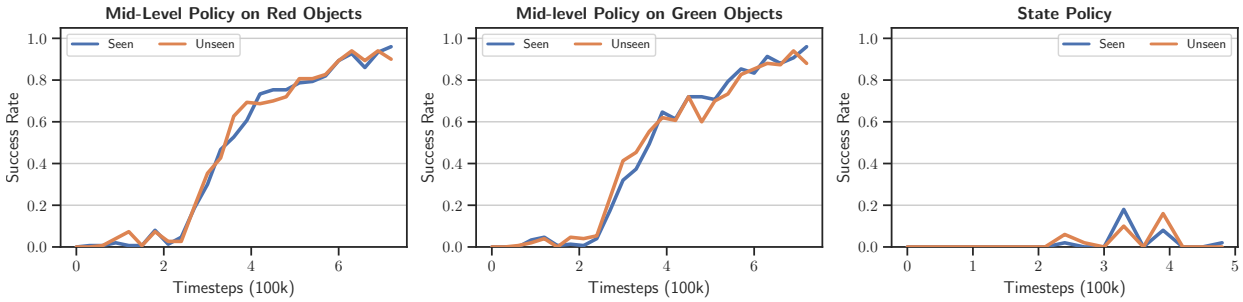


Figure B.9: **Train/test curves for policies trained to pick + place various objects.**

Test curves for policies trained from state

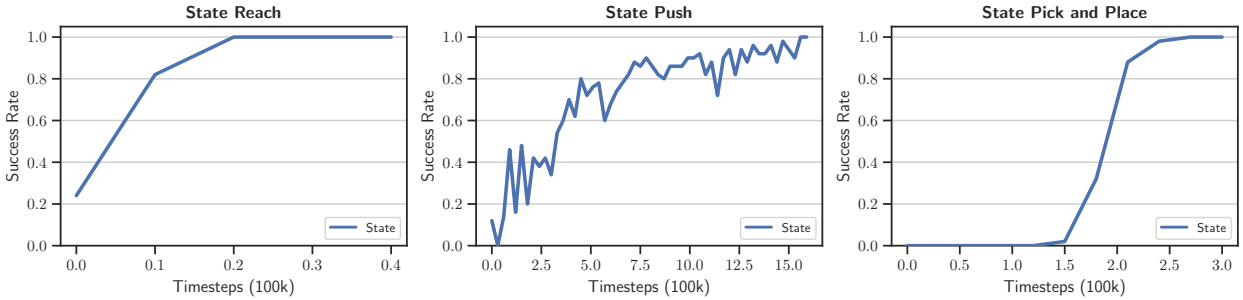


Figure B.10: **Train/test curves for policies trained from true environment state.**

Appendix C

Chapter 4 Supplementary Material

Supplementary Material
http://consistency.epfl.ch

The supplementary material provides qualitative and quantitative results and method details which were moved out of the main paper in interest of space. Specifically:

I. A video evaluation showing various networks applied frame-by-frame to a Youtube video (§ C.1).
II. A live demo where you can apply our networks to your own query images (§ C.2).
III. Results of consistency with unsupervised tasks (§ C.3).
IV. Discussion of how the perceptual loss formulation handles ill-posed tasks and associated experiments (§ C.4).
V. Description of our strategy for balancing different loss terms (§ C.5).
VI. Plots showing that minimizing the direct term does not minimize consistency (§ C.6).
VII. Derivation of generic path length criterion (§ C.7).
VIII. Sensitivity analysis: Comparison of different edge selection strategies (§ C.8).
IX. Sensitivity analysis: Training with consistency over multiple path lengths (§ C.9).
X. Results with standard error over multiple initialization seeds (§ C.10).
XI. Results on *NYUv2* dataset (§ C.11).
XII. Results on *Taskonomy* dataset reported with *additional perceptual tasks* (§ C.12) and using common *task-specific metrics* (e.g., mean and median angular error for surface normals) (§ C.12).
XIII. More qualitative results provided in § C.13 and on the project website.
XIV. More qualitative results on estimating surface normal out of middle domains before and after enforcing consistency. (§ C.13).

XV. Definition and visualizations of the "Blind Guess" (statistically informed guess) for the Taskonomy dataset (§ C.14).

XVI. Code: including pretrained models, runnable examples, and a Docker (§ C.15).

C.1 Video Evaluation

The project website includes video clips that provide various results from the proposed framework as well as the baselines. In particular, the video clips provide results of different stages of the method applied on the YouTube video used in [237], which we find insightful. We recommend watching the clips.

C.2 Live Demo

The project website includes a live demo that allows you to upload your own query images. The demo will run that query through our servers and return the results, so that you can visualize the predictions for various tasks from networks trained with and without consistency, as well as compare to baselines. The demo page also contains a link to the "demo archive" where you can browse uploads from other users.

C.3 Consistency with Unsupervised Tasks

As described in the main paper, the tasks a network is constrained to be consistent with could be *unsupervised* or *self-supervised* too. This is particularly useful if the dataset in hand is either single-task (as most common datasets are) or includes few tasks. Unsupervised tasks allow generating new domains without any additional supervision, thus enable utilizing denser cross-task consistency constraints during training. Examples of such tasks are 2D texture edges and 2D keypoints (SURF[22]), which were included in our dictionary.[1] Such tasks have fixed operators thus can be applied on any images to produce their respective domains with no additional supervision.

Interestingly, enforcing cross-task consistency with such unsupervised domains, even without using any supervised ones, still led to better fitting the data and producing improved predictions. Fig. C.1 shows the results of learning to predict surface normals, while the two consistency domains were 2D texture edges and 2D keypoints. In other words, using the notation $x \rightarrow y_1 \rightarrow Y$, here x was RGB image, y_1 was surface normals, and the two domains in Y were 2D edges and 2D keypoints. The loss was $\mathcal{L}^{perceptual}_{xy_1Y}$.

The unsupervised consistency domains improved certain relevant features in the predicted normals. We expect the improvement to increase with more tasks added, though the gain will plateau beyond a certain point as the tasks will start to become redundant. Thus we expect the

[1] See [237] for more examples. Any task that has a fixed operator is a candidate.

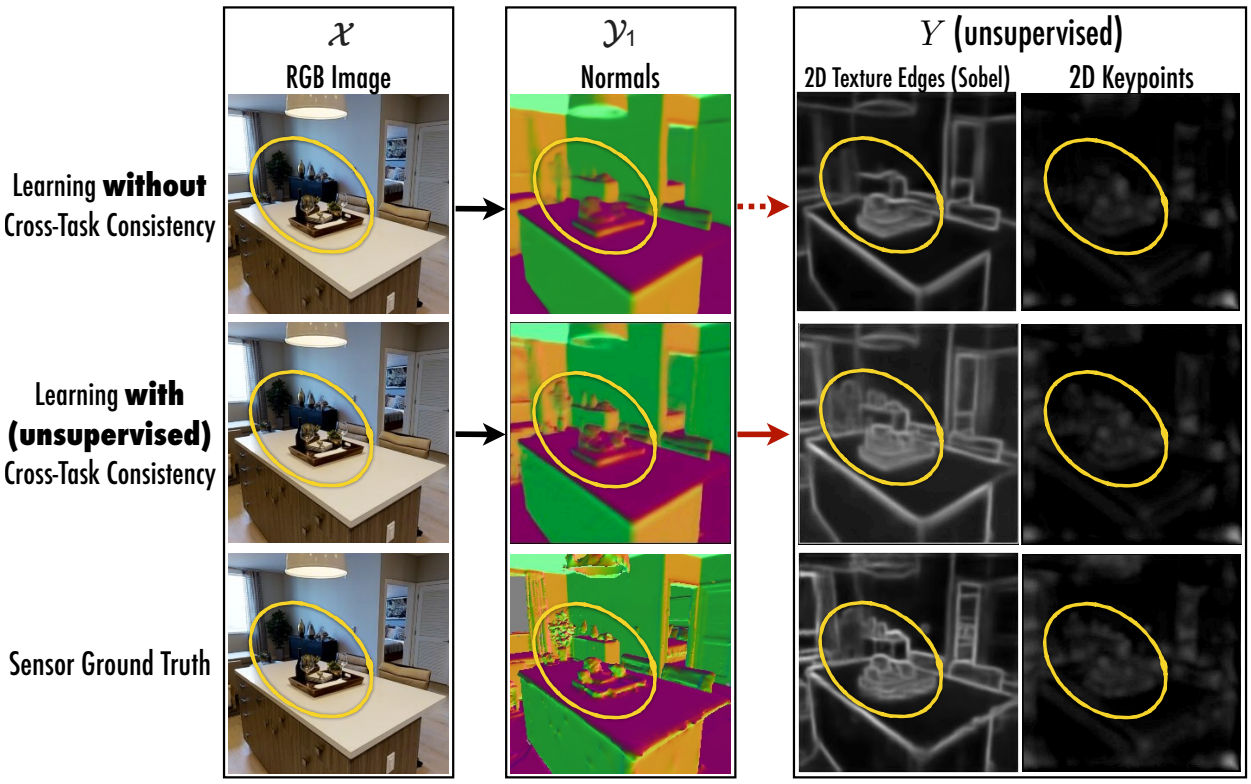


Figure C.1: **Learning with cross-task consistency with *unsupervised domains* shown for a sample query**. The conventions of the figure is the same as Fig. 4 of the main paper. Using the notation $x \rightarrow y_1 \rightarrow Y$, here x is RGB image, y_1 is surface normals, and two domains in Y are 2D texture edges (Sobel filter) and 2D keypoints., depth, and occlusion edges. Learning with Cross-Task Consistency constraints still improved the results, even though no supervision was used in the introduced consistency constraints.

consistency tasks would need to be 'engineered' beyond a certain point to squeeze out further improvements.

Generally speaking, the usefulness of unsupervised tasks extends the applicability of the proposed method, in terms of improving the fit to the data, to single/few task datasets.

C.4 Handling of Ill-Posed Tasks

As noted in the introduction and Section 3.1.2 of the main paper, the task sets we consider may not be *informationally equivalent* – in the sense in that, in theory, one may not always be able to convert one to the other without some uncertainty. For example, we can expect to derive normals from depth, but the opposite direction is ill-posed. Therefore employing such an ill-posed link $normal \rightarrow depth$ as a loss for training a task like $RGB \rightarrow normal$ (in other words the triangle $RGB \rightarrow normal \rightarrow depth$) may appear problematic. That would be because, in theory it is impossible to infer the correct depth

even given ground truth normals. This is what causes cycle consistency to 'hide' information in the predicted image, shown in Fig. C.2. Here we describe three points toward resolving this issue:

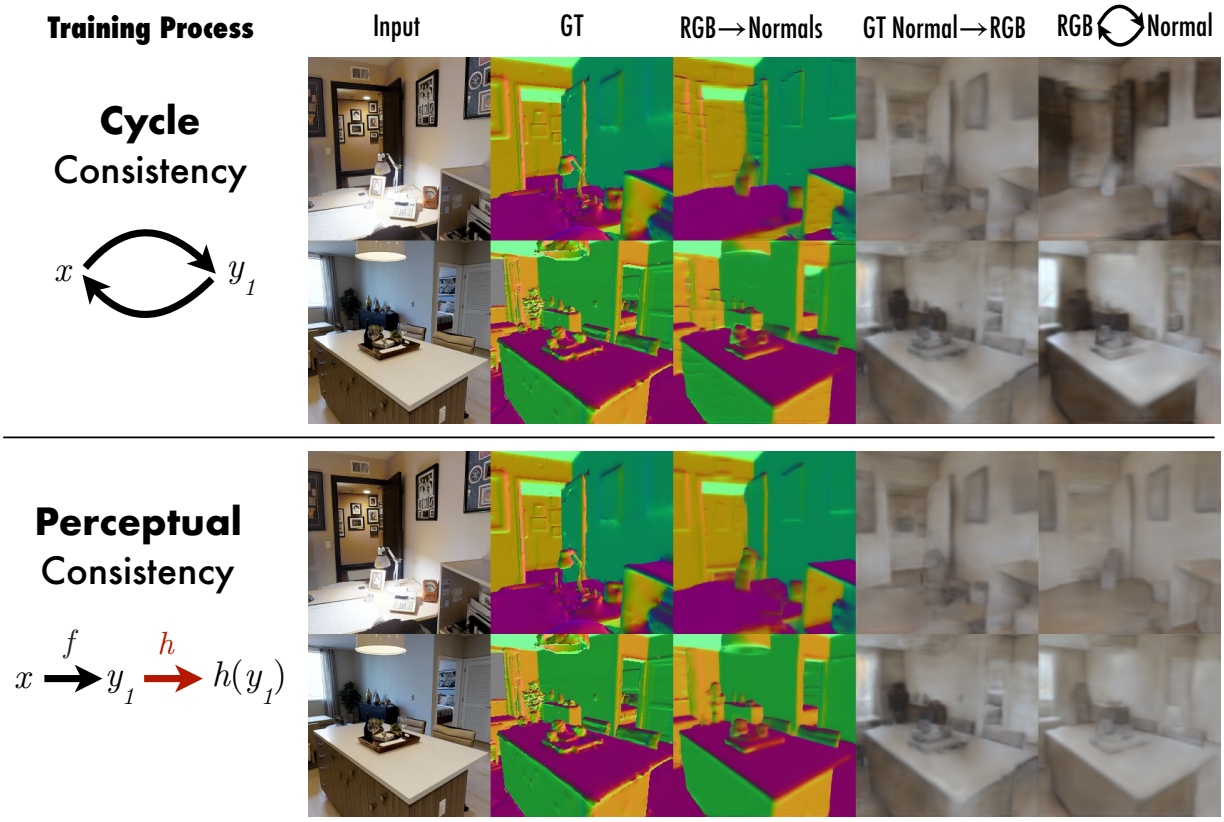


Figure C.2: **Cycle consistency vs. the proposed perceptual formulation**. **Top**: Using naive cycle consistency for ill-posed tasks (e.g. predicting RGB from surface normals) leads to the well-known problem where networks hide hide information about the higher-information domain in the lower-information prediction. In this case, the network encodes dark areas of the RGB image with a grid-like pattern in the predicted surface normals. **Bottom:** The proposed perceptual formulation eliminiates this degenerate solution and, as expected, the artifacts disappear. Surface normal prediction quality also improves.

I) Most links are not ill-posed to learn, despite analytical definitions: Even though a link may be ill-posed in theory, the contextual information visible in the datapoint often resolves the uncertainty; just like the fact that when humans look at ground truth normals, they can infer the depth of the same scene based on the semantics and contextual information visible in the normal image. We found that the trained neural networks were surprisingly good at using this knowledge. This is well presented in Fig.4-(lower row) of the main paper, where the normals could be well predicted out of intuitively surprising and ill-posed tasks (e.g. see $TextureEdges \rightarrow normal$ or $3DCurvature \rightarrow normal$ or $SurfKeypoints \rightarrow normal$). Hence most of the links in the complete

graph of tasks turn out to be not significantly ill-posed, despite their analytical definition. This also underscores the advantages of a fully computational and data-driven method vs those that primarily rely on analytical relationships [166], as many of the cross-task links we successfully use would be analytically ill-posed.

II) The perceptual loss formulation handles the residual error: Even if a link is not ill-posed, we still learn them using neural networks, which likely results in a small but non-zero loss after convergence. This means the link $\mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$ will *not* yield the perfect $\mathcal{Y}_2^*$ even if provided with perfect $\mathcal{Y}_1^*$ in the input, i.e. $f_{\mathcal{Y}_1\mathcal{Y}_2}(\mathcal{Y}_1^*) \neq \mathcal{Y}_2^*$. Thus, there will always be residual imperfections in the link, and consequently when trying to employ it to optimize the task $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1$ using the triangle $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$, the imperfections of $\mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$ may corrupt the link $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1$.[2] (Again, that will be due to the fact that even outputting perfect $\mathcal{Y}_1^*$ by $f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X})$ would not result in estimating $\mathcal{Y}_2^*$ in the end of the triangle $f_{\mathcal{Y}_1\mathcal{Y}_2} \circ f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X})$ since $f_{\mathcal{Y}_1\mathcal{Y}_2}(\mathcal{Y}_1^*) \neq \mathcal{Y}_2^*$.) Therefore using a loss of the form $\mathcal{L} = \|f_{\mathcal{Y}_1\mathcal{Y}_2} \circ f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X}) - \mathcal{Y}_2^*\|$ in the framework to train $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1$ would be problematic and corrupt the network.

We handle this by adopting a *perceptual loss* [103] based formulation. This basically means we use the loss

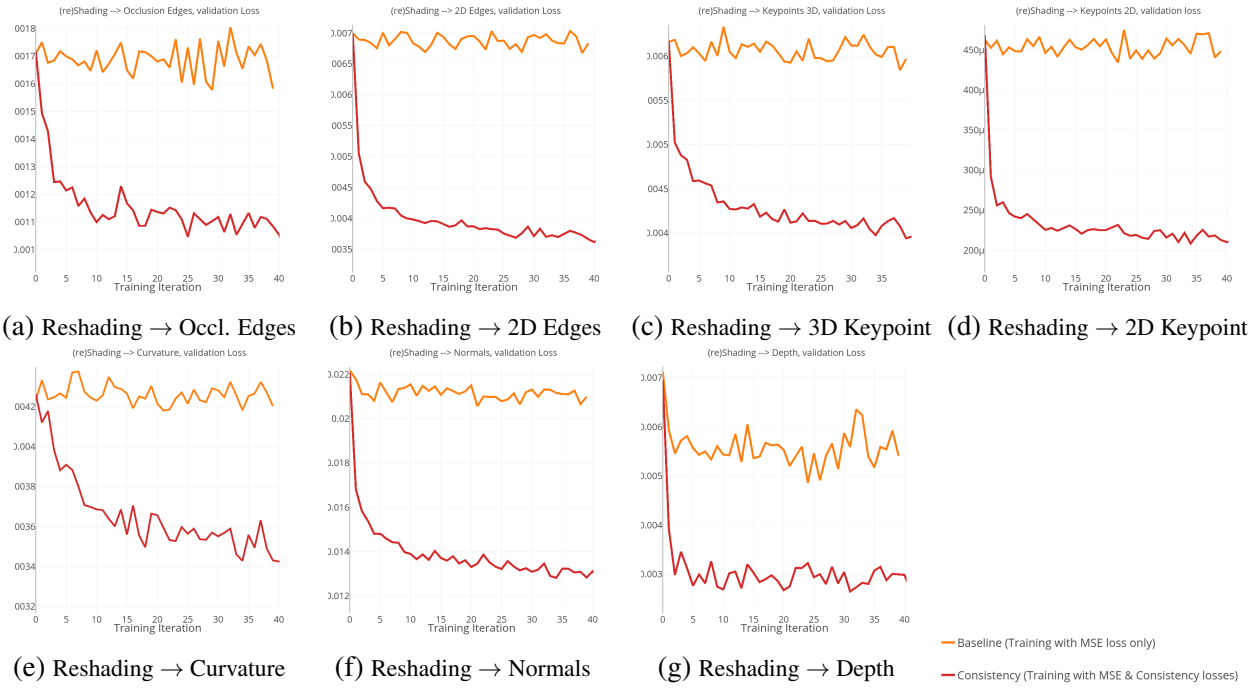
$$\mathcal{L} = \|f_{\mathcal{Y}_1\mathcal{Y}_2} \circ f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X}) - f_{\mathcal{Y}_1\mathcal{Y}_2}(\mathcal{Y}_1^*)\|, \tag{C.1}$$

in lieu of $\mathcal{L} = \|f_{\mathcal{Y}_1\mathcal{Y}_2} \circ f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X}) - \mathcal{Y}_2^*\|$. This simple trick enforces that predicting perfect $\mathcal{X}_2^*$ by $f_{\mathcal{X}\mathcal{Y}_1}(\mathcal{X})$ would result in exactly loss 0 in the triangle loss, hence the residual imperfections of $\mathcal{Y}_1 \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_2$ would not propagate to learning of $\mathcal{X} \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_1$ via the triangle loss $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$.

In addition, the above perceptual loss based formulation makes training datasets with pair annotations $(\mathcal{X}, \mathcal{Y}_1)$ & $(\mathcal{Y}_1, \mathcal{Y}_2)$, rather than triplet $(\mathcal{X}, \mathcal{Y}_1, \mathcal{Y}_2)$ or higher order annotations, sufficient for learning with the cross-task consistency triangle $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$. That is because the ground truth for the third domain $\mathcal{Y}_2^*$ is not used in the above perceptual loss in Eq. C.1, hence at no point all three domains $(\mathcal{X}, \mathcal{Y}_1, \mathcal{Y}_2)$ need be *simultaneously* known for one datapoint.

III) Removing the low-mutual-information edges: If a link between two domains is severely problematic to the extent that the points **I** and **II** do not address it, we simply remove them from the complete task graph. We do that by performing vanilla pre-training of links between all pairs of domains $(\mathcal{X}_a \times \mathcal{X}_b)$ and removing those that do not results in a sufficiently low loss. However, we barely faced such links (see point **I** above) and noticed removing or not removing them did not make a notable difference in the overall results after optimizing the entire system.

[2] Note that $\mathcal{X} \xrightarrow{f_{\mathcal{X}\mathcal{Y}_1}} \mathcal{Y}_1 \xrightarrow{f_{\mathcal{Y}_1\mathcal{Y}_2}} \mathcal{Y}_2$ is the same as $\mathcal{X} \longrightarrow \mathcal{Y}_1 \longrightarrow \mathcal{Y}_2$ in the rest of the paper, just annotated with functions for ease of followed notations.



(a) Reshading → Occl. Edges (b) Reshading → 2D Edges (c) Reshading → 3D Keypoint (d) Reshading → 2D Keypoint

(e) Reshading → Curvature (f) Reshading → Normals (g) Reshading → Depth

Figure C.3: **Optimizing the standard direct loss does not lead to optimizing cross-task losses.** Various cross-task losses are plotted for a network that is being trained only for the standard direct MSE loss (orange curve) and one that is being trained with the standard direct MSE loss as well as cross-task losses (red curve). The network being optimized with only the direct loss does not reduce the perceptual losses, despite the fact that the direct loss was being successfully optimized till full convergence. This echos the necessity of augmentation the training process with explicit losses based on cross-task consistencies.

C.5 Balancing Different Loss Terms

As discussed in the main paper, the final (total) loss is:

$$\sum_{i=1}^{N}\left(\left|f_{\mathcal{X}\mathcal{Y}_i}(x)-y_i\right|+\lambda\sum_{j=1}^{N}\left|f_{\mathcal{Y}_i\mathcal{Y}_j}\circ f_{\mathcal{X}\mathcal{Y}_i}(x)-f_{\mathcal{Y}_i\mathcal{Y}_j}(y_i)\right|\right),$$

which we can rewrite as:

$$\sum_{i=1}^{N}\left(\mathcal{L}^{\text{direct}}+\lambda\sum_{j=1}^{N}\mathcal{L}^{\text{percept}}_{\mathcal{X}\mathcal{Y}_i\mathcal{Y}_j}\right),$$

One issue is that if the number of perceptual losses is large, the perceptual losses will come to dominate the loss. We found that choosing λ to compensate for this effect improved the quality of the final networks.

We set λ per-batch and per-loss in order to make the magnitude of the perceptual losses' total gradient contribution independent of the number of perceptual losses used. Specifically, we set λ as follows:

$$\lambda_{ij}(x) = \frac{|\nabla \mathcal{L}^{\text{direct}}| + \sum_{k \neq i,j} |\nabla \mathcal{L}^{\text{percept}}_{\mathcal{X}\mathcal{Y}_i\mathcal{Y}_j}|}{(N-1)\left(|\nabla \mathcal{L}^{\text{direct}}| + \sum_k |\nabla \mathcal{L}^{\text{percept}}_{\mathcal{X}\mathcal{Y}_i\mathcal{Y}_j}|\right)},$$

where $|\cdot|$ is the ℓ^1 norm.

C.6 Optimizing the standard direct loss does not lead to optimizing cross-task losses

In this section we experimentally demonstrate that optimizing the standard direct loss (e.g. MSE loss) for a certain task does not naturally lead to optimizing the related cross-task losses; even if the direct loss appears to go down. In other words, optimizing for $\mathcal{X}\rightarrow\mathcal{Y}_1$ does not substantially optimize the red cross-task loss in $\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2$.

This is demonstrated in Fig. C.3, where various cross-task losses (i.e. the '$\rightarrow$'s) are plotted for a network that is being trained only for the standard direct MSE loss (orange curve) and one that is being trained with the standard direct MSE loss as well as cross-task losses (red curve). For both cases, the optimization starts from a baseline MSE-only network that is halfway to its convergence. As apparent in the figure, the network being optimized with only the direct loss does not reduce the perceptual losses, despite the fact that the direct loss was being successfully optimized till full convergence. This is while the global minimum of both the direct loss ($\mathcal{X}\rightarrow\mathcal{Y}_1$) and the cross-task losses ($\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2$) are the same (predicting the ground truth for $\mathcal{Y}_1$). This again echoes that, given the existing sub-optimal optimizers, the process of training neural networks benefits from explicit augmentation of cross-task consistency based losses. This is inline with the quantitative and qualitative results provided in the main paper.

The results were reported using (re)Shading task. The conclusion was the same for other tasks.

C.7 Derivation of Generic Consistency Criterion

In the main paper we derived the triangle consistency constraint. The derivation for the general path-consistency constraint is similar.

Here's the setup: say that we have two paths ($P = P_1 ... P_{k-1} P_\ell$ and $Q = Q_1 ... Q_{\ell-1} Q_\ell$). We want to train the first edge in the path P, $f_{\mathcal{X}P_1}$. Assume this edge does not also appear in Q. Note that we are requiring $P_1 = Q_1 = \mathcal{X}$ and $P_k{=}Q_\ell$. We can then define the path composition function:

$$f_P(x) \triangleq f_{P_{k-2}P_{k-1}} \circ f_{P_{k-1}P_k} \circ ... \circ f_{\mathcal{X}P_1}(x),$$

and the analogous functions: f_Q for Q, and $f_{P_{1:k'}}$ for subpaths[3] $P_{1:k'}$ of P. The associated consistency constraint is:

$$|f_P(x) - f_Q(x)|$$

[3] $P_{1:k'}$ is the same as P up until and including $P_{k'}$ for $k' \leq k$.

Path selection method	MSE (↓)
Maximally Violating Path	**1.78**
Random Path Selection	1.83
All Paths	1.88

Table C.1: Comparison of different path selection strategies in message passing.

.

The derivation requires mapped triplets $(x, p_{k-1}, p_k) \sim \mathcal{X} \times P_{k-1} \times P_k)$. In contrast, the final equation (and, therefore, training) only requires paired data $(x, p_{k-1}) \sim \mathcal{X} \times P_{k-1}$.

Proof. In the derivation, we'll absorb terms that are constant (w.r.t. the optimization parameters) into some catch-all, C.

$$
\begin{aligned}
&|f_{_P}(x) - f_{_Q}(x)| \\
&\quad \leq |f_{_P}(x) - p_k| + |p_k - f_{_Q}(x)| \\
&\quad = |f_{_P}(x) - p_k| + C \\
&\quad \leq |f_{P_{k-1}P_k} \circ f_{P_{1:k-1}}(x) - f_{P_{k-1}P_k}(p_{k-1})| \\
&\qquad\qquad + |f_{P_{k-1}P_k}(p_{k-1}) - p_k| + C \\
&\quad = |f_{P_{k-1}P_k} \circ f_{P_{1:k-1}}(x) - f_{P_{k-1}P_k}(p_{k-1})| + C
\end{aligned}
$$

□

C.8 Sensitivity Analysis: Edge Selection

As described in Sec.3.2 of the main paper, multiple possibilities exists for selecting the path to be optimized at each iteration. We adopted selecting the most inconsistent (*Maximally Violating*) path at the time. We provide a discussion and experimental justification here.

Table C.1 compares different strategies for selecting the path to optimize in message passing (i.e. *SelectNetwork* in Algorithm 1). *Random* Path Selection represents selecting one of the feasible paths $\mathcal{X} \to \mathcal{Y}_1 \to \mathcal{Y}_2$ randomly. *Maximally Violating* Path represents selecting the most inconsistent path at the time. That is akin to optimizing to reduce the upper bound of inconsistency in the system. *All* Paths represents randomly picking a $\mathcal{X} \to \mathcal{Y}_1$ then using *all* of the perceptual losses that start from $\mathcal{Y}_1$, with the total loss being the sum of all of them. Consequently, this method is slow and has a large memory requirement as all many networks are in use at the same time.

As mentioned in the main paper, the results in Table C.1 shows picking the *Maximally Violating* Path gave the best results, though there was not a significance difference among the methods in terms of the final performance. In terms of the computation and memory requirements, using *all* paths is significantly more expensive. This signifies the potential value in adopting a proper selection criterion in message passing. We used *Maximally Violating* selection method in our experiments.

The amount of violation/inconsistency of a path $\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2$ is defined to be the loss of that path at the predicting $\mathcal{Y}_2$ given $\mathcal{X}$, normalized by a fixed constant which was found by via grid search.

Table C.1 is reported using (re)Shading task. The conclusions remain the same for different tasks.

C.9 Sensitivity Analysis: Path Lengths

The proposed method is applicable to optimizing cross-task consistency with any inferences path lengths. By path length, we are referring to the the number of cascaded edges in $\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2\rightarrow\mathcal{Y}_3\rightarrow...$. Here we provide a discussion and experiments on this aspect. Table C.2 compares the final results of optimizing with paths with maximum length 3 vs maximum length 2.

Method	MSE
Path length 3 ($\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2\rightarrow\mathcal{Y}_3$)	1.80
Path length 2 ($\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_3$)	1.78

Table C.2: Impact of different path lengths in optimization.

As described in the main paper, there was a negligible difference associated with using longer paths in our experiments. We believe this is due to the fact that our task graph is nearly complete, hence for any path $\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_2\rightarrow\mathcal{Y}_3$ with length 3, the corresponding path with length 2 with the same end points, i.e. $\mathcal{X}\rightarrow\mathcal{Y}_1\rightarrow\mathcal{Y}_3$, is also included in the optimization. Hence the cross-task consistency value of a path with length 3 is already enforced by a path with length 2, as far as optimizing the link $\mathcal{X}\rightarrow\mathcal{Y}_1$ is concerned. The same discussion applies to paths longer than 3 as well. Hence, as described in Sec. 3.2, we limited the length of sampled paths to 2 in our experiments as shorter paths are computationally cheaper. However, the proposed framework is applicable to arbitrary lengths, especially if the task graph is not complete.

Table C.2 is reported using (re)Shading task. The conclusion was the same for other tasks.

C.10 Standard Error Over Multiple Seeds

The tables below add *Standard Error* columns to the tables 1-3 of the main paper. The Standard Error is reported over multiple (3) independent runs ("**Std. Err. (3 Runs)**") of our network and the main baseline as well as across test set images ("**Std. Err. (Test Set)**"). The low Standard Error denotes statistical significance in the observed trends. Also, please note that the reported improvement margins (e.g. 0.87 vs 1.00) are quite significant (13% relative improvement) and geometrically noteworthy.

Surface Normal Estimation Method	MSE	**Std. Err. (Test Set)**	**Std. Err. (3 Runs)**
Baseline (UNet with MSE loss)	1.00	**0.0037**	**0.0059**
Consistency (UNet with consistency loss)	0.87	**0.0052**	**0.0050**

	Error	
Method (Depth)	MSE	**Std. Err. (Test)**
Baseline (UNet MSE)	0.40	**0.0006**
Consistency	0.36	**0.0007**

	Error	
Method (Reshading)	MSE	**Std. Err. (Test)**
Baseline (UNet MSE)	2.20	**0.0005**
Consistency	1.78	**0.0003**

Table C.3: **Tables 1-3 of main paper with addition of *Standard Error* values (in bold)**. Results show statistical significance in observed trends, as the standard errors indicate that these results are extremely unlikely to occur from chance.

(Surface Normal Est.)	Error (↓)				Accuracy (↑)		
Method	Mean°	Med.°	L1	MSE	≤11.25°	22.5°	30°
Baseline (UNet)	**9.91**	**6.16**	**8.65**	**2.08**	**0.71**	**0.88**	**0.93**
Cycle Consistency	12.83	9.85	11.01	2.63	0.57	0.83	0.91
GeoNet (Impr.)	10.27	6.47	8.97	2.22	0.70	0.87	0.92
GeoNet (Orig.)	**10.08**	6.49	**8.79**	**2.09**	**0.71**	**0.88**	**0.93**
Multitask	11.63	7.50	10.15	2.69	0.66	0.84	0.91
Pix2Pix	13.07	10.04	11.29	2.72	0.57	0.84	0.91
Prcpt. Loss (ImageNet)	11.06	7.14	9.65	2.53	0.68	0.85	0.91
Prcpt. Loss (Random)	11.29	7.45	9.85	2.57	0.66	0.85	0.91
Consistency	**9.88**	**6.06**	**8.62**	**2.11**	**0.71**	**0.88**	**0.93**

Table C.4: **Evaluating surface normal prediction on NYU v2 test set.** Values **bolded and starred*** indicate the best-performing method. Values that are **bolded** but not starred indicate methods that were statistically indistinguishable from the best-performing method (2-sample paired t-test, $\alpha = 0.01$).

C.11 Results on NYUv2 Dataset

We used NYU v2 [193] dataset in addition to Taskonomy [237] and Replica [203] for evaluations. Table C.4 shows our method and the main baseline, denoting that trends remain the same. Note that NYU v2 was used for evaluation only, so the training dataset remained Taskonomy. These results show the evaluation trends reported in the main paper were not specific to Taskonomy dataset. They also suggest Taskonomy provided a better training data for achieving good geometric prediction results compared to NYU v2 (as training our baseline UNet on Taskonomy and testing on NYU v2 reported better results compared to training on NYU v2 and testing on NYU v2– see Table 1 of [166]) possibly due to its larger size and type of sensor ground truth. Hence we adopted Taskonomy as the training dataset for a more reliable experimentation.

C.12 More Metrics

We provide additional evaluations on the Taskonomy and Replica datasets here in the supplementary material to show that the results are not specific to the ℓ^1-norm. Specifically we provide additional task-specific and perceptual metrics that were omitted, for space, from Table 1 in the main paper. The trends and conclusions remain the same as in the main paper.

Additional Task-Specific Metrics for Direct Prediction

Some tasks (surface normal estimation, depth estimation) also have additional common standard metrics beyond MSE or ℓ^1. Common task-specific metrics evaluating models on direct prediction of RGB to X are provided for

- **Surface normals**: Angular error metrics are provided for Taskonomy (Table C.10) and Replica (Table C.7), and NYUv2 (Table C.4).
- **Depth estimation**: Depth error metrics are provided for Taskonomy (Table C.8) and Replica (Table C.5).

For reShading, since there are not commonly-accepted metrics, we provide both L1 and MSE error.

Common angular error metrics: The Tables C.10 & C.7 show five metrics beyond L1 and MSE: mean and median of angular surface normal error, and the proportion of predictions within 11.25°, 22.5°, and 30° of the ground-truth. The trends and conclusions remain the same as in the main paper.

Common depth error metrics: The Tables C.8 & C.5 show the same results as the direct L1 prediction from Table 1 in the main paper, with the addition of standard depth estimation metrics: Relative Error (Rel. Err), Scale Invariant Logarithmic error (SILog, main benchmark for KITTI [67]), Inverse Root Mean Squared Error (IRMSE, RMSE is used for NYUv2 [193]) and Logarithmic Error ($\log_{10}$). The trends and conclusions remain the same as in the main paper.

Additional Perceptual Metrics

We also provide additional perceptual metrics for both Replica and Taskonomy. Fig. C.7 shows results using additional perceptual tasks (*Keypoints 2D, Keypoints 3D*, and *Edges 3D*) and metrics (MSE) to those shown in Table 1 on Taskonomy in the main paper. In Replica, there are currently only the three provided tasks, so we show MSE loss in addition to L1 from the main paper. The trends and conclusions remain the same as in the main paper.

In general, evaluation of high dimensional regression tasks, in comparison to lower dimensional classification, is more challenging as often no single metric captures all the desirable properties of a prediction. Thus using multiple metrics simultaneously provide a more complete evaluation picture for such cases.

Replica Direct Prediction (Additional Metrics)

(Depth Estimation) Method	Error (↓) L1	MSE	IRMSE	Log10	SI Log
Baseline UNet	1.99	0.07	**0.00***	0.13	**0.01***
Constant Pred.	4.81	0.38	0.00	0.32	0.07
GeoNet (Impr.)	1.83	0.06	**0.00**	0.13	**0.01**
GeoNet (Orig.)	4.01	0.30	0.00	0.23	0.04
Multitask	2.44	0.11	0.00	0.16	0.02
Taskonomy	3.72	0.24	0.00	0.23	0.02
Consistency	**1.63***	**0.05***	**0.00**	**0.12***	**0.01**

Table C.5: **Depth estimation on Replica.**

(reShading) Method	Error (↓) L1	MSE
Baseline UNet	**9.55**	**1.77**
Constant Pred.	16.45	4.35
Multitask	10.32	1.95
Taskonomy	11.43	2.27
Consistency	**9.22***	**1.69***

Table C.6: **ReShading estimation on Replica.**

(Surface Normal Est.) Method	Error (↓) Mean°	Median°	L1	MSE	Accuracy (↑) ≤11.25°	22.5°	30°
Baseline UNet	5.76	**2.63**	4.96	1.02	0.87	0.95	0.97
Constant Pred.	19.13	16.27	16.02	4.94	0.36	0.65	0.77
Cycle Consistency	8.36	4.84	7.13	1.54	0.77	0.91	0.95
GeoNet (Impr.)	**5.46***	**2.59**	**4.70***	**0.95***	**0.88***	**0.95***	**0.97***
GeoNet (Orig.)	11.50	7.19	7.48	1.98	0.68	0.86	0.91
Multitask	7.02	3.51	6.03	1.45	0.84	0.92	0.95
Pix2Pix	9.03	5.95	7.70	1.60	0.75	0.91	0.95
Prcpt. Loss (ImageNet)	**5.62**	**2.57***	4.85	**0.99**	0.87	**0.95**	**0.97**
Prcpt. Loss (Random)	5.78	2.74	4.99	1.00	0.86	**0.95**	**0.97**
Taskonomy	19.13	16.27	16.02	4.94	0.36	0.65	0.77
Consistency	**5.60**	**2.63**	**4.80**	**0.99**	0.88	**0.95**	**0.97**
0.25% Data: Baseline (UNet)	8.83	4.13	7.61	2.27	0.78	**0.89**	**0.91**
0.25% Data: Consistency	**8.46**	**3.77**	**7.28**	**2.05**	**0.79**	**0.89**	**0.92**

Table C.7: **Surface normal estimation on Replica.**

Figure C.4: **Extended quantitative results on Replica.** Values **bolded and starred*** indicate the best-performing method. Values that are **bolded** but not starred indicate methods that were statistically indistinguishable from the best-performing method (2-sample paired t-test, $\alpha = 0.01$)

Taskonomy Direct Prediction (Additional Metrics)

(Depth Estimation) Method	Error (↓) L1	MSE	IRMSE	Log_{10}	Rel. Err.	SI Log
Baseline (UNet)	**2.27**	**0.17**	**0.00**	**0.13**	**0.17**	**0.02**
Constant Pred.	7.07	0.87	0.00	0.41	0.63	0.07
GeoNet (Impr.)	**2.26***	**0.16***	**0.00***	**0.13***	**0.17***	**0.02**
GeoNet (Orig.)	4.07	0.44	0.00	0.22	0.31	0.05
Multitask	2.81	0.20	0.00	0.17	0.22	0.02
Taskonomy	4.55	0.44	0.00	0.26	0.28	0.03
Consistency	**2.29**	**0.16**	**0.00**	0.13	**0.17**	**0.02***

Table C.8: **Depth estimation on Taskonomy.**

(reShading) Method	Error (↓) L1	MSE
Baseline (UNet)	**10.45***	**3.21**
Constant Pred.	24.85	8.91
Multitask	11.61	3.36
Taskonomy	16.58	5.36
Consistency	**10.52**	**3.21***

Table C.9: **ReShading estimation on Taskonomy.**

(Surface Normal Est.) Method	Error (↓) Mean°	Median°	L1	MSE	Accuracy (↑) ≤11.25°	22.5°	30°
Baseline (UNet)	**6.86**	**2.42**	**5.95**	**1.58**	**0.81**	**0.91**	**0.94**
Constant Pred.	21.06	19.14	17.80	5.73	0.27	0.56	0.76
Cycle Consistency	10.09	6.29	8.68	2.06	0.69	0.87	0.93
GeoNet (Impr.)	**6.81***	**2.37***	**5.91***	**1.57***	**0.81***	**0.91***	**0.94***
GeoNet (Orig.)	15.49	11.38	9.58	2.86	0.51	0.76	0.86
Multitask	8.17	3.66	7.07	1.84	0.78	0.89	0.93
Pix2Pix	10.92	7.28	9.40	2.26	0.68	0.87	0.92
Prcpt. Loss (ImageNet)	6.98	2.50	6.06	1.62	0.81	0.91	0.94
Prcpt. Loss (Random)	7.10	2.68	6.17	1.62	0.81	0.91	0.94
Taskonomy	8.70	4.21	7.54	1.96	0.76	0.89	0.93
Consistency	7.01	2.52	6.08	1.63	0.81	0.91	0.94
0.25% Data: Baseline	**9.43**	**4.05**	**8.17**	**2.43**	**0.74**	**0.86**	**0.91**
0.25% Data: Consistency	10.63	5.17	9.19	2.78	0.70	0.84	0.89

Table C.10: **Surface normal estimation on Taskonomy.**

Figure C.5: **Extended quantitative results on the Taskonomy test set.** Values **bolded and starred*** indicate the best-performing method. Values that are **bolded** but not starred indicate methods that were statistically indistinguishable from the best-performing method (2-sample paired t-test, $\alpha = 0.01$)

Replica Perceptual Results (Extended)

Depth Est. → Method	Surface Normal L1 (↓)	Surface Normal MSE (↓)	reShading L1 (↓)	reShading MSE (↓)
Baseline UNet	10.47	3.08	12.99	3.12
Constant Pred.	22.23	7.80	19.94	6.00
GeoNet (Impr.)	10.47	3.07	12.75	3.03
GeoNet (Orig.)	13.88	5.03	14.03	4.30
Multitask	15.30	5.33	16.14	4.56
Taskonomy	18.06	6.04	15.39	3.86
Consistency	**7.01***	**1.71***	**11.21***	**2.57***

Table C.11: **Perceptual results for depth estimation on Replica.**

reShading → Method	Surface Normal L1 (↓)	Surface Normal MSE (↓)	Depth Estimation L1 (↓)	Depth Estimation MSE (↓)
Baseline UNet	6.90	1.59	2.74	0.12
Constant Pred.	15.74	5.13	5.14	0.41
Multitask	7.24	1.81	3.36	0.18
Taskonomy	8.70	2.41	3.85	0.23
Consistency	**5.50***	**1.18***	**1.96***	**0.07***

Table C.12: **Perceptual results for reShading on Replica.**

Surface Normal → Method	Depth Estimation L1 (↓)	Depth Estimation MSE (↓)	reShading L1 (↓)	reShading MSE (↓)
Baseline UNet	4.69	0.37	13.15	2.80
Constant Pred.	4.75	0.43	33.31	15.15
Cycle Consistency	5.65	0.54	22.39	7.30
GeoNet (Impr.)	4.62	0.36	12.79	2.65
GeoNet (Orig.)	6.23	0.81	19.34	7.24
Multitask	5.58	0.53	22.11	7.20
Pix2Pix	4.52	0.39	19.03	5.87
Prcpt. Loss (ImageNet)	3.45	0.20	**8.31***	**1.43***
Prcpt. Loss (Random)	4.88	0.41	15.34	3.61
Taskonomy	3.73	0.26	33.31	6.62
Consistency	**2.07***	**0.08***	9.99	1.69
0.25% Data: Baseline	5.65	0.53	21.76	7.03
0.25% Data: Consistency	**2.41**	**0.10**	**12.26**	**2.69**

Table C.13: **Perceptual results for surface normal estimation on Replica.**

Figure C.6: **Quantitative perceptual results for surface normal estimation on Replica.** Values **bolded and starred*** indicate the best-performing method. Values that are **bolded** but not starred indicate methods that were statistically indistinguishable from the best-performing method (2-sample paired t-test, $\alpha = 0.001$)

C.13 More Qualitative Results

We provide some randomly selected (non-cherry picked) sample images to give a sense of the general performance of the consistency-trained and the baseline networks.

Queries From the Taskonomy Test Set

Fig. C.9 shows the results on various test images sampled from two of the buildings in Taskonomy's test set. The buildings were picked randomly. Each row shows a query and the columns show the predictions, the ground truth, and error maps, for different prediction domains. **This figure is the same as Fig. 4-middle row of the main paper, but for more queries.**

Surface Normals From Middle Domains

Fig. C.10 (without consistency) and Fig. C.11 (with consistency) provide more results in the same format of Figure 3 in the main paper, with addition of error images and ground truth.

Taskonomy Perceptual Results (Extended)

Depth Estimation →	Surface Normal		reShading		Princ. Curvature		Keypts. (2D, SURF)		Keypts (3D, NARF)		Edges (2D, Sobel)		Edges (3D, Occ.)	
Method	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)
Baseline UNet	13.62	4.33	15.68	5.28	7.31	2.19	3.50	0.36	7.56	1.10	12.61	3.49	**1.46***	**0.12**
Constant Pred.	22.37	8.31	27.27	10.72	7.96	2.29	3.85	0.42	7.88	1.56	12.77	4.82	1.97	0.16
GeoNet (Impr.)	13.77	4.41	15.76	5.27	7.52	2.26	3.49	0.36	7.69	1.12	12.67	3.50	**1.46**	**0.12***
GeoNet (Orig.)	15.44	5.94	18.73	7.63	4.03	1.04	**2.66***	**0.27***	6.56	1.09	10.78	**2.76***	2.18	0.22
Multitask	17.18	6.12	19.55	7.14	7.54	2.28	3.39	0.35	9.55	1.53	13.67	3.68	1.91	0.13
Taskonomy	18.82	6.40	20.83	7.17	6.65	1.75	3.44	0.36	9.72	1.48	14.10	4.16	1.94	0.14
Consistency	**9.46***	**2.56***	**12.66***	**4.06***	**3.61***	**0.85***	3.55	0.35	**5.69***	**0.88***	**9.82***	3.00	1.52	0.13

Table C.14: **Perceptual results for depth estimation on Taskonomy.**

reShading →	Surface Normal		Depth Estimation		Princ. Curvature		Keypts. (2D, SURF)		Keypts (3D, NARF)		Edges (2D, Sobel)		Edges (3D, Occ.)	
Method	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)
Baseline UNet	9.58	2.59	3.38	0.25	3.78	0.92	2.98	0.28	6.22	1.06	10.85	3.50	1.53	0.15
Constant Pred.	19.96	7.40	7.14	0.88	3.53	0.91	3.17	0.33	7.48	1.67	12.62	5.53	1.83	0.17
Multitask	9.19	2.48	3.54	0.27	3.56	0.86	2.96	0.27	6.23	1.10	10.75	3.30	1.53	0.15
Taskonomy	11.72	3.45	4.69	0.49	3.54	0.90	3.09	0.31	6.93	1.44	11.19	4.26	1.60	0.16
Consistency	**7.13***	**1.88***	**2.51***	**0.18***	**3.28***	**0.79***	**2.93***	**0.24***	**5.40***	**0.83***	**9.38***	**2.85***	**1.35***	**0.12***

Table C.15: **Perceptual results for reShading on Taskonomy.**

Surface Normal →	Depth Estimation		reShading		Princ. Curvature		Keypts. (2D, SURF)		Keypts (3D, NARF)		Edges (2D, Sobel)		Edges (3D, Occ.)	
Method	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)	L1 (↓)	MSE (↓)
Baseline UNet	8.17	1.21	20.94	8.09	3.41	0.84	3.91	0.40	6.48	1.06	9.98	3.38	1.52	0.16
Cycle Consistency	8.81	1.44	30.33	14.83	3.84	0.92	3.88	0.40	7.53	1.18	10.26	3.11	1.69	0.17
GeoNet (Impr.)	8.18	1.21	20.84	8.05	3.40	0.83	3.91	0.40	6.43	1.06	9.99	3.40	1.52	0.16
GeoNet (Orig.)	7.71	1.46	27.35	15.27	**3.32**	0.81	2.97	0.31	7.64	1.30	**9.09***	**2.51***	1.48	0.15
Multitask	8.78	1.41	27.32	12.82	3.65	0.91	3.94	0.41	7.21	1.12	10.16	3.38	1.64	0.17
Pix2Pix	8.12	1.27	26.23	11.23	3.83	0.93	3.92	0.40	7.80	1.21	10.33	3.39	1.75	0.17
Prcpt. Loss (ImageNet)	6.86	0.88	17.36	5.70	3.36	0.80	3.77	0.38	6.01	0.98	9.63	3.08	1.45	0.14
Prcpt. Loss (Random)	8.59	1.36	23.98	10.26	3.41	0.83	3.91	0.40	6.75	1.10	10.01	3.40	1.56	0.16
Consistency	**4.32***	**0.36***	**12.15***	**3.38***	**3.29***	**0.76***	**2.94***	**0.24***	**5.48***	**0.89***	9.50	2.89	**1.36***	**0.12***
0.25% Data: Baseline (UNet)	8.86	1.42	26.91	12.33	3.78	0.97	3.95	0.41	7.16	1.14	10.31	3.54	1.60	0.16
0.25% Data: Consistency	**5.07**	**0.50**	**15.96**	**5.01**	**3.74**	**0.90**	**3.77**	**0.38**	**6.35**	**1.04**	**9.93**	**2.97**	**1.57**	**0.15**

Table C.16: **Perceptual results for surface normal estimation on Taskonomy.**

Figure C.7: **Extended quantitative perceptual results on the Taskonomy test set.** Values **bolded and starred*** indicate the best-performing method. Values that are **bolded** but not starred indicate methods that were statistically indistinguishable from the best-performing method (2-sample paired t-test, $\alpha = 0.001$). MSE is shown in addition to L1, and results are shown for additional tasks (*Keypoints 2D, Keypoints 3D*, and *Edges 3D*).

C.14 Blind Guess (Statistically Informed Guesses)

As described in the paper, we compared all networks against a "statistically informed guess". This is a guess in that it does not look at the input x when predicting the label y. This is statistically informed in that it is the best-possible such guess given the "guess" constraint. Specifically, we compute the guess, g^* as

$$g^* \triangleq \arg\min_g E_y[|g - y|]$$

.

A visualization of these guesses for the *normals*, *reshading*, and *depth* are provided in Fig. C.8.

C.15 Code, Examples, and Docker

We've open-sourced our code and are providing tools for training and evaluating models using consistency. The repository is available here, and contains (among other things):

- Pretrained models
- Demo code
- Uncertainty energy estimation code
- Training scripts
- Docker and installation instructions

Please see the README in that repository for the most up-to-date information.

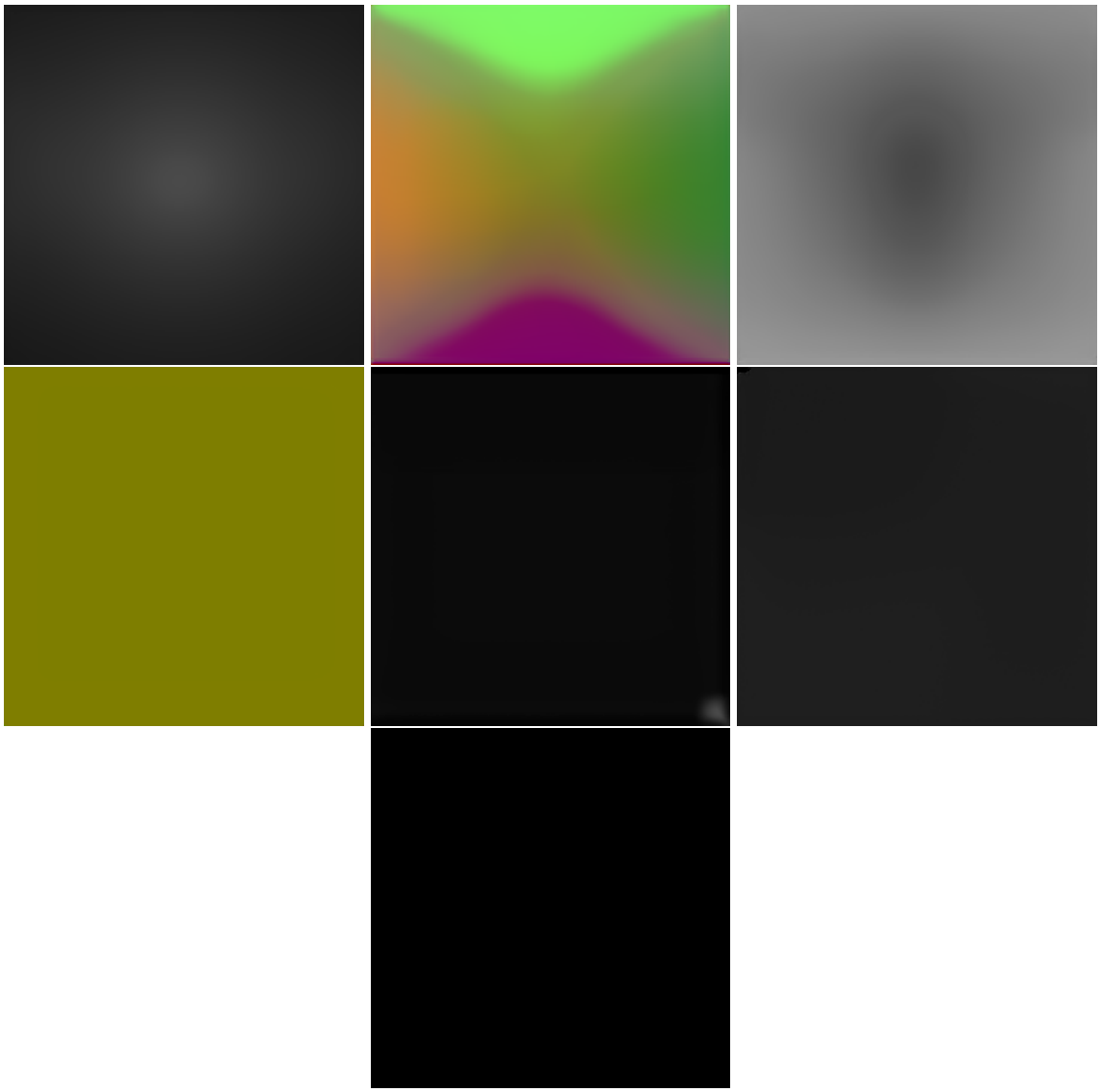


Figure C.8: **Statistically informed guesses ("Blind Guess") on the Taskonomy dataset.** Left to right, by row: Depth, Surface Normals, (re)Shading. Curvature, 2D Keypoints, 3D Keypoints. Occlusion Edges. These images minimize expected L1 error on the training dataset: $\min_g E_y[|g-y|]$.

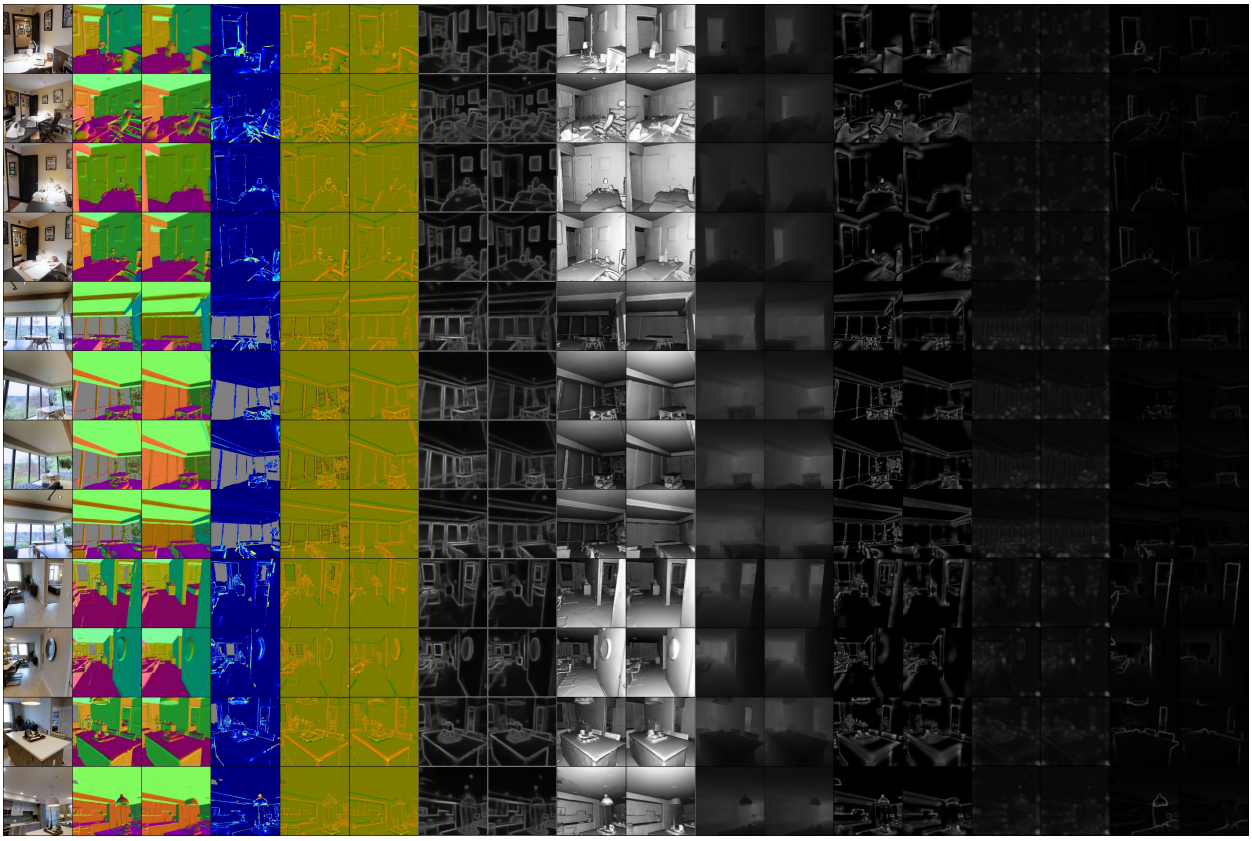


Figure C.9: More qualitative results on test set images of Taskonomy dataset. **This figure is the same as Fig. 4-middle row of the main paper, but for more queries.** The first four columns show: query, surface normal prediction, surface normal ground truth, and error map. Other domains show prediction and ground-truth. The domains from left to right are: surface normals, 3D curvature, Sobel edges, reshading, depth, 3D keypoints, 2D keypoints, occlusion edges. [best seen on screen & zoomed in]

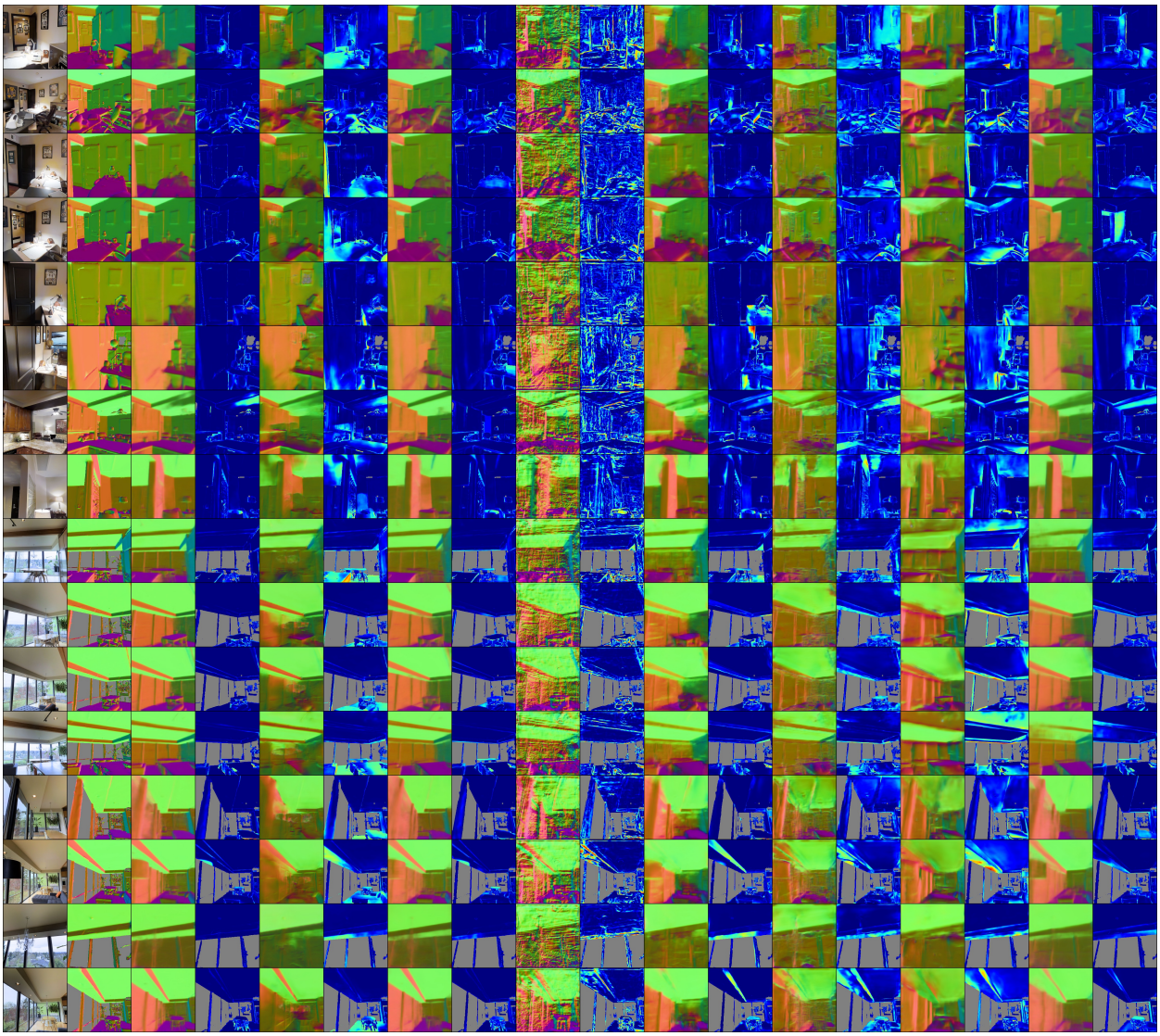


Figure C.10: More results on estimating surface normal out of middle domains without enforcing consistency. **This figure is the same as Fig. 3-upper row of the main paper, but for more queries**. The ground truth and RGB image are shown on the left. The error map of each prediction is show on its right. [best seen on screen & zoomed in]

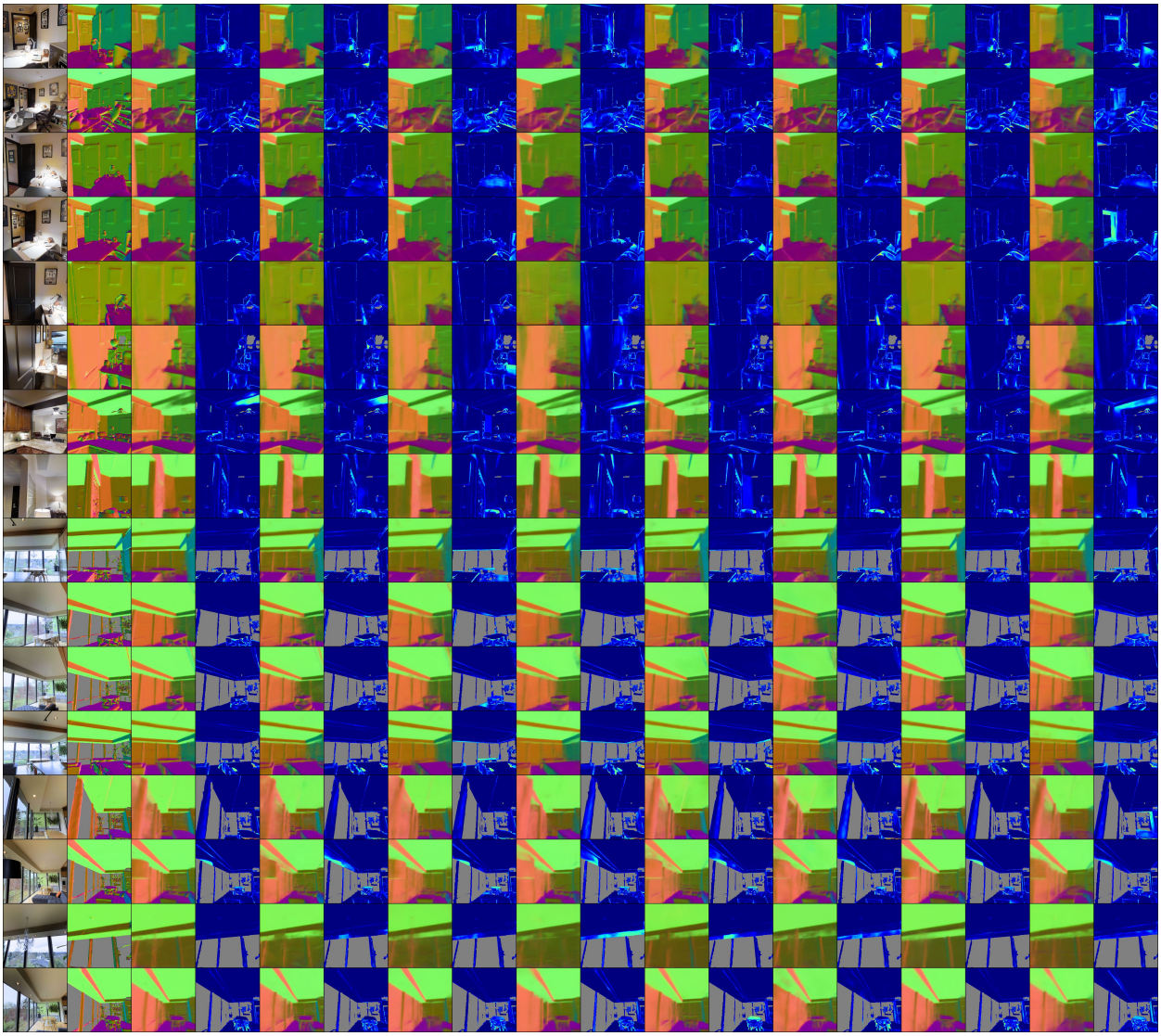


Figure C.11: More results on estimating surface normal out of middle domains after enforcing consistency. **This figure is the same as Fig. 3-lower row of the main paper, but for more queries**. The ground truth and RGB image are shown on the left. The error map of each prediction is show on its right. [best seen on screen & zoomed in]

✗

Appendix D

Chapter 5 Supplementary Material

Omnidata: A Scalable Pipeline for Making Multi-Task Mid-Level Vision Datasets from 3D Scans

The following items are provided in the supplementary material:

I. A live demo to run our networks on your own query images and a dataset design tool to visualize the effects of different dataset design parameters (§ D.1).

II. Code, Docker, runnable examples and a documentation of usage for the annotator, tools, and the starter dataset (§ D.2).

III. Mid-level cues provided by `Omnidata` annotator and their definitions (§ D.3).

IV. Results of surface normal estimation with refocusing augmentation on blurred data (§ D.4).

V. A description of GSO+Replica dataset generation process (§ D.5).

VI. Dataset ablation analysis on surface normal estimation and panoptic segmentation for the starter set (§ D.6).

VII. Visualization and evaluation of the "Blind Guess" (statistically informed guess) for the starter set (§ D.7).

VIII. More qualitative results of surface normal estimation on OASIS dataset (§ D.8).

IX. Full experimental setup for multi-task learning rank reversal experiment (§ D.9).

D.1 Online Demos

The project website includes a live demo that allows to run our pretrained networks on your own uploaded query images. You can visualize the predictions for different tasks and see a comparison of `Omnidata` models to various baselines. The demo page also contains a link to the "demo archive" where you can browse uploads from other users. We also provide a dataset design tool that allows playing with different dataset design choices to visualize their effect on the sampled data.

D.2 Dockerized Pipeline, Tools, and Documentation

We provide a Dockerized Pipeline with all necessary software (Blender [45], MeshLab [42], and other libraries) installed, Pytorch dataloaders for loading the generated data and applying the necessary transforms for reading in each modality to analytic values, a starter dataset along with download scripts and other utilites. We also provide Omnidata Docs which includes a documentation on how to use all the open-sourced material of our paper.

D.3 Mid-level Cues Provided

This section describes the default mid-level cues and additional outputs provided by the `Omnidata` annotator.

2D Cues

***2D Unsupervised Segmentation*:** Gestalt psychology proposes grouping as a primary mechanism through which humans learn to perceive the world as a set of coherent objects [219]. The annotator provides groupings based on normalized cuts [191] of the RGB image into perceptually similar spatially coherent groups.

Texture Edges: offer low-level cues about object boundaries. Classic computer vision pipelines commonly use edges as an intermediate representation in a larger processing pipeline. The annotator provides edges from a Canny [30] edge detector without nonmax suppression.

2D Keypoints: are designed to indicate possibly important pixels and identify them across images. These are frequent in both vision [69] and robotics [139, 138] pipelines. The annotator provides pre-nonmax-suppression SURF intensity maps to identify potentially important regions of the RGB image, and the fragments cue (see below) can be used to link points across images.

Single-View 3D Cues

***Depth: Z-Buffer*:** For each pixel, the metric distance from the point to the camera plane. The most common form of depth in computer vision + robotics.

***Depth: Euclidean*:** For each pixel, the metric distance from the point to the camera's optical center. This can be used (e.g.) for adding lens blur (Sec. 6.2 of the main paper).

***Surface Normals*:** Crucial for computer vision and robotics tasks (e.g. for computing lighting, grasp estimation, etc.): the tangent vector relative to the camera of the corresponding point on the mesh.

Principal Curvature: For each pixel, the principal curvatures κ_1 and κ_2, which are also sufficient for computing Gaussian $(\kappa_1 \cdot \kappa_2)$ and mean $((\kappa_1 + \kappa_2)/2)$ curvature. These quantities are invariant under rigid transformations, and curvature is known to be important in primate visual processing [234].

***Occlusion Edges*:** indicate boundaries where one pixel occludes something behind it. While 2D edges respond to to changes in texture, 3D edge features depend only on 3D geometry and are invariant to color and lighting.

***(re)Shading*:** One cue to infer scene geometry from an RGB image is "shape from shading" [20] via the intrinsic image decomposition $I = A \cdot S$ into an albedo A and a shading function S parameterized by lighting and depth. The decomposition is thought to be useful for human visual perception [1]. We define a (re)shading cue for S as follows: Given an RGB image, the label is the shading function S that results from having a single point light at the camera origin, and S is multiplied by a constant fixed albedo A.

***3D Keypoints*:** 3D keypoints, like 2D, are designed to indicate possibly important points and link them across viewpoints. Unlike 2D keypoints, 3D keypoints are often designed to be invariant to informative (but possibly distracting) cues such as texture [245, 200, 141, 235, 114]. Based on its specificity and robustness, we use the pre-nonmax-suppressed output of [200] for this cue.

***2.5D Unsupervised Segmentation*:** uses the same graphcut algorithm as 2D, but the labels are computed jointly from the RGB, depth image, and surface normals. Thus the 2.5D segmentation cue incorporates information about scene geometry that is not present in the RGB image but readily inferred by humans.

***Manhattan Vanishing Points*:** Vanishing points offer useful information about the scene geometry [142, 118], particularly a "Manhattan world" [48, 240, 23] with three dominant vanishing points (X, Y, and Z axis). We provide the X, Y, and Z Manhattan vanishing points (Gaussian sphere format).

***Camera Intrinsics*:** Deep networks are excessively sensitive to changes in camera intrinsics such as field-of-view. We provide camera intrinsics for each image.

Multi-View 3D Cues

***Camera Extrinsics*:** provides camera RT matrices for each image.

***Point Matching*:** indicates which other preselected points are present in this view. Useful for point matching tasks such as [69].

Fragments (Optical Flow): Each space_point_view image contains an image whose pixel values encode the corresponding mesh face, and these values are consistent across images in the space and can be decoded to approximate global 3D coordinates or used for optical flow. This would be akin to perfect feature descriptors for either 2D or 3D keypoints.

Semantic Cues

If the dataset supports, the annotator can provide cues for the following:

***Class Presence*:** labels provide a present/not present indicator used for image classification.

***Instances*:** identify the instance identity of each pixel. Regardless of class, this gives an object-centric grouping.

***Semantic Class*:** the semantic category for each pixel.

***Panoptic Segmentation*[1]:** combination of semantic segmentation and instance identification [111].

Additional Information

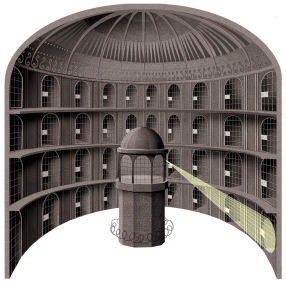
***RGB*:** RGB images can be real image scans if provided, or they can be rendered from the textured mesh.

Masks that indicate whether the pixel corresponds to an area missing from the mesh.

D.4 Surface Normal Estimation with Refocusing Augmentation

As described in Sec. 6.2 of the main paper, the mid-level cues can be used as data augmentations in addition to training targets. While defocus cue can be useful in depth estimation [79, 32, 197], we explore it as refocusing augmentation on our dataset, which is possible due to availability of camera parameters and euclidean depth. We provide quantitative and qualitative surface normal estimation results for training with this augmentation. Tab. D.1 compares 2 models trained with and without this augmentation evaluated on both refocused and blurred test data from our starter set. We use Gaussian blur with kernel size 3 and sigma uniformly chosen in the range (0.1, 2). As shown by the results, the model trained with refocusing augmentation shows much better performance on the blurred data. The gap is clearer as shown by images in Fig. D.1. The figure shows that the baseline model would easily fail with a small amount of blur present in the input, and the refocusing augmentation has a substantial effect in increasing the robustness to these blur effects. We repeat the experiment with different levels of blur in the input using different kernel sizes (3, 5, 7, 9). Fig. D.2 shows the performance of the models for each amount of blur. As the plots show, the model trained with augmentation shows good performance even for high levels of blur, while the accuracy drops significantly in the baseline model as the blur increases.

[1] A concrete name like "Per-Pixel Category and Instance Segmentation" would be clearer and less provocative than "Panoptic Segmentation". The *Panopticon* (from 18th-century philosopher Jeremy Bentham) was conceived as the ideal prison; an institutional system of control-by-surveillance whereby a centralized security guardhouse can observe all prisoners in one view, while subjects are unable to tell whether they are being watched. Though instantiated as a building, Bentham intended it as a method for any institution, with the threat of observation, to force compliance and docility. Expanded and popularized by Foucault [64] in the 20th century, *Panopticism* remains influential among disciplines across the humanities and social sciences. The *panopticon* is also well-known in popular culture; as "Big Brother" in the surveillance narrative *Nineteen Eighty-Four*, for example, and by name in the cover story of the most recent edition of *The Economist* ("The People's Panopticon," *7 Aug. 2021 edition*).



An illustrated panopticon. Two of the first 10 images returned for the Google query *"modern panopticon"* make reference to Facebook, including one that is simply the above image with the Facebook logo superimposed.

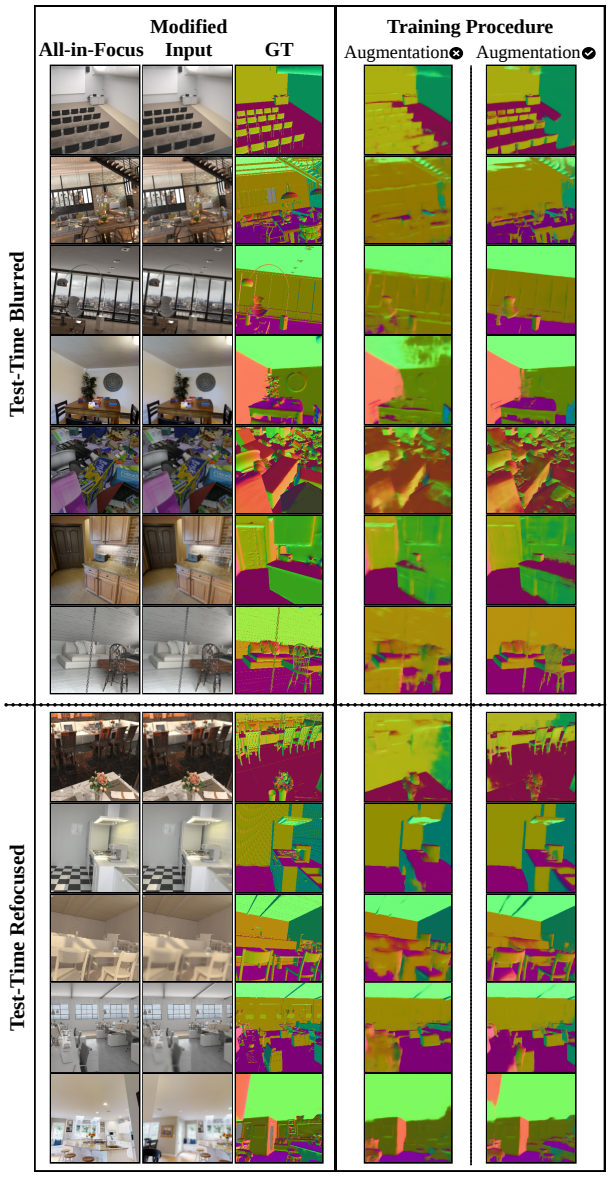


Figure D.1: **Qualitative results for refocusing augmentation.** The results compare the models trained with and without refocusing augmentation on both refocused and blurred data from the test splits of the starter set. Same parameters as training are used for refocusing the test data. We also use Gaussian blur with kernel size 3 for blurring the input. Clearly the model trained with the augmentation shows much more robustness to blur effects while the baseline model easily fails with a small amount of blur [best viewed zoomed in].

Refocusing Augmentation	Test Data	Error (↓) L1	Error (↓) MSE	Angular Error° (↓) Mean	Angular Error° (↓) Median	% Within $t°$ (↑) 11.25°	% Within $t°$ (↑) 22.5°	% Within $t°$ (↑) 30°
✗	Blurred	7.54	2.04	16.86	8	61.91	75.73	81.17
✓		**6.44**	**1.61**	**14.37**	**6.40**	**66.20**	**79.31**	**84.53**
✗	Refocused	6.45	1.63	14.42	6.52	66.36	79.55	84.67
✓		**6.14**	**1.48**	**13.685**	**6.108**	**67.23**	**80.46**	**85.64**

Table D.1: **Surface normal estimation with refocusing augmentation.** The models are evaluated on blurred and refocused test split of the starter set. Gaussian blur with kernel size 3 is used for blurring the input. As the results show, refocusing augmentation improves the performance of the model on blurred data.

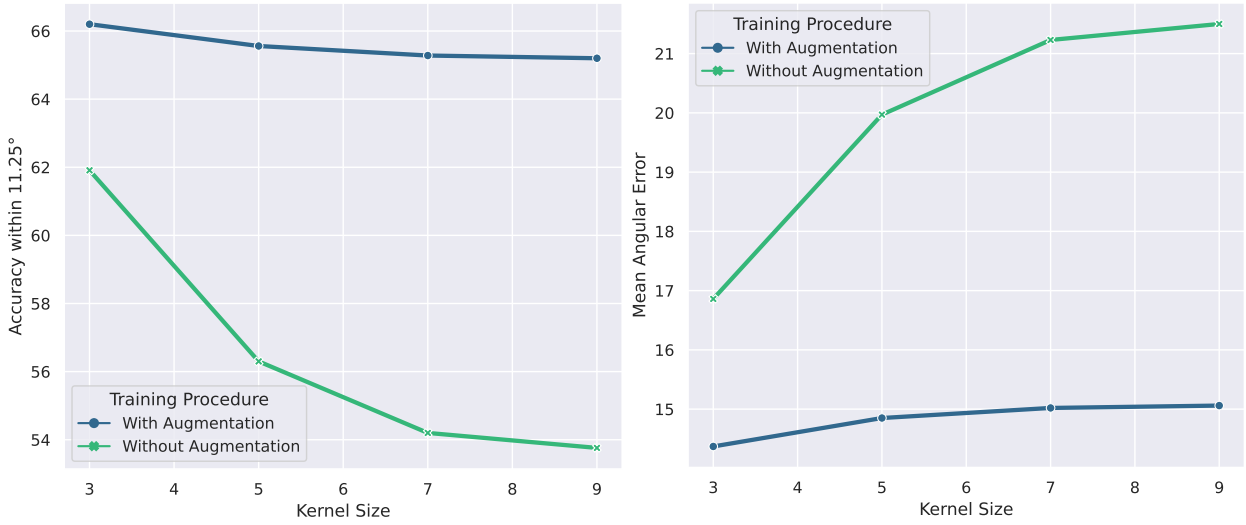


Figure D.2: **Performance of the 2 training procedures (w/ and w/o refocusing augmentation) for different amounts of blur in test data.** We use Gaussian blur with kernel sizes 3, 5, 7, and 9, to produce different amounts of blur in the input. The plots show the "Accuracy within 11.25°" (left) and "Mean Angular Error" (right) for the 2 models for each kernel size. It is shown that the performance of the baseline model significantly drops with increasing the amount of blur while the model trained with augmentation shows much more robustness.

D.5 GSO+Replica Dataset Generation Process

We scatter Google Scanned Objects [95] around Replica [202] buildings to create object-centric views. Habitat [137] environment is used to generate physically plausible scenes. The dataset is provided in 3 different object densities for each space (3, 6, 15 objects per square meter which we refer to as low, medium, high density). Objects are randomly sampled from 1032 objects provided in Google Scanned Objects, and they are scattered uniformly across the building according to the density. To create object-centric views, thousands of cameras are generated in each space using

Figure D.3: **Sample images from GSO + Replica dataset.** Images shown are from the 3 different object densities (3, 6, 15 objects/m^2) included in GSO+Replica dataset.

Poisson Disc Sampling. Points of interest are only sampled from the objects rather than the whole mesh. For each point-of-interest, a subset of cameras with an unobstructed line-of-sight of the point are selected. Cameras are filtered according to an additional constraint so that the point-camera distance is between 0.2 and 1 meter to make sure we have an object-centric view. The views are saved for each camera-point combination in which the camera is fixated on the point-of-interest. Examples of images from each object density are shown in Fig. D.3.

D.6 Dataset Ablation Analysis of the Starter Set

To assess the contribution of each single dataset in our starter set, we list the zero-shot transfer performance to OASIS [38] and COCO [130] for models trained on each single dataset of the starter set, and some combinations of them.

The results listed in Tab. D.2 and D.3 provide an understanding of the impact of each dataset component in our starter set. Models trained on only scene-level data such as Taskonomy [238], Hypersim [176], or Replica [202] result on poor performance on objects, while the model trained on GSO+Replica will have an object-centric bias with poor performance on backgrounds and scenes. We provide a starter dataset with both scene- and object-centric views which, as shown by the results, is necessary for final best performance and generalization to in-the-wild data. Furthermore, including all datasets is necessary since the diversity present in the whole starter dataset will further improve the generalization.

Training Data					Anglular Error° (↓)		% Within $t°$ (↑)			Relative Normal (↑)	
Taskonomy	Replica	Hypersim	Replica+GSO	BlendedMVG	Mean	Median	11.25°	22.5°	30°	AUC_o	AUC_p
✓					29.68	21.78	25.73	51.29	62.66	0.6220	0.6163
	✓				33.06	26.03	19.03	43.60	56.31	0.5711	0.6099
		✓			29.94	22.81	21.64	49.36	62.28	0.6375	0.6311
			✓		33.22	26.46	15.81	41.97	56.21	0.5669	0.5893
				✓	29.77	23.23	23.47	48.64	61.23	0.5661	0.6033
✓	✓				28.62	21.59	24.01	51.85	64.40	0.6260	0.6248
✓	✓	✓			28.61	21.55	23.81	51.94	64.78	0.6614	0.6566
✓	✓	✓	✓		27.73	20.43	25.72	54.06	66.51	**0.6686**	0.6596
✓	✓	✓	✓	✓	**26.34**	**19.39**	**28.66**	**56.37**	**68.43**	0.6572	**0.6832**

Table D.2: **Zero-shot transfer performance on OASIS dataset.** Models are evaluated on val split of OASIS dataset. The results show the impact of each single dataset in the starter set on the performance of surface normal estimation and generalization to in-the-wild data.

Training Data			Taskonomy			Replica						Hypersim					
Taskonomy	Replica	Hypersim	PQ_{th}(↑)	SQ_{th}(↑)	RQ_{th}(↑)	PQ_{th}(↑)	SQ_{th}(↑)	RQ_{th}(↑)	PQ_{st}(↑)	SQ_{st}(↑)	RQ_{st}(↑)	PQ_{th}(↑)	SQ_{th}(↑)	RQ_{th}(↑)	PQ_{st}(↑)	SQ_{st}(↑)	RQ_{st}(↑)
✓			8.39	38.88	9.54	6.99	37.67	9.32	-	-	-	21.76	68.46	26.94	-	-	-
	✓		1.01	17.39	1.31	28.82	56.45	36.01	**55.12**	69.72	**65.08**	2.89	35.68	3.90	6.11	28.95	8.99
		✓	9.35	54.25	**11.90**	14.67	55.56	18.65	13.48	29.73	18.57	23.90	65.73	29.94	26.87	52.29	35.14
✓	✓		**10.27**	45.85	11.82	28.66	57.87	35.98	54.17	**70.18**	63.76	8.90	38.29	11.04	10.25	27.02	14.61
✓		✓	8.70	40.67	10.13	9.44	50.56	12.07	15.37	33.35	20.61	26.28	67.64	32.43	30.72	54.03	38.97
	✓	✓	9.09	**61.48**	11.69	**44.07**	**75.94**	**53.88**	48.99	64.00	58.10	24.67	68.51	30.46	19.04	37.52	24.68
✓	✓	✓	9.14	41.95	10.29	30.14	57.92	37.50	52.35	64.87	61.67	**27.79**	**68.86**	**34.28**	**32.53**	**54.77**	**40.84**

Table D.3: **Ablation of training datasets for panoptic segmentation.** Transfers to and from Taskonomy only evaluate *things* labels, as Taskonomy does not feature any stuff labels.

D.7 Blind Guesses (Statistically Informed Guesses) for the Starter Set

Similar to [236], we compute the blind guesses (query-agnostic statistically informed guess) from the starter set for each domain. We evaluate the blind guess for surface normals on OASIS data, and the test split of the starter set in Tab. D.4. The reported results will provide an estimation of the lower bound performance for these datasets. We also compare our blind guesses to the ones computed only from the Taskonomy dataset. A visualization of these guesses for surface normals and reshading are provided in Fig. D.4. Comparing the blind guesses for the 2 datasets demonstrates that there is less bias present in the starter set compared to the Taskonomy alone, such as the ceiling bias present in the top part of the image for surface normal blind guess of Taskonomy which is not the case in our starter set. Better performance of the starter set blind guess (compared to the Taskonomy alone) on OASIS data, as shown in Tab. D.4, will further prove the point.

D.8 Surface Normal Estimation on OASIS Dataset

In this section, we include additional qualitative results from our surface normal estimation experiments on OASIS [38]. Fig. D.5 qualitatively compares the models trained on Full Taskonomy and the starter set on some sample images from the val split of OASIS. As shown by the figure, the model trained on Full Taskonomy has poor performance on objects and largely misses the details as opposed to the model trained on the starter set.

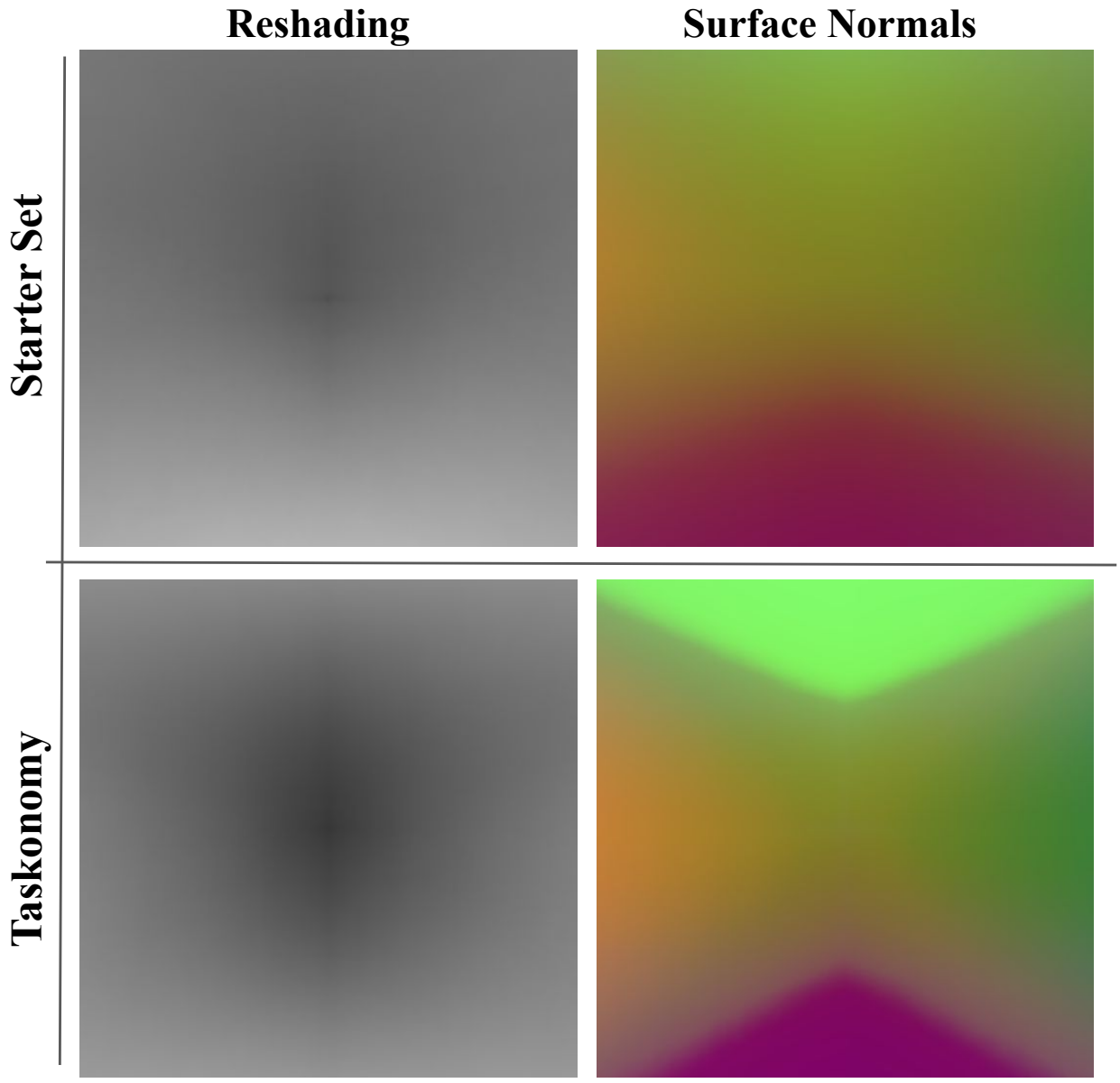


Figure D.4: **Statistically informed guesses ("Blind Guess") on the Starter Set.** Blind guesses computed from the starter set and Taskonomy alone are shown for 2 domains. Comparing the surface normal blind guess for the 2 datasets will show that there is less bias present in our starter set comparing to Taskonomy alone (the ceiling bias which is only present in Taskonomy blind guess).

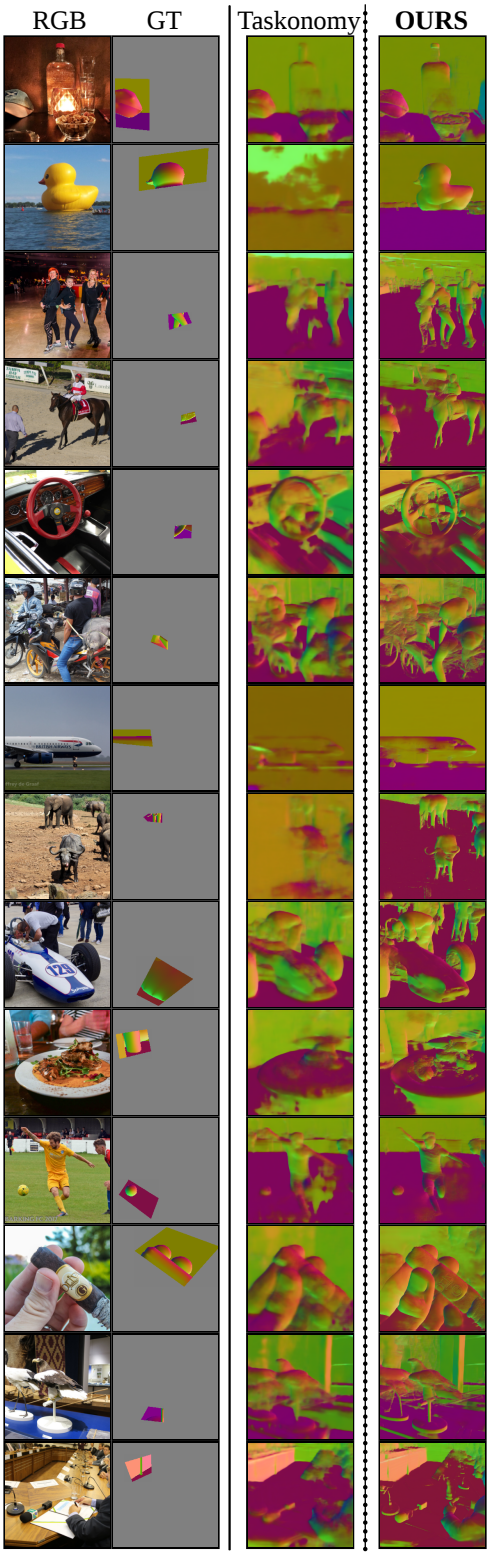


Figure D.5: **Qualitative results on OASIS data.** The 2 models are trained on Full Taskonomy and the starter set. The Taskonomy model has poor performance on objects and largely misses the details.

Test Data	Blind Guess	Angular Error° (↓) Mean	Median	% Within $t°$ (↑) 11.25°	22.5°	30°	Relative Normal (↑) AUC_o	AUC_p
OASIS	Starter Set	**35.28**	**30.64**	**14.48**	**36.7**	**49.03**	**0.5352**	0.4302
	Taskonomy	41.73	35.80	10.28	29.00	41.38	0.5282	**0.4404**
Starter Set	Starter Set	**43.72**	**43.04**	7.41	21.6	**32.17**	-	-
	Taskonomy	44.88	44.66	**8.87**	**22.57**	31.97	-	-

Table D.4: **Blind guess evaluation on OASIS and starter set.** The blind guesses computed from our starter set and Taskonomy alone are evaluated on val split of OASIS and test split of the starter set. The results will provide a lower bound for performance on these benchmarks.

D.9 Multi-Task Learning Rank Reversal Experimental Setup

In this section, we explain the experimental setup for the multi-task learning rank reversal experiment provided in the section 5.3 of the main paper. Similar to [212], we use a simple shared-encoder MTL model, a single task baseline, as well as 2 other common MTL approaches (MTAN [131] and Cross-stitch [145]) for our experiment. Each encoder is a ResNet-50 model with dilated convolutions and pre-trained on ImageNet [50]. We use Deeplab [35] head for the task specific decoders. Each multi-task model is trained on the 4 following tasks: semantic segmentation, 3D keypoints, depth z-buffer, and occlusion edges. We use medium Taskonomy, Replica, and Hypersim as the training data, and evaluate the models performance on semantic segmentation and 3D keypoints on tiny Taskonomy test set. Table 4 of the main paper provides the results for this experiment.