Dense Visual Learning for Robot Manipulation

by

Peter R. Florence

A.B., Princeton University (2012)
M.Phil., Cambridge University (2013)
S.M., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Author . . . . . .
Department of Electrical Engineering and Computer Science
September 30, 2019

Certified by . . . . . .
Russ Tedrake
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Dense Visual Learning for Robot Manipulation

by

Peter R. Florence

Submitted to the Department of Electrical Engineering and Computer Science
on September 30, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We would like to have highly useful robots which can richly perceive their world, semantically distinguish its fine details, and physically interact with it sufficiently for useful robotic manipulation. This is hard to achieve with previous methods: prior work has not equipped robots with the scalable ability to understand the dense visual state of their varied environments. The limitations have both been in the state representations used, and how to acquire them without significant human labeling effort. In this thesis we present work that leverages self-supervision, particularly via a mix of geometrical computer vision, deep visual learning, and robotic systems, to scalably produce dense visual inferences of the world state. These methods either enable robots to teach themselves dense visual models without human supervision, or they act as a large multiplying factor on the value of information provided by humans. Specifically, we develop a pipeline for providing ground truth labels of visual data in cluttered and multi-object scenes, we introduce the novel application of dense visual object descriptors to robotic manipulation and provide a fully robot-supervised pipeline to acquire them, and we leverage this dense visual understanding to efficiently learn new manipulation skills through imitation. With real robot hardware we demonstrate contact-rich tasks manipulating household objects, including generalizing across a class of objects, manipulating deformable objects, and manipulating a textureless symmetrical object, all with closed-loop, real-time vision-based manipulation policies.

Thesis Supervisor: Russ Tedrake
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

There are many people I'd like to thank for making my time during graduate school a fantastic journey.

First and foremost I would like to thank my advisor, Russ. Russ has a remarkable mix of abilities - combining deeply rigorous technical skills with inspiration and vision for the future of robotics. I'm also immensely thankful to Russ for taking me, a kid who had only previously done research in Chemistry and Physics, into our incredible lab to start working on robotics.

I'd also like to thank the rest of my committee for their time, guidance, and support throughout this process. Thank you Leslie, Nick, and Dieter.

There have been many other professors at MIT that have been generous with their time and encouragement. In particular I'd like to thank Tomas Lozano-Perez, Alberto Rodriguez, Bill Freeman, and Antonio Torralba.

I have had a world-class set of labmates to work with and learn from. I am indefinitely grateful to Andy Barry and John Carter for teaching me so many things during the start of my PhD. In the second half of my PhD I've been lucky to work closely with Lucas Manuelli, Pat Marion, and Wei Gao. Extra thanks to Lucas for being a ski-buddy, friend, and research partner in my later work. Additionally, a broad cast of other characters have been great to work with and have around the lab for inspiring discussions: Greg Izatt, Matthew O'Kelly, Benoit Landry, Ani Majumdar, Vincent Tjeng, Robin Deits, Twan Koolen, Aykut Satici, Hongkai Dai, Michael Posa, Frank Permenter, Shen Shen, Geronimo Mirano, Tao Pang, and YunZhu Li. Steve Proulx provided fantastic hardware engineering with various robot projects. Also thanks to Mieke Moran and Gretchen Jones for being awesome at supporting our lab.

I've also learned from and enjoyed working with great folks at organizations outside of our lab. My internship at Facebook Reality Labs was hugely influential in developing my appreciation of perception. I'd like to thank: Steven Lovegrove, Richard Newcombe, Julian Straub, and Jeong Joon Park. Additionally, it was an awesome and fun learning experience to work on DARPA FLA with people from both MIT and Draper Laboratory:

(from MIT) John Carter, Jake Ware, Brett Lopez, Nick Greene, Kris Frey, and Katherine Liu, (from Draper) Ted Steiner, Steve Paschall, Julius Rose, Rob Truax, Scott Rasmussen, Chris Wardman, and Tye Brady. It's also been great trading ideas with the team at Toyota Robotics Institute, in particular: Duy Nguyen, Alex Alspach, Eric Cousineau, Sammy Creasey, Siyuan Feng, and Allison Fastman.

And of course I would like to thank my family! Mom, Dad, and TJ, thank you for everything. Susanne, you are very wonderful and thank you for all your support. Varun – you were my MIT family for a big chunk there! Special thank you also goes to: Murphy, Huxley, Sidney, Maisy.

Funding Acknowledgement

This work was supported by Lockheed Martin Corporation, Award No. RPP2016-002, Draper Laboratory, Incorporated; Award No. SC001-0000001002, and Amazon, Award No. 2D-01029900.

Contents

Chapter 1

Introduction

A nice motivational insight for this thesis work comes from Geoff Hinton: "The brain has about 10^{14} synapses and we only live for about 10^9 seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^5 dimensions of constraint per second."[1] We view this comment as particularly relevant to the challenges of robotic manipulation and computer vision "in the wild". We would like robots, for example, to be able to inspect arbitrary scenes of objects and complete arbitrary tasks with them. Yet from the firehose of raw perception data, for example 2.8×10^7 dimensions per second for just a medium-resolution camera[2], it is challenging to provide enough constraints with just manual specification or sparse rewards.

In this thesis our primary interest is in how to scalably enable robots to acquire perceptual knowledge useful for robotic manipulation. This leads us to naturally ask: *what type of perceptual knowledge is useful, and how can it scalably be acquired?* In this thesis we address both of these, questioning both the representations that are used, and how to acquire them. We aim to enable robots to efficiently supervise their perceptual inputs entirely by themselves, or to leverage self-supervision in order to make the most out of minimal human input. The benefits of this perceptual self-supervision may be interpreted in various ways: on the one hand, it may provide the ability to lift raw data into regimes in which

[1] November 7, 2014, Reddit A.M.A., https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/

[2] 640 width pixels *480 height pixels* 3 color channels * 30 Hz

explicit geometrical planning can thrive, and alternatively, perceptual self-supervision can be thought of as a type of multi-task learning providing a sizable source of otherwise inaccessible supervision.

To address our goal, we leverage the structure of robotic 3D perception data and a variety of tools including multi-view geometry, new object representations, deep visual models, general-purpose grasping pipelines, and robot learning from demonstration. One of the main contributions of this thesis is in introducing dense visual description of objects as a representation useful for manipulation (Dense Object Nets [1]), and in showing that this representation can be fully autonomously acquired without human input. Another area of contribution is a method to provide ground-truth labels of 6DOF object poses in cluttered scenes (LabelFusion [2]), which does require human input but does so in a highly efficient manner due to dense 3D-based labeling. Finally, we demonstrate using self-supervised correspondence learning to efficiently learn visuomotor policies from demonstrations of manipulation tasks.

Problem Statement

This thesis places primary importance on robotic visual learning being both *dense* and *scalable*. Perhaps the best way to appreciate our definition of *dense* is by analogy and contrast (Figure 1-1) with a different domain area which was a focus of the first half of my PhD: obstacle avoidance for autonomous vehicles in static environments. For this task, perceiving the *dense* state of the environment is not strictly necessary. In obstacle avoidance, the primary goal is to not touch anything, and it is sufficient to have conservative approximations of free space [3], without ever perceiving, for example, large portions of fine geometric detail, or distinguishing semantics.

For the applications of focus for this thesis, however, the detail at dense resolution is critical. In this thesis we take a broad view of the definition of *dense* to refer to all types of understanding at approximately the millimeter-scale of what is relevant to the physics of robot fingers. This includes both the geometry of the world's objects, but also *what this geometry is*. For some tasks, robots may need to understand the explicit semantics of objects in cluttered environments ("this is a drill"), and for other tasks we may need robots

Figure 1-1: The lack of necessity for dense visual understanding in some tasks, and the need in others. The comparison between (a) and (b) is adapted from Bachrach [4] who nicely observes: "Two corridors with very different geometry, but similar feasible trajectories. For planning purposes, there is no reason to expend effort modeling all of the intricacies of the tree trunks and branches." In contrast, the task of cleaning up a kitchen requires considerable understanding of the difference in intricacies between (c) and (d). Image sources in appendix.

to be able to densely correspond the fine structure of objects ("grab *this* part of the object and put it on top of *that* part of the other object"). In some of our work, *dense* is used as it is commonly in the computer vision literature for 2D images, which means "every pixel", but the term also applies to 3D representations, as in *dense* 3D reconstructions.

In addition to robotic manipulation, there are many other applications of computer vision "in the wild" for which this type of dense visual understanding is useful. Automated computer vision systems such as those that enable automatic-checkout grocery stores (i.e., Amazon Go) require semantic understanding of considerable density. In augmented reality, dense understanding is critical for virtual-physical interactions (i.e., a virtual person sitting down in a real-world chair), and occlusions of virtual agents. For autonomous navigation, beyond just the static obstacle avoidance problem, agents may benefit from densely understanding the environment in order to leverage priors on, for example, its layout beyond the sensor horizon [5]. Additionally, in dynamic environments for autonomous cars it's highly useful to identify all bicyclists, pedestrians, and cars with their 3D locations and semantic

classes in order to increase the prediction accuracy of their behavior.[3]

The second primary robotic vision property of interest in this thesis is *scalability*, which concerns whether the desired perception knowledge is actually attainable at large scale. In other words, "how do we actually get the data"? The key underlying factor determining scalability is how much cost (human time, computation time, hardware cost, etc.) is required in order to provide the requisite data. It is probably unattainable, for example, to have precise 3D mesh models of every object a general-purpose manipulation robot would interact with, particularly in a world full of shape variation, deformable objects, granular media, and fluids.

In a recent review, Mason [6] gives a careful definition of manipulation as: "manipulation refers to an agent's control of its environment through selective contact". As depicted in Figure 1-2, a core current challenge in robotic manipulation is in enabling manipulation of arbitrary scenes of objects, rather than controlled environments with pre-known object configurations. Accordingly, the primary problem this thesis addresses is how to enable robots to perceptually understand arbitrary scenes of objects in order to inform the choice of selective contact.

1.1 Contributions

This thesis develops novel contributions in robotic dense visual learning, including in introducing dense visual object representations to robotic manipulation, in self-supervised systems to scalably acquire data for dense visual understanding, and in leveraging self-supervised vision to increase the ability to accomplish complex tasks demonstrated by humans.

In particular, in Chapter 2 we develop a pipeline to rapidly generate ground truth labels for RGBD data of cluttered scenes, which researchers can use to build their own datasets that include pixelwise labels and 6DOF poses of multi-object data with occlusions, with the only hardware requirement being the RGBD sensor itself. To demonstrate the useful-

[3]Note: in this application, perhaps centimeter-scale or pixel-scale is sufficient, but still, it is a fine-scale visual understanding, and the world scale relevant to fast-moving cars is much larger than the world scale immediately relevant to a manipulation robot.

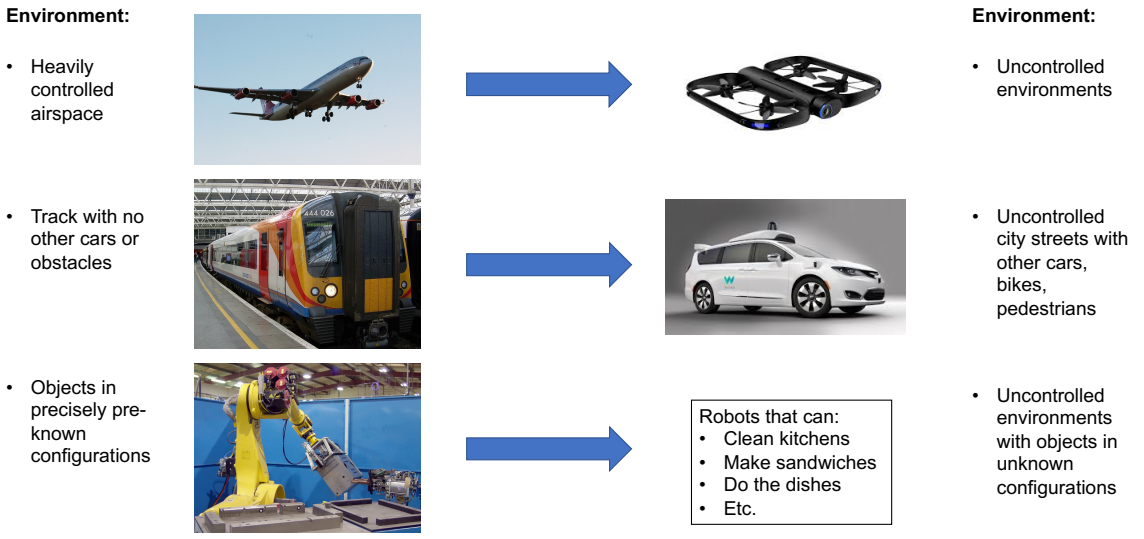


Figure 1-2: A primary challenge for robotic manipulation is in moving from controlled environments to increasingly uncontrolled environments, similar to the comparison of moving from commercial planes and subways to modern autonomous drones and cars.

ness of this pipeline we generated the largest known RGBD dataset with object-pose labels (352,000 labeled images, 1,000,000+ object instances) in only a few days. In Chapter 3 we introduce dense descriptors as a representation useful for robotic manipulation. We also show that self-supervised dense visual descriptor learning can be applied to a wide variety of potentially non-rigid objects and classes (47 objects so far, including 3 distinct classes), can be learned quickly (approximately 20 minutes), and can enable new manipulation tasks. In example tasks we grasp specific points on objects across potentially deformed configurations, do so with object instance-specificity in clutter, or transfer specific grasps across objects in a class. We also contribute novel techniques to enable multi-object distinct dense descriptors, and show that by modifying the loss function and sampling procedure, we can either acquire descriptors which generalize across classes of objects, or descriptors that are distinct for each object instance. In Chapter 4, we develop a new formulation for improved correspondence localization. Finally, in Chapter 5, we leverage self-supervised correspondence learning to improve visual imitation learning, and improve the sample efficiency and precision with which robots can imitate manipulation skills.

AI Level	Definition
$\emptyset$	below the level of an average human toddler
I	at or above the level of the average human toddler
II	at or above the level of the average human adult
III	above the world's best human expert

Figure 1-3: A simple scale for categorizing the level of artificial intelligence (AI level) for a given task. The scale compares skill on a task to three levels of human skill: average toddler, average adult, and world expert human.

1.2 Motivation

We would like to have friendly robots be able to do labor-intensive things for us like clean the kitchen, make us a sandwich, do the dishes, or help the elderly with daily manipulation tasks. It should come as no surprise to anyone, however, that we don't have C-3PO or Baymax robots walking around in anybody's homes that are generally capable of these types of tasks.

Remarkable progress, however, has been made in the last few years in related domains of other artificial agents: we now have autonomous cars that are capable of navigating some degree of regular traffic in mapped areas, autonomous drones that are capable of navigating in fully unstructured environments and are available for purchase by consumers, humanoid robots that can perform backflips, and artificial agents that are capable of beating the world's best human player in the game of Go. One qualitative way (Table 1-3) to analyze the intelligence level for these different AI systems is by a simple metric: how do these systems compare in their capabilities to the skill levels of average human toddlers, average human adults, and the world's best human expert? These artificial agents exhibit high levels of mastery, in large parts comparable or better than human adults at these respective tasks (Table 1-4).

Why, then, are artificial agents still at approximately toddler level or below in visual scene understanding and robot manipulation? This thesis aims to increase our understanding of the inherent challenges and provide tools that increase the skill level of autonomous systems at these tasks.

A set of primary challenges though with the route of bringing perception information into the right state for dense visual understanding and robot manipulation is: *what even*

Task	AI Level	Metric	Explanation
Repeatedly welding a known pattern	III	Speed and accuracy	Robot arms have been capable of superhuman repeated precise patterns since at least the 1980s.
Playing the game of Go	III	Rules of Go	AlphaGo beat Lee Sedol in 2016.
Piloting a drone in unstructured environments	II or III	Safe (obstacle-free) and fast navigation	A commercially-available Skydio drone is easily superior to the average human at navigating a drone. The world's best FPV (first-person-view) pilots probably still win in gracefulness and speed, even if they are riskier.
Piloting a car in urban environments	I or II	Safe navigation on streets	There is no doubt that a modern autonomous car is safer than a human toddler, so above level $\emptyset$. For large parts of the autonomous driving problem, arguably autonomous cars are above average human level (level II), although some challenges remain (highly robust perception, other driver behavior prediction, etc.).
Bipedal walking on unstructured terrain	$\emptyset$ or I or II	Ability to not fall and gracefully navigate difficult terrain	Modern Boston Dynamics robots can do some highly dynamics skills more gracefully than average humans (like backflips), and have been shown to handle a wide variety of terrains, although once children learn to walk they are formidable.
Visual scene understanding	$\emptyset$ or I	Ability to answer questions asked by a human of a scene	While image classification is "super-human" on a static dataset like ImageNet, we do not have computer vision systems capable of answering arbitrary questions like, "why is the human standing under the umbrella?", so below level II.
Manipulating arbitrary objects in unstructured environments	$\emptyset$ or I	Ability to be presented a new object and accomplish an arbitrary manipulation task	The average adult can easily tie a shoe, open a book, hang a mug on a rack, etc, so below level II.

Figure 1-4: Respective levels of intelligence of existing AI systems as measured by the simple scale (Table 1-3).

is the right state of the world, and how should it be acquired? This question may inspire a variety of responses from the research community. An easily stated and often preferred answer has been to have the 6DOF pose of each object instance in the world (with this pose referring to a pre-known 3D mesh model), but this raises scalability concerns, and doesn't apply to deformable objects. Additionally, once we involve occlusions and deformable objects, we must accept the world may not be fully observable. Note that a partial view of a rigid object may be enough to infer its state, but for deformable objects, often "self-occlusions" (one part of the object occluding another part) prevent full observability, since for example we can not see the bottom of a clumped-up towel. Another challenge is that many details of the environment are not relevant to tasks, but *some* of the fine detail matters – for example, to zip open a backpack, the small zipper handle of the backpack must be manipulated. To compare again with research challenges for other robotics platforms, these types of challenges (partial observability, and a small proportion of state that is task-critical) are prevalent as well in other primary unsolved challenges – for example, for autonomous vehicles to ideally model the behavior of other cars on the road, the mental state of those drivers is both very task-relevant, and not fully observable. For the three robot domains of autonomous cars, autonomous drones, and legged robots, these domains are all centered around mobility, and the primary problem is to navigate the world safely and avoid obstacles. Coarse approximations of the world state may still be viable for planning – for example, all obstacles can be inflated without estimating fine geometric detail. The primary contract in autonomous mobility, for the most part, is to not hit anything. In robot manipulation, in contrast, the goal is to actually change the state of the world, and fine geometric detail does matter, at least in the places where the robot makes contact with the environment.

Also, as robot manipulation skills become more sophisticated, I expect the communication of goals to the robot to increasingly be a bottleneck. The complexity of possible different goals to communicate is worth emphasizing in contrast to autonomous navigation robots — while mobility platforms may need to handle some level of variability in the tasks they are asked to perform ("take me here", "go there", etc), these are essentially three-dimensional goal states. In robot manipulation the variability of the possible tasks

are enormous. Even just putting a deformable object into a specific 3D configuration is an infinite-dimensional goal state (although probably approximated sufficiently well with a modest number of dimensions), and real useful manipulation problems often involve the combinatorial complexity of many different objects interacting. One route to shortcutting the challenge of communicating goals is to specify goals to the robot through demonstration. Indeed, *Learning from Demonstration* (LfD) has been a long-studied area of research in robotics and related fields. In the final chapter of this thesis, we take an LfD approach to teaching robot manipulation skills.

1.3 Related Work

There is a broad array of related work to this thesis. Rather than have repeated sections of related work, in this section I will briefly comment on the overall body of work in the literature that relates to this thesis, and highlight where additional detail can be found within each chapter.

This thesis largely sits between the worlds of computer vision and robotic manipulation, and these two worlds are intimately related in their origins. Although what we might consider the study of computer vision to hark back to at least early studies of perspective vision by Da Vinci, the first known work using digital computers to actually perform the analysis of image pixels was Lawrence Roberts' 1963 MIT PhD thesis [7], which started the study of what we affectionately know as "Blocks World" [8]. As nicely summarized by Steve Seitz [4], this study inspired for example the 1970 MIT "copy-demo" by Patrick Winston and colleagues [5], which used the computer vision principles from Roberts' Blocks World to manipulate blocks with a robot arm. Seitz highlights that a key takeaway from the MIT copy-demo was that the task of reliably finding edges in images proved to be a primary challenge. This helped spawn a significant amount of research focus in the computer vision field in edge detection, providing us with well-known results such as Canny Edge

[4] Steve Seitz. "3D Computer Vision: Past, Present, and Future" - February 2012, on YouTube.

[5] Although I could not find any published formal description of this demo, an informal description of the event can be found online, for example at https://people.csail.mit.edu/bkph/phw_copy_demo.shtml

Detection [9]. This process, of having a perceptual representation in mind, testing it out with real robot hardware and discovering what the real challenges are, and then investing new research into those challenges, is a fruitful process. It is also useful to return to the representation itself (in this case, blocks) and question what are its fundamental limitations.

A few broad trends in computer vision have had an enormous impact on this thesis. For one, the decades-long development of modern geometrical computer vision including visual Simultaneous Localization and Mapping (SLAM) has underpinned each and every chapter in this thesis. Comprehensive reviews of these topics can be abundantly found, for example [10, 11]. Additionally underpinning each chapter in this thesis has been the maturation of deep learning [12]. While geometrical computer vison and deep learning have each impacted huge swaths of research in perception, the overlap between them is of particular interest in this thesis. Additionally, and perhaps more uniquely related, is our interest in *dense* vision: dense SLAM, dense correspondence, and dense deep learning. Dense SLAM in real-time has been made possible by systems like KinectFusion [13], and dense correspondence *across scenes* was introduced by Liu et al. [14]. Within deep learning, there are many different tasks that we might consider "dense" – any task that involves a prediction for every pixel. This includes pixelwise semantic or instance segmentation, and video prediction. We are especially interested in methods which learn dense correspondence, i.e. with deep dense descriptor models [15, 16]. We provide a review of works in correspondence in Chapter 4.

A central theme of this thesis is in robots teaching themselves about vision. There have been many inspiring works in self-supervision as a theme to train visual systems [17]. In Chapter 2's related work we review a variety of systems that have demonstrated automating the labeling of certain types of perceptual representations. A special type of a robot teaching itself about vision is a robot that actually interacts with the world via its own actions, and through this interaction learns more about the world. Primary examples of this type of paradigm are works that self-supervise video prediction models [18, 19] which can then be used to accomplish robot manipulation tasks. In Chapter 3, since our robot can autonomously reorient objects in order to continue its correspondence learning process, our work also fits into this category of interactive perception. We review additional types

of robots actively teaching themselves perception in Chapter 3, and for a more extensive review of interactive perception we refer the reader to Bohg et al. [20].

In the final chapter of this thesis, we interact with an additional body of work in the literature, namely that of Imitation Learning and Learning from Demonstration (LfD). We refer the reader to existing reviews of imitation learning in robotics [21–23], and focus our review of related work on visual imitation methods [24–26]. Chapter 5 reviews these works in more detail.

Chapter 2

LabelFusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes

2.1 Introduction

Advances in neural network architectures for deep learning have made significant impacts on perception for robotic manipulation tasks. State of the art networks are able to produce high quality pixelwise segmentations of RGB images, which can be used as a key component for 6DOF object pose estimation in cluttered environments [27, 28]. However for a network to be useful in practice it must be fine tuned on labeled scenes of the specific objects targeted by the manipulation task, and these networks can require tens to hundreds of thousands of labeled training examples to achieve adequate performance. To acquire sufficient data for each specific robotics application using once-per-image human labeling would be prohibitive, either in time or money. While some work has investigated closing the gap with simulated data [29–32], our method can scale to these magnitudes with real data.

In this chapter we tackle this problem by developing an open-source pipeline that vastly reduces the amount of human annotation time needed to produce labeled RGBD datasets for

training image segmentation neural networks. The pipeline produces ground truth segmentations and ground truth 6DOF poses for multiple objects in scenes with clutter, occlusions, and varied lighting conditions. The key components of the pipeline are: leveraging dense RGBD reconstruction to fuse together RGBD images taken from a variety of viewpoints, labeling with ICP-assisted fitting of object meshes, and automatically rendering labels using projected object meshes. These techniques allow us to label once per scene, with each scene containing thousands of images, rather than having to annotate images individually. This reduces human annotation time by several orders of magnitude over traditional techniques. We optimize our pipeline to both collect many views of a scene and to collect many scenes with varied object arrangements. Our goal is to enable manipulation researchers and practitioners to generate customized datasets, which for example can be used to train any of the available state-of-the-art image segmentation neural network architectures. Using this method we have collected over 1,000,000 labeled object instances in multi-object scenes, with only a few days of data collection and without using any crowd sourcing platforms for human annotation.

Our primary contribution is the pipeline to rapidly generate labeled data, which researchers can use to build their own datasets, with the only hardware requirement being the RGBD sensor itself. We also have made available our own dataset, which is the largest available RGBD dataset with object-pose labels (352,000 labeled images, 1,000,000+ object instances). Additionally, we contribute a number of empirical results concerning the use of large datasets for practical deep-learning-based pixelwise segmentation of manipulation-relevant scenes in clutter – specifically, we empirically quantify the generalization value of varying aspects of the training data: (i) multi-object vs single object scenes, (ii) the number of background environments, and (iii) the number of views per scene.

2.2 Related Work

We review three areas of related work. First, we review pipelines for generating labeled RGBD data. Second, we review applications of this type of labeled data to 6DOF object pose estimation in the context of robotic manipulation tasks. Third, we review work related

to our empirical evaluations, concerning questions of scale and generalization for practical learning in robotics-relevant contexts.

2.2.1 Methods for Generating Labeled RGBD Datasets

Rather than evaluate RGBD datasets based on the specific dataset they provide, we evaluate the methods used to generate them, and how well they scale. Firman [33] provides an extensive overview of over 100 available RGBD datasets. Only a few of the methods used are capable of generating labels for 6DOF object poses, and none of these associated datasets also provide per-pixel labeling of objects. One of the most related methods to ours is that used to create the T-LESS dataset [34], which contains approximately 49K RGBD images of textureless objects labeled with the 6DOF pose of each object. Compared to our approach, [34] requires highly calibrated data collection equipment. They employ fiducials for camera pose tracking which limits the ability of their method to operate in arbitrary environments. Additionally the alignment of the object models to the pointcloud is a completely manual process with no algorithmic assistance. Similarly, [27] describes a high-precision motion-capture-based approach, which does have the benefit of generating high-fidelity ground-truth pose, but its ability to scale to large scale data generation is limited by: the confines of the motion capture studio, motion capture markers on objects interfering with the data collection, and time-intensive setup for each object.

Although the approach is not capable of generating the 6 DOF poses of objects, a relevant method for per-pixel labeling is described in [28]. They employ an automated data collection pipeline in which the key idea is to use background subtraction. Two images are taken with the camera at the exact same location – in the first, no object is present, while it is in the second. Background subtraction automatically yields a pixelwise segmentation of the object. Using this approach they generate 130,000 labeled images for their 39 objects. As a pixelwise labeling method, there are a few drawbacks to this approach. The first is that in order to apply the background subtraction method, they only have a single object present in each scene. In particular there are no training images with occlusions. They could in theory extend their method to support multi-object scenes by adding objects to the

scene one-by-one, but this presents practical challenges. Secondly the approach requires an accurately calibrated robot arm to move the camera in a repeatable way. A benefit of the method, however, is that it does enable pixelwise labeling of even deformable objects.

The SceneNN [35] and ScanNet [36] data generation pipelines share some features with our method. They both use an RGBD sensor to produce a dense 3D reconstruction and then perform annotations in 3D. However, since SceneNN and ScanNet are focused on producing datasets for RGDB scene understanding tasks, the type of annotation that is needed is quite different. In particular their methods provide pixelwise segmenation into generic object classes (floor, wall, couch etc.). Neither SceneNN or ScaneNet have gometric models for the specific objects in a scene and thus cannot provide 6DOF object poses. Whereas ScanNet and SceneNN focus on producing datasets for benchmarking scene understanding algorithms, we provide a pipeline to enable rapid generation labeled data for your particular application and object set.

2.2.2 Object-Specific Pose Estimation in Clutter for Robotic Manipulation

There have been a wide variety of methods to estimate object poses for manipulation. A challenge is object specificity. [27] and [28] are both state of the art pipelines for estimating object poses from RGBD images in clutter – both approaches use RGB pixelwise segmentation neural networks (trained on their datasets described in the previous section) to crop point clouds which are then fed into ICP-based algorithms to estimate object poses by registering against prior known meshes. Another approach is to directly learn pose estimation [37]. The upcoming SIXD Challenge 2017 [38] will provide a comparison of state of the art methods for 6DOF pose estimation on a common dataset. The challenge dataset contains RGBD images annotated with ground truth 6DOF object poses. This is exactly the type of data produced by our pipeline and we aim aim to submit our dataset to the 2018 challenge. There is also a trend in manipulation research to bypass object pose estimation and work directly with the raw sensor data [39–41]. Making these methods object-specific in clutter could be aided by using the pipeline presented here to train segmentation net-

works.

2.2.3 Empirical Evaluations of Data Requirements for Image Segmentation Generalization

While the research community is more familiar with the scale and variety of data needed for images in the style of ImageNet [42], the type of visual data that robots have available is much different than ImageNet-style images. Additionally, higher object specificity may be desired. In robotics contexts, there has been recent work in trying to identify data requirements for achieving practical performance for deep visual models trained on simulation data [29–32], and specifically augmenting small datasets of real data with large datasets of simulation data [29–32]. We do not know of prior studies that have performed generalization experiments with the scale of real data used here.

2.3 Data Generation Pipeline

One of the main contributions of this chapter is an efficient pipeline for generating labeled RGBD training data. The steps of the pipeline are described in the following sections: RGBD data collection, dense 3D reconstruction, object mesh generation, human assisted annotation, and rendering of labeled images.

2.3.1 RGBD Data Collection

A feature of our approach is that the RGBD sensor can either be mounted on an automated arm, as in Figure (2-1a), or the the RGBD sensor can simply be hand-carried. The benefit of the former option is a reduced human workload, while the benefit of the latter option is that no sophisticated equipment (i.e. motion capture, external markers, heavy robot arm) is required, enabling data collection in a wide variety of environments. We captured 112 scenes using the handheld approach. For the remaining 26 scenes we mounted the sensor on a Kuka IIWA, as shown in Figure (2-1a). The IIWA was programmed to perform a scanning pattern in both orientation and azimuth. Note that the arm-automated method

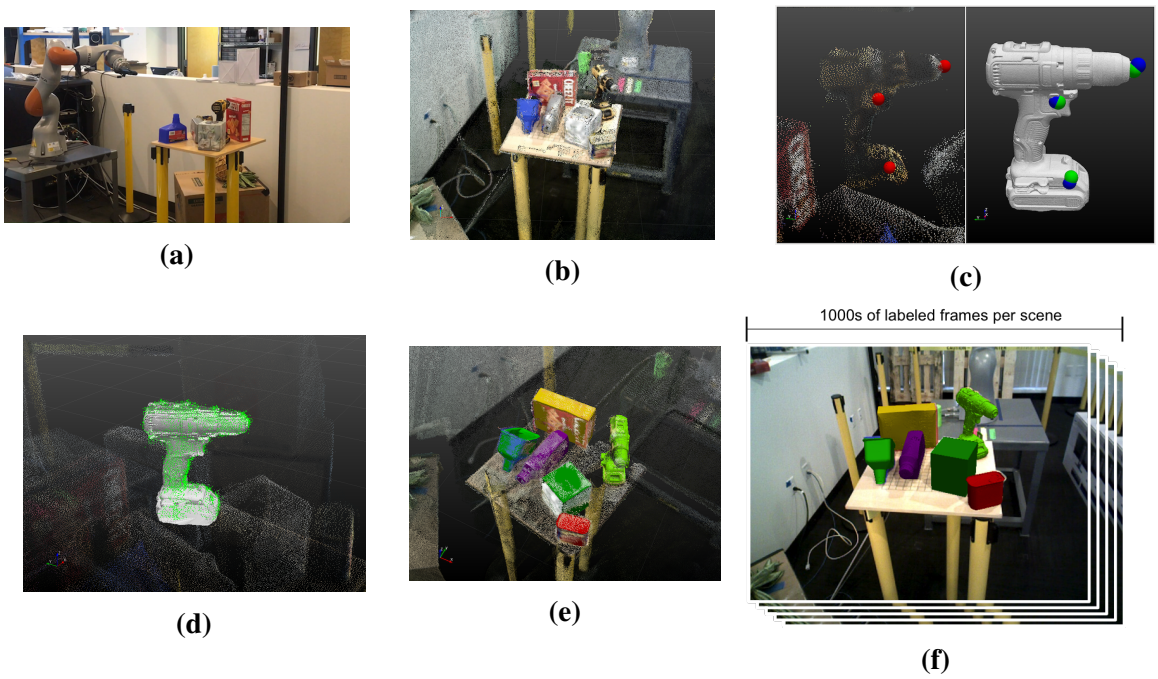


Figure 2-1: Overview of the data generation pipeline. (a) Xtion RGBD sensor mounted on Kuka IIWA arm for raw data collection. (b) RGBD data processed by ElasticFusion into reconstructed pointcloud. (c) User annotation tool that allows for easy alignment using 3 clicks. User clicks are shown as red and blue spheres. The transform mapping the red spheres to the green spheres is then the user specified guess. (d) Cropped pointcloud coming from user specified pose estimate is shown in green. The mesh model shown in grey is then finely aligned using ICP on the cropped pointcloud and starting from the user provided guess. (e) All the aligned meshes shown in reconstructed pointcloud. (f) The aligned meshes are rendered as masks in the RGB image, producing pixelwise labeled RGBD images for each view.

does not require one to know the transform between the robot and the camera; everything is done in camera frame. Our typical logs averaged 120 seconds in duration with data captured at 30Hz by the Asus Xtion Pro.

2.3.2 Dense 3D Reconstruction

The next step is to extract a dense 3D reconstruction of the scene, shown in Figure (2-1b), from the raw RGBD data. For this step we used the open source implementation of ElasticFusion [43] with the default parameter settings, which runs in realtime on our desktop with an NVIDIA GTX 1080 GPU. ElasticFusion also provides camera pose tracking relative to the local reconstruction frame, a fact that we take advantage of when rendering labeled images. Reconstruction performance can be affected by the amount of geometric features and RGB texture in the scene. Most natural indoor scenes provide sufficient texture, but

large, flat surfaces with no RGB or depth texture can occasionally incur failure modes. Our pipeline is designed in a modular fashion so that any 3D reconstruction method that provides camera pose tracking can be used in place of ElasticFusion.

2.3.3 Object Mesh Generation

A pre-processing step for the pipeline is to obtain meshes for each object. Once obtained, meshes speed annotation by enabling alignment of the mesh model rather than manually intensive pixelwise segmentation of the 3D reconstruction. Using meshes necessitates rigid objects, but imposes no other restrictions on the objects themselves. We tested several different mesh construction techniques when building our dataset. In total there are twelve objects. Four object meshes were generated using an Artec Space Spider handheld scanner. One object was scanned using Next Engine turntable scanner. For the four objects which are part of the YCB dataset [44] we used the provided meshes. One of our objects, a tissue box, was modeled using primitive box geometry. In addition our pipeline provides a volumetric meshing method using the VTK implementation of [45] that operates directly on the data already produced by ElasticFusion. Finally, there exist several relatively low cost all-in-one solutions [46], [47], [48] which use RGBD sensors such as the Asus Xtion, Intel RealSense R300 and Occipital Structure Sensor, to generate object meshes. The only requirement is that the mesh be sufficiently high quality to enable the ICP based alignment (see section 2.3.4). RGB textures of meshes are not necessary.

2.3.4 Human Assisted Annotation

One of the key contributions of the chapter is in reducing the amount of human annotation time needed to generate labeled per-pixel and pose data of objects in clutter. We evaluated several global registration methods [49–51] to try to automatically align our known objects to the 3D reconstruction but none of them came close to providing satisfactory results. This is due to a variety of reasons, but a principle one is that many scene points didn't belong to any of the objects.

To circumvent this problem we developed a novel user interface that utilizes human

input to assist traditional registration techniques. The user interface was developed using Director [52], a robotics interface and visualization framework. Typically the objects of interest are on a table or another flat surface – if so, a single click from the user segments out the table. The user identifies each object in the scene and then selects the corresponding mesh from the mesh library. They then perform a 3-click-based initialization of the object pose. Our insight for the alignment stage was that if the user provides a rough initial pose for the object, then traditional ICP-based techniques can successfully provide the fine alignment. The human provides the rough initial alignment by clicking three points on the object in the reconstructed pointcloud, and then clicking roughly the same three points in the object mesh, see Figure (2-1c). The transform that best aligns the 3 model points, shown in red, with the three scene points, shown in blue, in a least squares sense is found using the vtkLandmarkTransform function. The resulting transform then specifies an initial alignment of the object mesh to the scene, and a cropped pointcloud is taken from the points within 1cm of the roughly aligned model, as shown in green in Figure (2-1d). Finally, we perform ICP to align this cropped pointcloud to the model, using the rough aligment of the model as the initial seed. In practice this results in very good alignments even for cluttered scenes such as Figure (2-1e).

The entire human annotation process takes approximately 30 seconds per object. This is much faster than aligning the full object meshes by hand without using the 3-click technique which can take several minutes per object and results in less accurate object poses. We also compared our method with human labeling (polygon-drawing) each image, and found intersection over union (IoU) above 80%, with approximately four orders of magnitude less human effort per image (supplementary figures on our website).

2.3.5 Rendering of Labeled Images and Object Poses

After the human annotation step of Section 2.3.4, the rest of the pipeline is automated. Given the previous steps it is easy to generate per-pixel object labels by projecting the 3D object poses back into the 2D RGB images. Since our reconstruction method, ElasticFusion, provides camera poses relative to the local reconstruction frame, and we have already

aligned our object models to the reconstructed pointcloud, we also have object poses in each camera frame, for each image frame in the log. Given object poses in camera frame it is easy to get the pixelwise labels by projecting the object meshes into the rendered images. An RGB image with projected object meshes is displayed in Figure (2-1f).

2.3.6 Discussion

As compared to existing methods such as [27, 34, 53] our method requires no sophisticated calibration, works for arbitrary rigid objects in general environments, and requires only 30 seconds of human annotation time per object per scene. Since the human annotation is done on the full 3D reconstruction, one labeling effort automatically labels thousands of RGBD images of the same scene from different viewpoints.

2.4 Results

We first analyze the effectiveness of the LabelFusion data generation pipeline (Section 2.4.1). We then use data generated from our pipeline to perform practical empirical experiments to quantify the generalization value of different aspects of training data (Section 2.4.2).

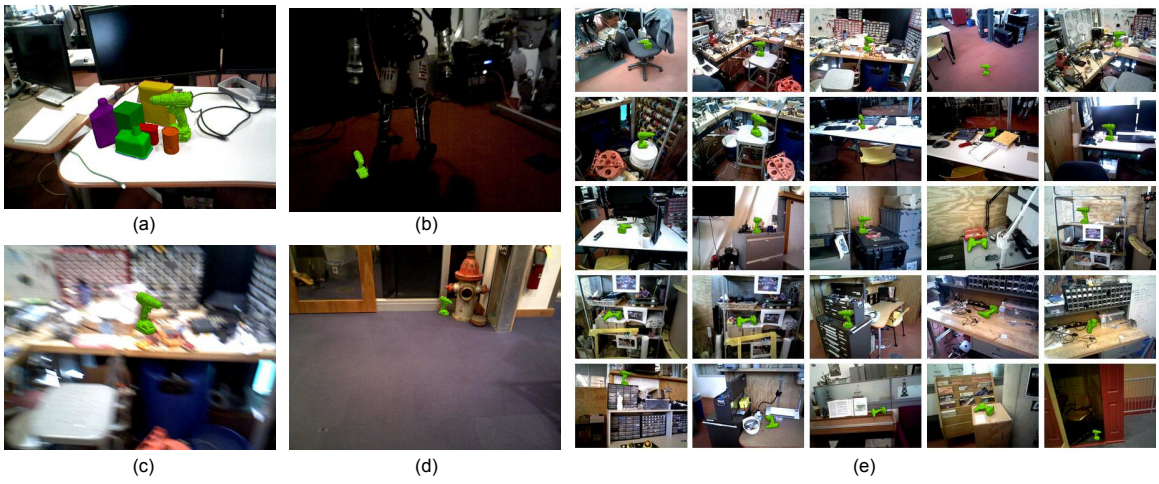


Figure 2-2: Examples of labeled data generated by our pipeline: (a) heavily cluttered multi-object, (b) low light conditions, (c) motion blur, (d) distance from object, (e) 25 different environments. All of these scenes were collected by hand-carrying the RGBD sensor.

# objects	12
# distinct scenes	105 single/double object 33 with 6+ objects
# unique object instances aligned	339
avg duration of single scene	120 seconds, 3600 RGBD frames
# labeled RGBD frames	352,000
# labeled object instances	1,000,000+

Table 2.1: Dataset Description

2.4.1 Evaluation of Data Generation Pipeline

LabelFusion has the capability to rapidly produce large amounts of labeled data, with minimal human annotation time. In total we generated over 352,000 labeled RGBD images, of which over 200,000 were generated in approximately one day by two people. Because many of our images are multi- object, this amounts to over 1,000,000 labeled object instances. Detailed statistics are provided in Table 2.1. The pipeline is open-source and intended for use. We were able to create training data in a wide variety of scenarios; examples are provided in Figure 2-2. In particular, we highlight the wide diversity of environments enabled by hand-carried data collection, the wide variety of lighting conditions, and the heavy clutter both of backgrounds and of multi-labeled object scenes.

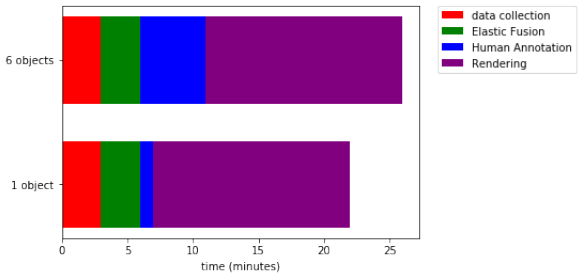


Figure 2-3: Time required for each step of pipeline.

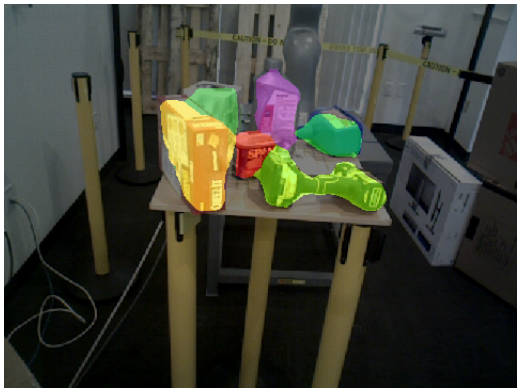


Figure 2-4: Example segmentation performance (alpha-blended with RGB image) of network (e) on a multi-object test scene.

For scaling to large scale data collection, the time required to generate data is critical. Our pipeline is highly automated and most components run at approximately real-time, as shown in Figure 2-3. The amount of human time required is approximately 30 seconds per object per scene, which for a typical single-object scene is less than real-time. Post-processing runtime is several times greater than real-time, but is easily parallelizable – in practice, a small cluster of 2-4 modern desktop machines (quad-core Intel i7 and Nvidia GTX 900 series or higher) can be made to post-process the data from a single sensor at real-time rates. With a reasonable amount of resources (one to two people and a handful of computers), it would be possible to keep up with the real-time rate of the sensor (generating labeled data at 30 Hz).

2.4.2 Empirical Evaluations: How Much Data Is Needed For Practical Object-Specific Segmentation?

With the capability to rapidly generate a vast sum of labeled real RGBD data, questions of "how much data is needed?" and "which types of data are most valuable?" are accessible. We explore practical generalization performance while varying three axes of the training data: (i) whether the training set includes multi-object scenes with occlusions or only single-object scenes, (ii) the number of background environments, and (iii) the number of views used per scene. For each, we train a state-of-the-art ResNet segmentation network [54] with different subsets of training data, and evaluate each network's generalization performance on common test sets. Further experimental details are provided in our supplementary material; due to space constraints we can only summarize results here.

First, we investigate whether there is a benefit of using training data with heavily occluded and cluttered multi-object scenes, compared to training with only single-object scenes. Although they encounter difficulties with heavy occlusions in multi-object scenes, [28] uses purely single-object scenes for training. We trained five different networks to enable comparison of segmentation performance on novel scenes (different placements of the objects) for a single background environment. Results of segmentation performance

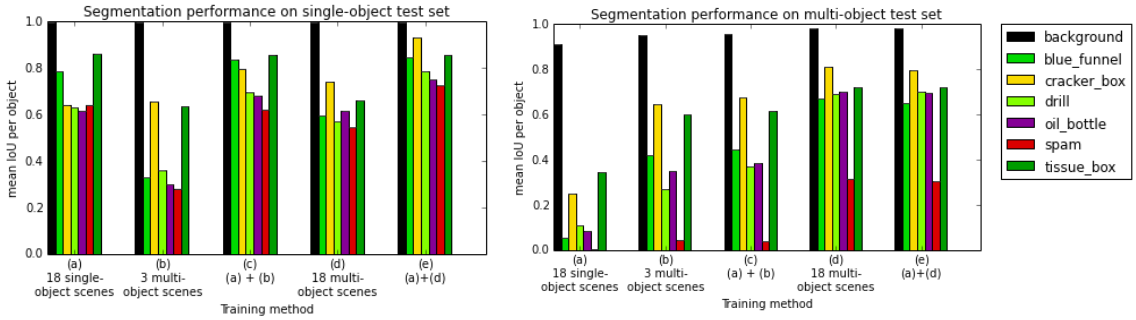


Figure 2-5: Comparisons of training on single-object vs. multi-object scenes and testing on single-object (left) and multi-object (right) scenes.

on novel scenes (measured using the mean IoU, intersection over union, per object) show an advantage given multi-object occluded scenes (b) compared to single-object scenes (a) (Figure 2-5, right). In particular, the average IoU per object increases 190% given training set (b) instead of (a) in Figure 2-5, right, even though (b) has strictly less labeled pixels than (a), due to occlusions. This implies that the value of the multi-object training data is more valuable per pixel than the single-object training data. When the same amount of scenes for the single-object scenes are used to train a network with multi-object scenes (d), the increase in IoU performance averaged across objects is 369%. Once the network has been trained on 18 multi-object scenes (d), an additional 18 single-object training scenes have no noticeable effect on multi-object generalization (e). For generalization performance on single-object scenes (Figure 2-5, left), this effect is not observed; single-object training scenes are sufficient for IoU performance above 60%.

Second, we ask: how does the performance curve grow as more and more training data is added from different background environments? To test this, we train different networks respectively on 1, 2, 5, 10, 25, and 50 scenes each labeled with a single drill object. The smaller datasets are subsets of the larger datasets; this directly allows us to measure the value of providing more data. The test set is comprised of 11 background environments which none of the networks have seen. We observe a steady increase in segmentation performance that is approximately logarithmic with the number of training scene backgrounds used (Figure 2-7, left). We also took our multi-object networks trained on a single background and tested them on the 11 novel environments with the drill. We observe an advantage of the multi-object training data with occlusions over the single-

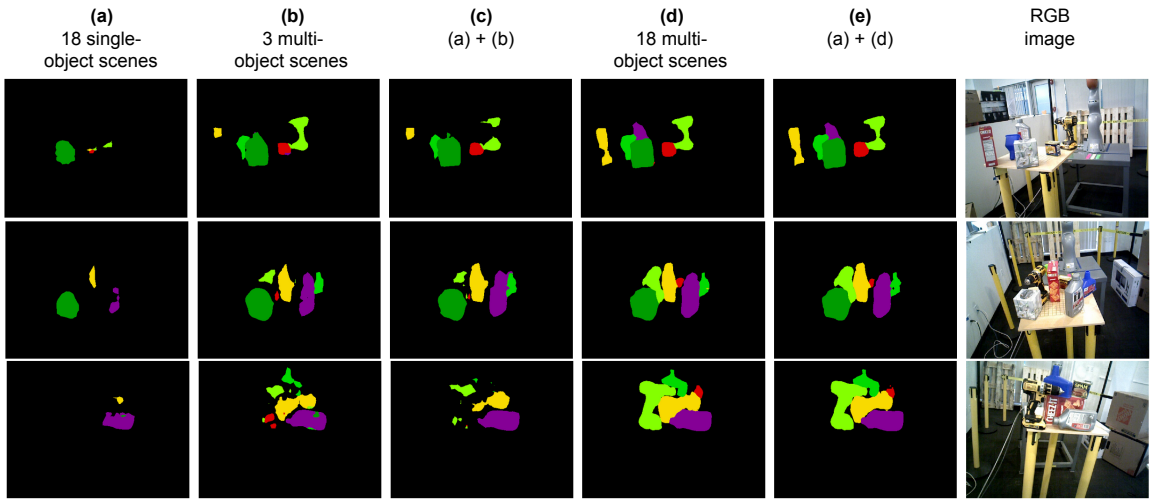


Figure 2-6: Comparison of segmentation performance on novel multi-object test scenes. Networks are either trained on (a) single object scenes only, (b,d), multi-object test scenes only, or a mixture (c,e).

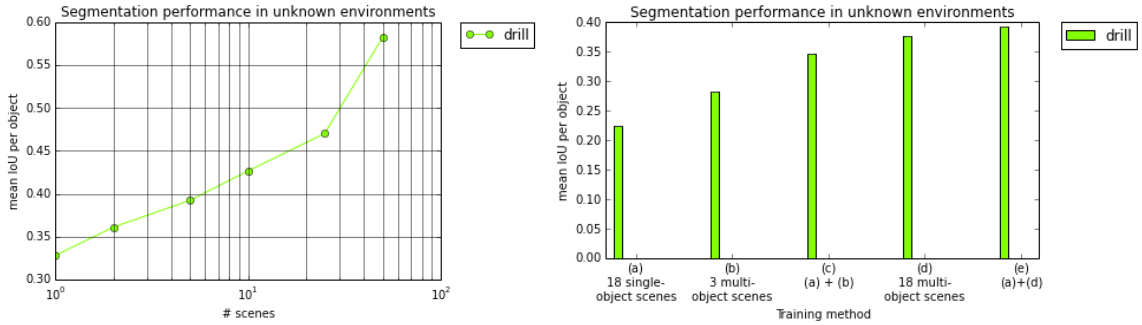


Figure 2-7: (left) Generalization performance as a function of the number of environments provided at training time, for a set of six networks trained on 50 different scenes or some subset ({1, 2, 5, 10, 25}) of those scenes. (right) Performance on the same test set of unknown scenes, but measured for the 5 training configurations for the multi-object, single-environment-only setup described previously.

object training data in generalizing to novel background environments (Figure 2-7, right).

Third, we investigate whether 30 Hz data is necessary, or whether significantly less data suffices (Figure 2-9). We perform experiments with downsampling the effective sensor rate both for robot-arm-mounted multi-object single-background training set (e), and the hand-carried many-environments dataset with either 10 or 50 scenes. For each, we train four different networks, where one has all data available and the others have downsampled data at respectively 0.03, 0.3, and 3 Hz. We observe a monotonic increase in segmentation performance as the effective sensor rate is increased, but with heavily diminished returns after 0.3 Hz for the slower robot-arm-mounted data ($\sim$0.03 m/s camera motion velocity). The hand-carried data ($\sim$0.05 - 0.17 m/s) shows more gains with higher rates.

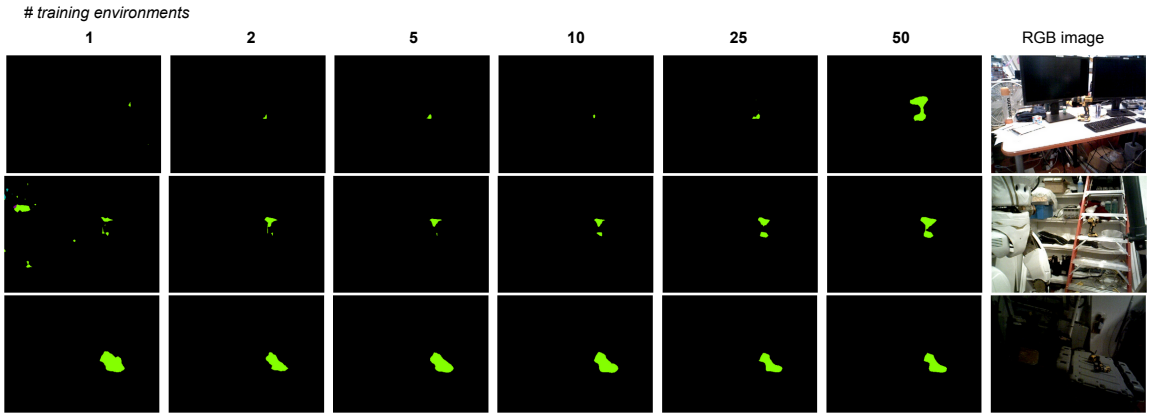


Figure 2-8: Comparison of segmentation performance on novel background environments. Networks were trained on {1, 2, 5, 10, 25, 50} background environments.

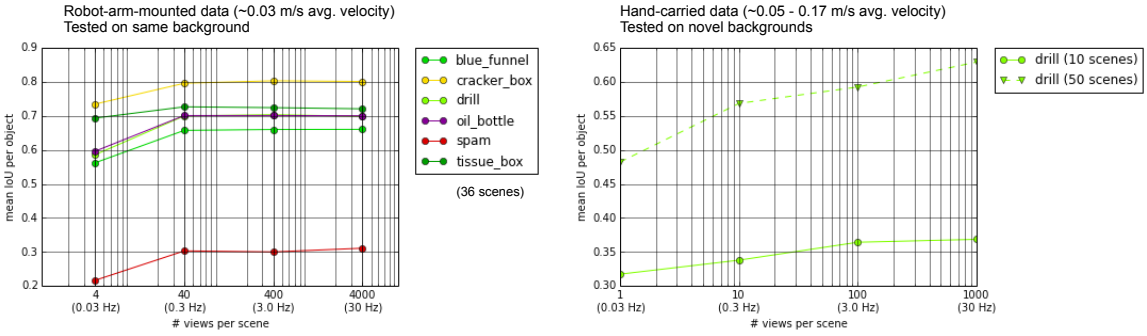


Figure 2-9: Pixelwise segmentation performance as a function of the number of views per scene, reduced by downsampling the native 30 Hz sensor to {0.03, 0.3, 3.0.} Hz.

2.5 Conclusion

This chapter introduces LabelFusion, our pipeline for efficiently generating RGBD data annotated with per-pixel labels and ground truth object poses. Specifically only a few minutes of human time are required for labeling a scene containing thousands of RGBD images. LabelFusion is open source and available for community use, and we also supply an example dataset generated by our pipeline [55].

The capability to produce a large, labeled dataset enabled us to answer several questions related to the type and quantity of training data needed for practical deep learning segmentation networks in a robotic manipulation context. Specifically we found that networks trained on multi-object scenes performed significantly better than those trained on single object scenes, both on novel multi-object scenes with the same background, and on single-object scenes with new backgrounds. Increasing the variety of backgrounds in the training data for single-object scenes also improved generalization performance for new backgrounds, with approximately 50 different backgrounds breaking into above-50% IoU on entirely novel scenes. Our recommendation is to focus on multi-object data collection in a variety of backgrounds for the most gains in generalization performance.

We hope that our pipeline lowers the barrier to entry for using deep learning approaches for perception in support of robotic manipulation tasks by reducing the amount of human time needed to generate vast quantities of labeled data for *your* specific environment and set of objects. It is also our hope that our analysis of segmentation network performance provides guidance on the type and quantity of data that needs to be collected to achieve desired levels of generalization performance.

Chapter Acknoweldgement

This chapter was joint work with Pat Marion, Lucas Manuelli, and Russ Tedrake. We thank Matthew O'Kelly for his guidance with segmentation networks and manuscript feedback. We also thank Allison Fastman and Sammy Creasey of Toyota Research Institute for their help with hardware, including object scanning and robot arm automation. David

Johnson of Draper Laboratory and Shuran Song of Princeton University provided valuable input on training. We are grateful we were able to use the robot arm testing facility from Toyota Research Institute. This work was supported by the Air Force/Lincoln Laboratory award no. 7000374874, by the Defense Advanced Research Projects Agency via Air Force Research Laboratory award FA8750-12-1-0321, and by NSF Contract IIS-1427050. The views expressed are not endorsed by the sponsors.

Chapter 3

Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation

3.1 Introduction

What is the right object representation for manipulation? We would like robots to visually perceive scenes and learn an understanding of the objects in them that (i) is task-agnostic and can be used as a building block for a variety of manipulation tasks, (ii) is generally applicable to both rigid and non-rigid objects, (iii) takes advantage of the strong priors provided by 3D vision, and (iv) is entirely learned from self-supervision. This is hard to achieve with previous methods: much recent work in grasping does not extend to grasping specific objects or other tasks, whereas task-specific learning may require many trials to generalize well across object configurations or other tasks. In this chapter we present Dense Object Nets, which build on recent developments in self-supervised dense descriptor learning, as a consistent object representation for visual understanding and manipulation. We demonstrate they can be trained quickly (approximately 20 minutes) for a wide variety of previously unseen and potentially non-rigid objects. We additionally present novel contributions to enable multi-object descriptor learning, and show that by modifying our training

procedure, we can either acquire descriptors which generalize across classes of objects, or descriptors that are distinct for each object instance. Finally, we demonstrate the novel application of learned dense descriptors to robotic manipulation. We demonstrate grasping of specific points on an object across potentially deformed object configurations, and demonstrate using class general descriptors to transfer specific grasps across objects in a class.

While task-specific reinforcement learning can achieve impressively dexterous skills for a given specific task [39], it is unclear which is the best route to efficiently achieving many different tasks. Other recent work [40, 41] can provide very general grasping functionality but does not address specificity. Achieving specificity, the ability to accomplish specific tasks with specific objects, may require solving the data association problem. At a coarse level the task of identifying individual objects to manipulate can be solved by instance segmentation, as demonstrated in the Amazon Robotics Challenge (ARC) [56, 57] or [58]. Whole-object-level segmentation, however, does not provide any information on the rich structure of the objects themselves, and hence may not be an appropriate representation for solving more complex tasks. While not previously applied to the robotic manipulation domain, recent work has demonstrated advances in learning dense pixel level data association [15, 16], including self-supervision from raw RGBD data [16], which inspired our present work.

In this chapter, we propose and demonstrate using dense visual description as a representation for robotic manipulation. We demonstrate the first autonomous system that can entirely self-supervise to learn consistent dense visual representations of objects, and the first system we know of that is capable of performing the manipulation demonstrations we provide. Specifically, with no human supervision during training, our system can grasp specific locations on deformable objects, grasp semantically corresponding locations on instances in a class, and grasp specific locations on specific instances in clutter. Towards this goal, we also provide practical contributions to dense visual descriptor learning with general computer vision applications outside of robotic manipulation. We call our visual representations Dense Object Nets, which are deep neural networks trained to provide

Code, data, and video available: github.com/RobotLocomotion/pytorch-dense-correspondence

dense (pixelwise) description of objects.

Contributions. We believe our largest contribution is that we introduce dense descriptors as a representation useful for robotic manipulation. We've also shown that self-supervised dense visual descriptor learning can be applied to a wide variety of potentially non-rigid objects and classes (47 objects so far, including 3 distinct classes), can be learned quickly (approximately 20 minutes), and can enable new manipulation tasks. In example tasks we grasp specific points on objects across potentially deformed configurations, do so with object instance-specificity in clutter, or transfer specific grasps across objects in a class. We also contribute novel techniques to enable multi-object distinct dense descriptors, and show that by modifying the loss function and sampling procedure, we can either acquire descriptors which generalize across classes of objects, or descriptors that are distinct for each object instance. Finally, we contribute general training techniques for dense descriptors which we found to be critical to achieving good performance in practice.

Chapter Organization. In Section 3.2 we describe related work. As preliminary in Section 3.3.1 we describe the general technique for self-supervising dense visual descriptor learning, which is from [16] but reproduced here for clarity. We then describe additional techniques we've developed for object-centric visual descriptors in Section 3.3.2, and Section 3.3.3 describes techniques for distinct multi-object descriptors. Section 3.4 describes our experimental setup for our autonomous system, and Section 5 describes our results: our learned visual descriptors for a wide variety of objects (Section 3.5.1) multi-object descriptors and selective class generalization (Sections 3.5.2 and 3.5.3), and robotic manipulation demonstrations (Section 3.5.4). The next chapter, Chapter 4, presents new formulations of the underlying visual correspondence problem.

3.2 Related Work

We review three main areas of related work: learned descriptors, self-supervised visual learning for robots, and robot learning for specific tasks. The task of correspondence estimation from multiple views of the same scene is fundamental in computer vision, whereas dense semantic correspondence across *different* scenes was popularized by [14]. Recent

advances have been made by introducing a pixel-wise variant of contrastive loss [59] combined with deep convolutional networks, as in Choy et al. [15] and Schmidt et al. [16]. For cross-instance semantic correspondence, [15] relies on human annotations, while [16] learns these unsupervised, as we do here. Other work [60] uses image warping to learn descriptors, and most require manually annotated labels [61–63]. Zeng et al. [64] also uses dense 3D reconstruction to provide automated labeling, but for descriptors of 3D volume patches. Some of these works [16, 60, 63], like ours, learn descriptors for specific object instances or classes, while others [64] learn descriptors for establishing correspondence of arbitrary data. None of these prior works in dense visual learning involve robots.

In the area of self-supervised visual robot learning, while some recent work has sought to understand 'how will the world change given the robot's action?" [18,65] in this work we instead ask "what is the current visual state of the robot's world?". We address this question with a dense description that is consistent across viewpoints, object configurations and (if desired) object classes. At the coarse level of semantic segmentation several works from the Amazon Robotics Challenge used robots to automate the data collection and annotation process through image-level background subtraction [56,57,66]. In contrast this work uses 3D reconstruction-based change detection and dense pixelwise correspondences, which provides a much richer supervisory signal for use during training.

In the area of robot learning for a specific task there have been impressive works on end-to-end reinforcement learning [39, 67]. In these papers the goal is to learn a specific task, encoded with a reward function, whereas we learn a general task agnostic visual representation. There have also been several works focusing on grasping from RGB or depth images [40, 41, 66, 68]. These papers focus on successfully grasping any item out of a pile, and are effectively looking for graspable features. They have no consistent object representation or specific location on that object, and thus the robotic manipulation tasks we demonstrate in Section 3.5.4, e.g. grasping specific points on an object across potentially deformed object configurations, are out of scope for these works.

3.3 Methodology

3.3.1 Preliminary: Self-Supervised Pixelwise Contrastive Loss

We use self-supervised pixelwise contrastive loss, as developed in [15, 16]. This learns a dense visual descriptor mapping which maps a full-resolution RGB image, $\mathbb{R}^{W \times H \times 3}$ to a dense descriptor space, $\mathbb{R}^{W \times H \times D}$, where for each pixel we have a D-dimensional descriptor vector. Training is performed in a Siamese fashion, where a pair of RGB images, I_a and I_b are sampled from one RGBD video, and many pixel matches and non-matches are generated from the pair of images. A pixel $u_a \in \mathbb{R}^2$ from image I_a is a match with pixel u_b from image I_b if they correspond to the same vertex of the dense 3D reconstruction (Figure 3-1 (c-f)). The dense descriptor mapping is trained via pixelwise contrastive loss. The loss function aims to minimize the distance between descriptors corresponding to a match, while descriptors corresponding to a non-match should be at least a distance M apart, where M is a margin parameter. The dense descriptor mapping $f(\cdot)$ is used to map an image $I \in \mathbb{R}^{W \times H \times 3}$ to descriptor space $f(I) \in \mathbb{R}^{W \times H \times D}$. Given a pixel u we use $f(I)(u)$ to denote the descriptor corresponding to pixel u in image I. We simply round the real-valued pixel $u \in \mathbb{R}^2$ to the closest discrete pixel value $u \in \mathbb{N}^2$, but any continuously-differentiable interpolation can be used for sub-pixel resolution. We denote $D(\cdot)$ as the L_2 distance between a pair of pixel descriptors: $D(I_a, u_a, I_b, u_b) \triangleq ||f(I_a)(u_a) - f(I_b)(u_b)||_2$. At each iteration of training, a large number (on the order of 1 million total) of matches N_{matches} and non-matches $N_{\text{non-matches}}$ are generated between images I_a and I_b. The images are mapped to corresponding descriptor images via $f(\cdot)$ and the loss function is

$$\mathcal{L}_{\text{matches}}(I_a, I_b) = \frac{1}{N_{\text{matches}}} \sum_{N_{\text{matches}}} D(I_a, u_a, I_b, u_b)^2 \tag{3.1}$$

$$\mathcal{L}_{\text{non-matches}}(I_a, I_b) = \frac{1}{N_{\text{non-matches}}} \sum_{N_{\text{non-matches}}} \max(0, M - D(I_a, u_a, I_b, u_b))^2 \tag{3.2}$$

$$\mathcal{L}(I_a, I_b) = \mathcal{L}_{\text{matches}}(I_a, I_b) + \mathcal{L}_{\text{non-matches}}(I_a, I_b) \tag{3.3}$$

3.3.2 Training Procedures for Object-Centric Descriptors

Prior work [16] has used dynamic reconstruction [69] of raw RGBD data for only within-scene data association and remarkably showed that even without cross-scene data association, descriptors could be learned that were consistent across many dynamic scenes of the upper body of a human subject. While dynamic reconstruction is powerful, the challenges of topology changes [70] and difficulties of occlusion make it difficult to reliably deploy for an autonomous system. Schmidt et al. [16] also used data associations from static scene reconstructions for the task of relocalization in the same static environment. In contrast we sought to use only static reconstruction but seek consistency for dynamic objects. Other work [60] obtains dense descriptor consistency for a curated dataset of celebrity faces using only image warping for data association.

Using our robot mounted camera we are able to reliably collect high quality dense reconstructions for static scenes. Initially we applied only static-scene reconstruction to learn descriptors for specific objects, but we found that the learned object descriptors were not naturally consistent for challenging datasets with objects in significantly different configurations. Subsequently we developed techniques that leverage 3D reconstruction change detection, data augmentation, and loss function balancing to reliably produce consistent object representations with only static-scene data association for the wide variety of objects we have tested. These techniques also improve the precision of correspondences, as is discussed in Section 3.5.1. While we have tried many other ideas,these are the techniques that were empirically found to significantly improve performance.

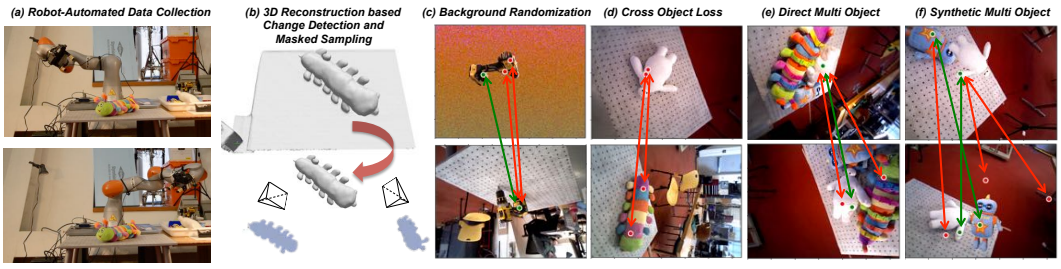


Figure 3-1: Overview of the data collection and training procedure. (a) automated collection with a robot arm. (b) change detection using the dense 3D reconstruction. (c)-(f) matches depicted in green, non-matches depicted in red.

Object masking via 3D change detection. Since we are trying to learn descriptors of

objects that take up only a fraction of a full image, we observe significant improvements if the representational power of the models are focused on the objects rather than the backgrounds. A 640×480 image contains $307,200$ pixels but an image in our dataset may have as few as 1,000 to 10,000 of those pixels, or .3%-3%, that correspond to the object of interest. Initial testing with human-labeled object masks [2] showed that if matches for data associations were sampled only on the object (while non-matches were sampled from the full image) then correspondence performance was significantly improved. In order to provide autonomous object masking without any human input, we leverage our 3D reconstructions and results from the literature on 3D change detection [71] to recover the object-only part of the reconstruction (Figure 3-1b). Projecting this geometry into each camera frame yields object masks for each image. We want to emphasize that automatic object masking enables many other techniques in this chapter, including: background domain randomization, cross-object loss, and synthetic multi-object scenes.

Background domain randomization. A strategy to encourage cross-scene consistency is to enforce that the learned descriptors are not reliant on the background. Since we have autonomously acquired object masks, we can domain randomize [72] the background (Figure 3-1c top) to encourage consistency – rather than memorizing the background (i.e. by describing the object by where it is relative to a table edge), the descriptors are forced to be representative of only the object.

Hard-negative scaling. Although as in [16] we originally normalized $\mathcal{L}_{\text{matches}}$ and $\mathcal{L}_{\text{non-matches}}$ by N_{matches} and $N_{\text{non-matches}}$, respectively, we found that what we call the "hard-negative rate", i.e. the percentage of sampled non-matches for which $M - D(I_a, u_a, I_b, u_b) > 0$ would quickly drop well below 1% during training. While not precisely hard-negative mining [73], we empirically measure improved performance if rather than scaling $\mathcal{L}_{\text{non-matches}}$ by $N_{\text{non-matches}}$, we adaptively scale by the number of hard negatives in the non-match sampling, $N_{\text{hard-negatives}}$, where $\mathbb{1}$ is the indicator function:

$$N_{\text{hard-negatives}} = \sum_{N_{\text{non-matches}}} \mathbb{1}(M - D(I_a, u_a, I_b, u_b) > 0) \tag{3.4}$$

$$\mathcal{L}_{\text{non-matches}}(I_a, I_b) = \frac{1}{N_{\text{hard-negatives}}} \sum_{N_{\text{non-matches}}} max(0, M - D(I_a, u_a, I_b, u_b))^2 \tag{3.5}$$

Data augmentation and normalization. While we collect only a modest number of scenes (4-10) per object or class, we ensure they are diverse in orientations, crops, and lighting conditions. We also applied synthetic 180-degree rotations randomly to our images. Additionally we find gains in performance by projecting all features to the unit sphere, i.e. $f(I)(u) \leftarrow \frac{f(I)(u)}{||f(I)(u)||}$ when using high-dimensional descriptors spaces (i.e., more than $D = 4$).

3.3.3 Multi-Object Dense Descriptors

We of course would like robots to have dense visual models of more than just one object. When we began this work it wasn't obvious to us what scale of changes to our training procedure or model architecture would be required in order to simultaneously (a) achieve individual single-object performance comparable to a single-object-only model, while also (b) learn dense visual descriptors for objects that are *globally distinct* – i.e., the bill of a hat would occupy a different place in descriptor space than the handle of a mug. To achieve distinctness, we introduce three strategies:

i. Cross-object loss. The most direct way to ensure that different objects occupy different subsets of descriptor space is to directly impose *cross-object loss* (Figure 3-1d). Between two different objects, we know that each and every pair of pixels between them is a non-match. Accordingly we randomly select two images of two different objects, randomly sample many pixels from each object (enabled by object masking), and apply non-match loss (with hard-negative scaling) to all of these pixel pairs.

ii. Direct training on multi-object scenes. A nice property of pixelwise contrastive loss, with data associations provided by 3D geometry, is that we can directly train on multi-object, cluttered scenes *without any individual object masks* (Figure 3-1e). This is in contrast with training pixelwise semantic segmentation, which requires labels for each individual object in clutter that may be difficult to attain, i.e. through human labeling. With pixel-level data associations provided instead by 3D geometry, the sampling of matches and the loss function still makes sense, even in clutter.

iii. Synthetic multi-object scenes. We can also synthetically create multi-object scenes

by layering object masks [56]. To use dense data associations through synthetic image merging, we prune matches that become occluded during layering (Figure 3-1f). A benefit of this procedure is that we can create a combinatorial number of "multi-object" scenes from only single object-scenes, and can cover a wide range of occlusion types without collecting physical scenes for each.

3.4 Experimental

Data Collection and Pre-Processing. The minimum requirement for raw data is to collect an RGBD video of an object or objects. Figure 3-1 shows our experimental setup; we utilize a 7-DOF robot arm (Kuka IIWA LBR) with an RGBD sensor (Primesense Carmine 1.09) mounted at the end-effector. With the robot arm, data collection can be highly automated, and we can achieve reliable camera poses by using forward kinematics along with knowledge of the camera extrinsic calibration. For dense reconstruction we use TSDF fusion [45] of the depth images with camera poses provided by forward kinematics. An alternative route to collecting data which does not require a calibrated robot is to use a dense SLAM method (for example, [13, 43]). In between collecting RGBD videos, the object of interest should be moved to a variety of configurations, and the lighting can be changed if desired. While for many of our data collections a human moved the object between configurations, we have also implemented and demonstrated (see our video) the robot autonomously rearranging the objects, which highly automates the object learning process. We employ a Schunk two-finger gripper and plan grasps directly on the object point cloud.If multiple different objects are used, currently the human must still switch the objects for the robot and indicate which scenes correspond to which object, but even this information could be automated by the robot picking objects from an auxiliary bin.

Training Dense Descriptors. For training, at each iteration we randomly sample between some subset of specified image comparison types (Single Object Within Scene, Different Object Across Scene, Multi Object Within Scene, Synthetic Multi Object), and then sample some set of matches and non-matches for each. In this chapter, we use only static-scene reconstructions, so pixel matches between images can be easily found by raycasting

and reprojecting against the dense 3D reconstruction model, and appropriately checking for occlusions and field-of-view constraints. For the dense descriptor mapping we train a 34-layer, stride-8 ResNet pretrained on ImageNet, but we expect any fully-convolutional network (FCN) that has shown effectiveness on semantic segmentation tasks to work well.

3.5 Results

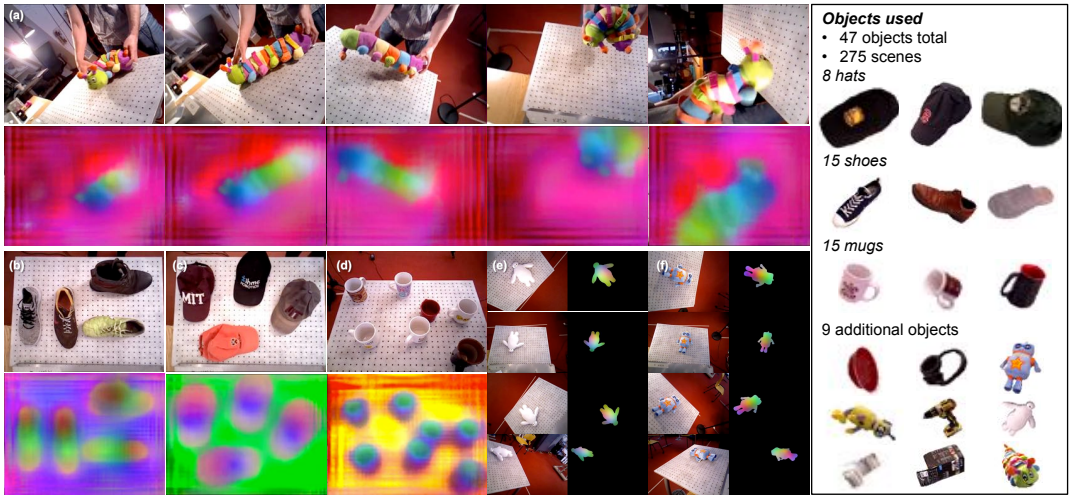


Figure 3-2: Learned object descriptors can be consistent across significant deformation (a) and, if desired, across object classes (b-d). Shown for each (a) and (b-d) are RGB frames (top) and corresponding descriptor images (bottom) that are the direct output of a feed-forward pass through a trained network. (e)-(f) shows that we can learn descriptors for low texture objects, with the descriptors masked for clear visualization. Our object set is also summarized (right).

3.5.1 Single-Object Dense Descriptors

We observe that with our training procedures described in Section 3.3.2, for a wide variety of objects we can acquire dense descriptors that are invariant to viewpoint, configuration, and deformation. The variety of objects includes moderately deformable objects such as soft plush toys, shoes, mugs, and hats, and can include very low-texture objects (Figure 3-2). Many of these objects were just grabbed from around the lab (including the authors' and labmates' shoes and hats), and dense visual models can be reliably trained with the same network architecture and training parameters. The techniques in Section 3.3.2 provide significant improvement in both (a) qualitative consistency over a wide variety of viewpoints, and (b) quantitative precision in correspondences. As with other works that learn pairwise

mappings to some descriptor space [74], in practice performance can widely vary based on specific sampling of data associations and non-associations used during training. One way to quantitatively evaluate correspondence precision is with human-labeled (used only for evaluation; never for training) correspondences across two images of an object in *different* configurations. Given two images I_a, I_b containing the same object and pixel locations $u_a^* \in I_a, u_b^* \in I_b$ corresponding to the same physical point on the object, we can use our dense descriptors to estimate u_b^* as $\hat{u}_b$:

$$\hat{u}_b \triangleq \underset{u_b \in I_b}{\arg\min}\ D(I_a, u_a^*, I_b, u_b) \tag{3.6}$$

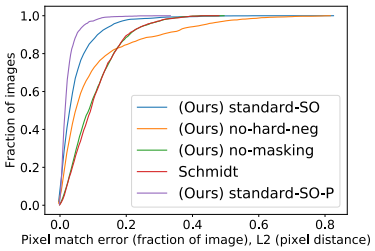
Figure 3-3 (b-c) shows a quantitative comparison of ablative experiments, for four different training procedures described in Figure 3-3a. Our new standard single-object training procedure (**standard-SO**) performs significantly better than our implementation of prior work's training procedures (**Schmidt**), and we isolate and measure significant improvement in correspondence precision for both object-masking and hard-negative scaling. We also find that for some low-texture objects, orientation randomization and background domain randomization are critical for attaining consistent object descriptors. Otherwise the model may learn to memorize which side of the object is closest the table, rather than a consistent object model (Figure 3-4b). Background domain randomization is most beneficial for smaller datasets, where it can significantly reduce overfitting and encourage consistency (Figure 3-4a); it is less critical for high-texture objects and larger datasets.

3.5.2 Multi-Object Dense Descriptors

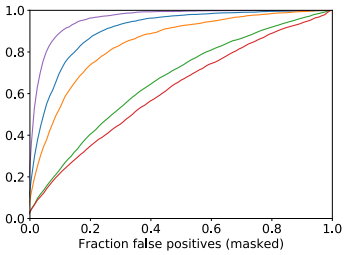
An early observation during experimentation was that *overlap* in descriptor space naturally occurs if the same model is trained simultaneously on different singulated objects, where sampling of matches and non-matches was only performed *within scene*. Since there is no component of the loss function that requires different objects to occupy different subsets of descriptor space, the model maps them to an overlapping subset of descriptor space, distinct from the background but not each other (Figure 3-5a). Accordingly we sought to answer the question of whether or not we could *separate* these objects into unique parts of

	single or multi object dataset	masked match sampling	scale by hard negatives	cross-object loss
standard-SO	single	✓	✓	
no-masking	single		✓	
no-hard-neg	single	✓		
Schmidt	single			
consistent	multi	✓	✓	
specific	multi	✓	✓	✓

(a)



(b)



(c)

Figure 3-3: (a) table describing the network training procedures referenced in experiments. (**standard-SO** = "standard single object". (b) Plots the cdf of the L2 pixel distance (normalized by image diagonal, 800 for a 640 x 480 image) between the best match $\hat{u}_b$ and the true match u_b^*, e.g. for **standard-SO** in 93% of image pairs the normalized pixel distance between u_b^* and $\hat{u}_b$ is less than 13%. All networks were trained on the same dataset. (c) Plots the cdf of the fraction of pixels u_b of the object pixels with $D(I_a, u_a^*, I_b, u_b) < D(I_a, u_a^*, I_b, u_b^*)$, i.e. they are closer in descriptor space to u_a^* than the true match u_b^*.

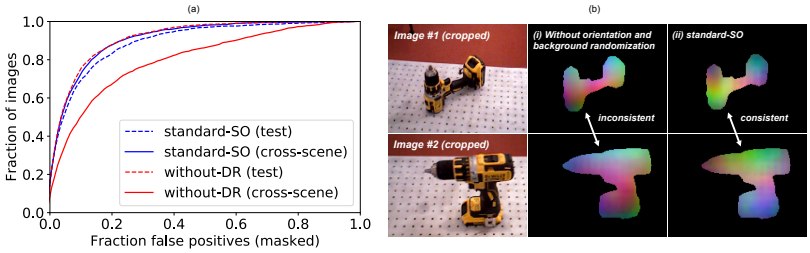
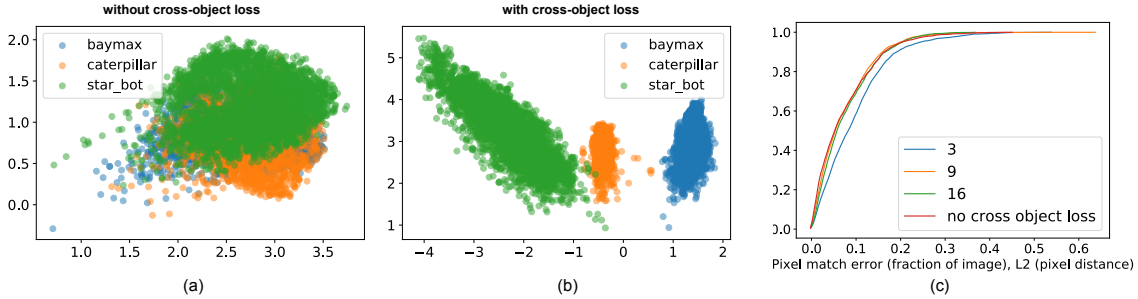


Figure 3-4: (a), with same axes as Figure 3-3b, compares **standard-SO** with **without-DR**, for which the only difference is that **without-DR** used no background domain randomization during training. The dataset used for (a) is of three objects, 4 scenes each. (b) shows that for a dataset containing 10 scenes of a drill, learned descriptors are inconsistent without background and orientation randomization during training (middle), but consistent with them (right).



(a) (b) (c)

Figure 3-5: Comparison of training without any distinct object loss (a) vs. using cross-object loss (b). In (b), 50% of training iterations applied cross-object loss and 50% applied single-object within-scene loss, whereas (a) is 100% single-object within-scene loss. The plots show a scatter of the descriptors for 10,000 randomly-selected pixels for each of three distinct objects. Networks were trained with $D = 2$ to allow direct cluster visualization. (c) Same axes as Figure 3-3 (a). All networks were trained on the same 3 object dataset. Networks with a number label were trained with cross object loss and the number denotes the descriptor dimension. no-cross-object is a network trained without cross object loss.

descriptor space.

By applying cross-object loss (Section 3.3.3.i, training mode **specific** in Figure 3-3a), we can convincingly separate multiple objects such that they each occupy distinct subsets of descriptor space (Figure 3-5b). Note that cross-object loss is an extension of sampling *across scene* as opposed to only *within scene*. Given that we can separate objects in descriptor space, we next investigate: does the introduction of object distinctness significantly limit the ability of the models to achieve correspondence precision for each individual object? For multi-object datasets, we observe that there is a measurable decrease in correspondence precision for small-dimensional descriptor spaces when the cross-object loss is introduced, but we can recover correspondence precision by training slightly larger-dimensional descriptor spaces (Figure 3-5c). For the most part, 3-dimensional descriptor spaces were sufficient to achieve saturated (did not improve with higher-dimension) correspondence precision for single objects, yet this is often not the case for distinct multi-object networks.

3.5.3 Selective Class Generalization or Instance Specificity

Surprisingly we find that when trained simultaneously on similar items of a class using training mode **consistent**, the learned descriptors naturally generalize well across sufficiently similar instances of the class. This result of converging descriptors across a class is similar to the surprising generalization observed for human datasets in [16, 60]. Here we show that we can obtain class consistent dense descriptors for 3 different classes of objects (hats, shoes, and mugs) trained with only static-scene data association. We observe that the descriptors are consistent despite considerable differences in color, texture, deformation, and even to some extent underlying shape. The training requirements are reasonably modest – only 6 instances of hats were used for training yet the descriptors generalize well to unseen hats, including a blue hat, a color never observed during training. The generalization extends to instances that a priori we thought would be failure modes: we expected the boot (Figure 3-6h) to be a failure mode but there is still reasonable consistency with other shoes. Sufficiently different objects are not well generalized, however – for example Baymax and Starbot (Figure 3-2e,f) are both anthropomorphic toys but we do not attain

general descriptors for them. While initially we proposed research into further encouraging consistency within classes, for example by training a Dense Object Net to fool an instance-discriminator, the level of consistency that naturally emerges is remarkable and was sufficient for our desired levels of precision and applications.

For other applications, however, instance-specificity is desired. For example, what if you would like your robot to recognize a certain point on hat A as distinct from the comparable point on hat B? Although we could separate very distinct objects in multi-object settings as discussed in the previous section, it wasn't obvious to us if we could satisfactorily separate objects of the same class. We observe, however, that by applying the multi-object techniques (**specific** in Figure 3-3) previously discussed, we can indeed learn distinct descriptors even for very similar objects in a class (Figure 3-6iv).

3.5.4 Example Applications to Robotic Manipulation: Grasping Specific Points

Here we demonstrate a variety of manipulation applications in grasping specific points on objects, where the point of interest is specified in a reference image. We emphasize there could be many other applications, as mentioned in the Conclusion. In our demonstrations, a user clicks on just one pixel u_a^* in one reference image. Now the robot has the ability to autonomously identify the corresponding point in new scenes via Equation 3.6. Akin to other works with similarity learning in metric spaces [74], we set a simple threshold to determine whether a valid match exists. If a match is identified in the new scene we can instruct the robot to autonomously grasp this point by looking up the corresponding location in the point cloud and using simple geometric grasping techniques.

The particular novel components of these manipulation demonstrations are in grasping the visual corresponding points for arbitrary pixels that are either in different (potentially deformed) configurations (Fig. 3-6i-ii), general across instances of classes (Fig. 3-6iii), or instance-specific in clutter (Fig. 3-6iv). Our video[1] best displays these tasks. Note that only a dense (as opposed to sparse) method can easily accommodate the arbitrary selection

[1]See video (https://youtu.be/L5UW1VapKNE) for extensive videos of the different types of robot picking.

of interaction points, and class-generalization is out of scope for hand-designed descriptors such as SIFT. This is also out of scope for general grasp planners like [40,41,66,68] which lack any visual object representation, and for segmentation based methods [56,57,66] since the visual representation provided by segmentation doesn't capture any information beyond the object mask.

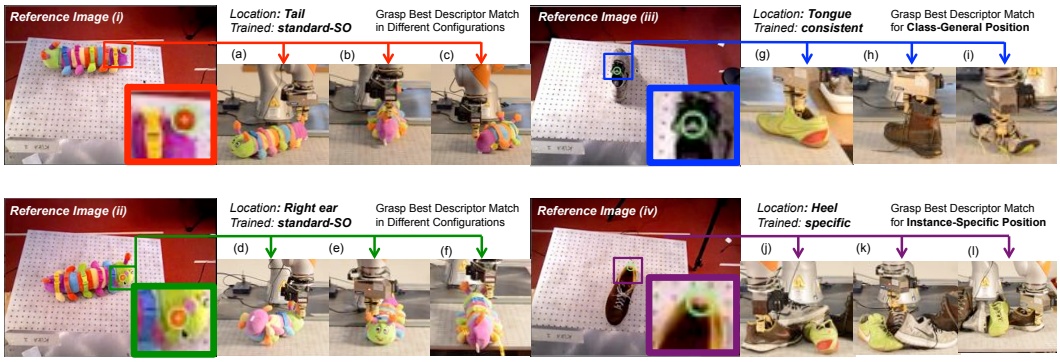


Figure 3-6: Depiction of "grasp specific point" demonstrations. For each the user specifies a pixel in a single reference image, and the robot automatically grasps the best match in test configurations. For single-object demonstrations, two different points for the caterpillar object are shown: tail (i) and right ear (ii). Note that the "right-ear" demonstration is an example of the ability to break symmetry on reasonably symmetrical objects. For class generalization (iii), trained with **consistent**, the robot grasps the class-general point on a variety of instances. This was trained on only 4 shoes and extends to unseen instances of the shoe class, for example (iii-i). For instance-specificity (iv) trained with **specific** and augmented with synthetic multi object scenes (3.3.3.iii), the robot grasps this point on the specific instance even in clutter.

3.6 Conclusion

This chapter introduces Dense Object Nets as visual object representations which are useful for robotic manipulation and can be acquired with only robot self-supervision. Building on prior work on learning pixel-level data associations we develop new techniques for object-centricness, multi-object distinct descriptors, and learning dense descriptors *by and for* robotic manipulation. Without these object centric techniques we found that data associations from static-scene reconstructions were not sufficient to achieve consistent object descriptors. Our approach has enabled automated and reliable descriptor learning at scale for a wide variety of objects (47 objects, and 3 classes). We also show how learned dense descriptors can be extended to the multi object setting. With new contrastive techniques we are able to train Dense Object Nets that map different objects to different parts of descriptor space. Quantitative experiments show we can train these multi object networks while still

retaining the performance of networks that do not distinguish objects. We also can learn class-general descriptors which generalize across different object instances, and demonstrated this result for three classes: shoes, hats, and mugs. Using class-general descriptors we demonstrate a robot transferring grasps across different instances of a class. Finally we demonstrate that our distinct-object techniques work even for objects which belong to the same class. This is demonstrated by the robot grasping a specific point on a target shoe in a cluttered pile of shoes. We believe Dense Object Nets can enable many new approaches to robotic manipulation, and are a novel object representation that addresses goals (i-iv) stated in the abstract. In future work we are interested to explore new approaches to solving manipulation problems that exploit the dense visual information that learned dense descriptors provide, and how these dense descriptors can benefit other types of robot learning, e.g. learning how to grasp, manipulate and place a set of objects of interest.

Chapter Acknowledgement

This chapter was joint work with Lucas Manuelli, and Russ Tedrake. We thank Duy-Nguyen Ta (calibration), Alex Alspach (hardware), Yunzhu Li (training techniques), Greg Izatt (robot software), and Pat Marion (perception and visualization) for their help. We also thank Tanner Schmidt for helpful comments in preparing the published paper corresponding to this chapter. This work was supported by: Army Research Office, Sponsor Award No. W911NF-15-1-0166; Draper Laboratory Incorporated, Sponsor Award No. SC001-0000001002; Lincoln Laboratory/Air Force, Sponsor Award No. 7000374874; Amazon Research Award, 2D-01029900; Lockheed Martin Corporation, Sponsor Award No. RPP2016-002. Views expressed in the paper are not endorsed by the sponsors.

Chapter 4

Dense Spatial-Temporal Distributional Correspondence Learning

4.1 Introduction

Correspondence is one of the most fundamental of computer vision tasks, underpinning the ability to infer the structure of the world. Recent powerful methods for correspondence learning have emerged in a variety of areas, including dense spatial correspondence models trained via novel pixel-to-pixel contrastive techniques [15, 76], dense 3D contrastive techniques [64], and temporal correspondence also trained via contrastive techniques [77]. Both spatial and temporal visual correspondence models can be learned in an entirely self-supervised fashion, no human annotation needed, by leveraging the inherent structure of high-dimensional data that is inherently spatio-temporal. These methods may be a powerful enabler for the future of artificial perception of geometry and beyond – for example enabling the ability for robots to fully autonomously learn dense visual correspondence [1].

In this chapter we introduce a novel formulation of correspondence learning on distributions. We separate detection from localization, and formulate our correspondence localization as a minimization of distributional divergence. Given our developed framework, a

As nicely recently quoted in [75], the story goes that Takeo Kanade once told a young graduate student that the three most important problems in computer vision are: "Correspondence, correspondence, correspondence!"

number of extensions become natural. Following the 2D case we extend into 3D, including an efficient factorized version, and then address occlusion handling and multi-sensor fusion. Temporal and then full spatio-temporal correspondence modeling also become natural extensions.

A primary motivation and application of this formulation is addressed in the subsequent chapter, in which we will use this computer vision method to allow robots to efficiently learn visuomotor policies. The success of the subsequent chapter is the primary result of the formulation in this chapter. While the previously-used contrastive loss formulation as in Chapter 3 provided impressive results, the new formulation is a significant step in terms of learning precise correspondences. Future work will further benchmark this approach against alternative methods.

4.2 Motivation

While self-supervised viusal correspondence learning presents an exciting, scalable route to learn the visual structure of the world, previous methods have been limited in a number of ways. For one, contrastive-loss-trained dense correspondence models can predict undesirably amodal correspondences, are highly sensitive to the relative scaling of match and non-match pairs, and existing paradigms do not easily separate *detection* of correspondences from *localization*, instead relying on hand-tuning thresholds for detection [1] and predicting delta distributions for localization. Additionally, dense visual correspondence models have been far from offering the correspondence precision of human-supervised template-based dense correspondence [79] or human-supervised keypoint detection [80, 81]. The high precision of these methods can directly translate, for example, into precision and interpretability for robots accomplishing tasks [82].

4.3 Background and Related Work

Self-Supervised and Generative Visual Learning. The methods presented in this chapter are perhaps most exciting in that they fit into the overarching theme of visual learning with

no human supervision. The broader topic of self-supervised visual learning has recently made exciting advances in areas such as identifying relative image patches [83], time correspondence learning [77], video prediction [84, 85], pose-supervised vision [86], monocular depth prediction trained from stereo images [87, 88] or structure-from-motion [89], and others. Relatedly, large advances have been made in generative models of entire images [90–94]. Some of the self-supervised methods exploit inherent structure freely available in the data, such as the structure of the images themselves, their timestamps and their sequence, while others may use reliable algorithms applied to full sequences, such as multi-view geometry [13,45,69,95]. Some methods can learn pixel-level tracking without spatial correlation data, for example using only time supervision [75, 77] or colorization learning [96]. In particular our work advances the line of dense image-to-image correspondence learning as developed in [1, 15, 60, 76].

Image-to-Image Pixel Correspondence. The aim of spatial correspondence between images is to find points in different images which correspond in a variously-defined meaningful way. In short, the history of image correspondence in computer vision has progressed as follows. At first, for example in the 1960s, humans provided 2D correspondences between images and computers solved the least squares estimation problems to solve for downstream tasks like 3D estimation. Throughout the latter half of the 20th century and into the 21st century, great advances were made in engineering highly robust sparse keypoint detection algorithms, for example SIFT [97], based on finding highly-discriminative points and corresponding them across images. A focus in research more recently have been to extend these methods to be more learning based, for example LIFT [98] and others [99]. Sparse, accurate keypoints are often sufficient for many problems, including camera pose estimation, and sparse [100, 101] or semi-dense [102] reconstruction.

More recently, fully dense learning methods have emerged [1,15,76], which fundamentally differ from the keypoint detection algorithms in that they do not aim to correspond discriminative points, but are trained to estimate correspondences between any image pixels, including for example areas of low texture which we may need to use full-image context in

See the video lecture "3D Computer Vision: Past, Present, and Future", by Steve Seitz, 2012, for a great overview of this history

order to infer the correspondence. Perhaps the most challenging problems are to find correspondences which differ dramatically in (i) viewpoint, (ii) lighting, (iii) scale, and/or (iv) context [103]. For some applications, such as for dense structure-from-motion (SFM) [95] and image editing [104], well-designed algorithms and simple metrics such as photometric distance may be remarkably successful between images that do not have too much variation in the mentioned challenges, (i-iv). Much progress has been made in learned depth-from-stereo [105, 106], which addresses dense correspondence in the small-baseline setting with remarkable results, but doesn't fundamentally address the mentioned challenges (i-iv). The fully dense correspondence learning methods differ, since they are capable of addressing those challenges (i-iv). Further, dense visual correspondence models, probably due to the information bottleneck in their final descriptor representation, have been shown to exhibit remarkable consistency across either deformable objects in different configurations or even across entire classes of objects [1, 76]. This extends dense descriptor learning into a separate challenge area, which is to identify dense correspondences *across scenes*. This problem was introduced originally by SIFT-Flow [14], which provides compelling results but cannot benefit from end-to-end learning of the full problem.

Keypoint Detection. A related but different set of problems is to detect a finite number of keypoints, where the goal is to learn an x to y mapping where x is an image and y is an ordered set of keypoints. A common problem is to find a supervised, labeled, set of keypoints [80, 81] either in 2D or 3D, which is highly applicable to problems such as human keypoint template regression, or any domain where applicable large human-labeled datasets exist. Some methods may also provide dense detection of a template by relying on a structured dense template [79]. While human-supervised methods can be highly precise, their utility must scale with human labeling effort, whereas self-supervised methods may scale with just data. Yet another related but different problem is to find an ordered set of keypoints, but where the keypoints are not provided but instead learned without direct supervision via a downstream supervised task [107].

3D Correspondence. Note also that a similar but parallel trend for correspondences has happened for learning on 3D representations, moving from engineered 3D descriptors [108] into fully learned 3D descriptors [64, 109]. Also when inference is performed over 3D,

then assumptions about the rigid or the regularized deformable structure of the world can provide correspondences by methods such as point set registration, including the popular ICP (Iterative Closest Point) algorithm [110]. There is a parallel between the methods in this chapter and probabilistic point-set registration such as CPD (Coherent Point Drift) [111] and related methods [112] which operate over a distribution of correspondences, rather than a assuming a one-to-one mapping as in ICP. In this chapter we pose our learning problem as learning from arbitrary multi-channel spatial-temporal data, for which a fused 3D representation is a special case.

Temporal Correspondence. A recent exciting line of work has worked on temporal correspondence learning [77, 113, 114]. These methods have been posed as metric learning frameworks via positive-negative samples provided by time-correlated data.

4.4 Contributions

This chapter contributes a novel generalization of space-time correspondence into a single unified learning framework on distributions. We formulate the localization problem for dense pixel-conditioned image-to-image correspondence learning over spatial distributions in 2D pixel space. We then extend this dense image-to-image correspondence learning into 3D, via pixel-conditioned image-to-image correspondence over known 3D distributions. We also formulate dense image-to-image correspondence learning into handling occlusions with either 2D or 3D learning and estimation, and novel extension of multi-camera inference over pixel-conditioned dense correspondences in 2D or 3D.

4.5 Formulation

The problem we are addressing is this: given an image, I, and another image I' and pixel coordinates in that image x', y', we would like to know: (i) is there (one or more) corresponding pixel coordinates in the image I, and if so then (ii) where is the corresponding pixel, or pixels?

This first question (i), is a *detection* problem. While detection is standard fare in many

computer vision tasks, for example object-level detection, and is performed in keypoint correspondence algorithms i.e. SIFT, to our knowledge we do not know of a prior work that has posed the *dense detection* problem as a learning task. It turns out that we can pose the dense detection problem as a standard pixelwise classification problem.

The second question (ii), is a localization problem, and is our primary interest in this work. In particular, we address a specific type of localization problem, which is:

$$p(x, y|I, I', x', y', \exists \text{ exactly 1 match in image } I \text{ for pixel } x', y' \text{ in image } I')$$

i.e., our estimation problem has already been conditioned on the assumption that there exists exactly one match we are looking for in image I. Given this scenario in which detection and has been handled upstream of localization, and detection has determined there is *exactly one* match, then we are able to create a valid probability distribution for the localization of this pixel match over the image, specifically $p(x, y|\cdot)$ becomes a valid probability distribution where the domain is the pixel space $[0 : W, 0 : H]$, for example for a W width and H height image.

How useful is this assumption, that something upstream of localization will be able to detect that there exists exactly 1 match? Specifically, what if instead there was 0 match? Or what if there was more than 1 match? This also of course depends on what we consider to be a match.

When we use as our definition of match that two pixels are only a match if their emanating rays intersect the world's geometry at exactly the same point, then it is clear that there cannot be more than 1 match, modulo absurd camera lenses and/or mirrors. Further, if the point is occluded or out of field-of-view, then this is something that we can easily determine given a dense 3D reconstruction of the world geometry, and a camera pose and camera model. So in this case where our match definition is explicitly spatial, then our assumption stands to be a good assumption, since there will never be more than 1 match in an image, and we can detect whether or not there is not a match.

The moment we start to use a broader definition of match, however, we must be careful. In particular, in the previous chapter, we considered corresponding points across instances

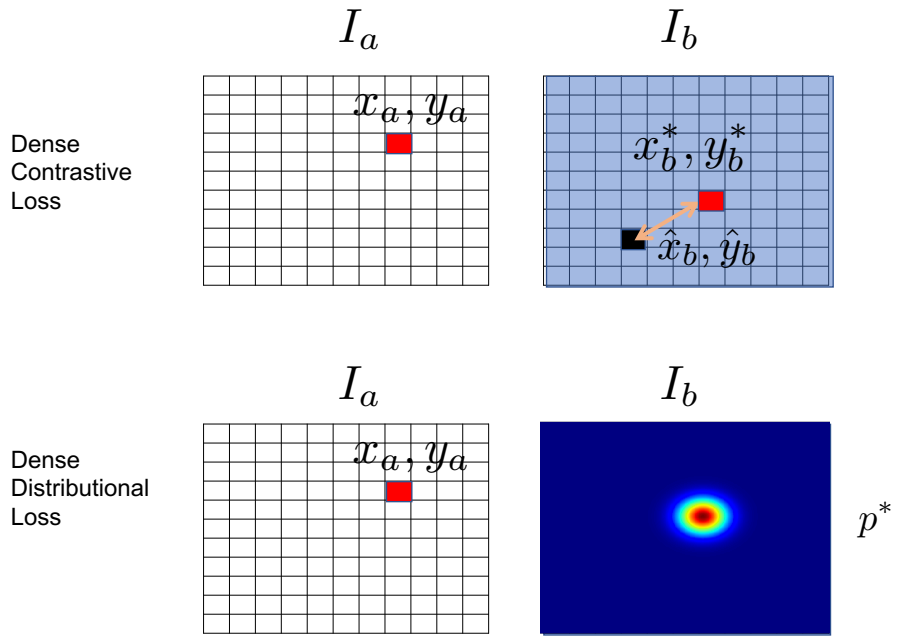


Figure 4-1: Comparison of dense distributional loss (this chapter) with dense contrastive loss (as in Chapter 3). To depict the comparison, a pair of simple low-resolution images, I_a and I_b are shown. With the contrastive loss, for some pixel x_a, y_a in image I_a, there is one ground truth correspondence x_b^*, y_b^* in image I_b, and the loss function implicitly has the shape such that the entire weight of the distribution is on the one discrete correct pixel (red), and everything else is equally wrong (blue). This loss does not take into account the error in pixel space (orange) for any given predicted correspondence $\hat{x}_b, \hat{y}_b$. In contrast with the distributional loss formulation, we consider "heat maps" p^* of the correspondence distribution in image I_b.

of a class. Were there to be two instances of a class in the same image, then $p(x, y|\cdot)$ as written above is not valid. Thankfully, we have solutions to the issue of multiple potential matches. One solution is to never show two instances of the same class in the same image during training. (This is what we will do in the subsequent chapter.) A less restrictive option would be to handle instance segmentation upstream of dense detection and localization, in which case $p(x, y|\cdot)$ could be applied to an image which was instance-segmented to have only one instance of the class that we expect to generalize across.

Our focus is in the *localization* formulation. The key concept is to formulate the localization problem with a probabilistic distribution, one that better leverages the spatial structure of the correspondence problem. A comparison with the previously-discussed contrastive loss is provided in Figure 4-1. To develop our formulation we will first review preliminaries (Section 4.5.1) in correspondence learning and probabilistic keypoint detection, and then discuss our approach (Section 4.5.2).

4.5.1 Preliminaries

First we provide background context and notation for prior image-to-image dense correspondence methods, and compare our formulation with dense correspondence learning based on sampled spatial-contrastive losses, time-contrastive losses, and with keypoint detection.

Following [1, 15, 76], dense image-to-image correspondence learning can be posed as the task of learning a dense descriptor mapping $f(\cdot)$ which maps a full-resolution RGB image $I \in \mathbb{R}^{W \times H \times 3}$, to a dense descriptor space $f(I) \in \mathbb{R}^{W \times H \times D}$, where for each pixel we have a D-dimensional descriptor vector. Training is performed in a Siamese fashion [74, 115, 116], where the input required is a pair of images, and many pixel-to-pixel matches between the images. A continuous pixel coordinate $[x_a, y_a] \in \mathbb{R}^2$ in image I_a is a match with pixel coordinate $[x_b, y_b] \in \mathbb{R}^2$ in image I_b if they correspond to pixels whose rays intersect the same point with the world's 3D geometry. Note that we can handle continuous-valued pixels by indexing into the image with interpolation. If desired, these pixel-level correspondences may be labeled by hand – alternatively, dense 3D reconstruction can automatically provide correspondences without human supervision [76], for example on the order of 10^{10} matches for even a small number of images and modest camera resolution of a scene. Prior works in deep dense correspondence learning [1, 15, 76] have primarily used pixelwise contrastive loss, as developed in [15, 76].

One view of the contrastive-loss-based correspondence approach is that the goal is to learn the delta function $p(x, y|I, I', x', y') = \delta[x^*, y^*]$ for the precise correspondence, and to train to do so via sparse sampling over the $WH - 1$ non-matches. If a predicted correspondence is to be wrong, we may prefer its error to be 0.3 pixels rather than 300.0 pixels, and the contrastive loss framework does not provide an effective way to do so. We have tried weighting match errors by their incorrect distance in pixel space or 3D space; we have not measured improved performance. Additionally, as can be seen in the measured performance in [1], the precision of contrastive-trained models can be highly sensitive to

A simple approximation is as follows: for a $W \times H$ image with a fraction overlap α between each image pair, and for N images each of which could be paired, we may have $WH \cdot \alpha \cdot N(N-1)$, i.e. this is 1.47×10^{10} for modest $W = 640$, $H = 480$, $N = 400$, $\alpha = 0.3$.

the relative weighting between positive-negative pairs. Also, although correspondence detection can be performed via choosing a hand-tuned descriptor norm threshold as in [1], this is somewhat arbitrary.

The methods of for example Time-Constrastive Networks [77] shares remarkable similarity to the pixelwise contrastive approaches, except the indexing is via temporal-wise-indexing into a video rather than pixel-wise-indexing into image. While a contrastive loss approach can successfully learn compelling time-descriptors with generalization across similar video sequences, it does share the similar traits of the pixelwise contrastive methods in that there is no separation of detection from localization, and in the original time-contrast framework it is unclear how to prefer correlation that is close in time but not far [77], i.e. 0.01 seconds error is not as bad as 10.0 seconds. Recent methods have aided this by adding classification over multiple time scales [113] and a Gaussian-shaped loss over the temporal dimension [114]. These methods also do not formulate how to address *detection*.

Motivation from Supervised Keypoint Detection

We are inspired by the popular method in keypoint detection to predict probability heatmaps of keypoint locations in either 2D or 3D [80, 81]. The aim to to regress a set of probability distributions ("heatmaps") for each keypoints $k = \{1, 2, ..., K\}$. In 2D we can consider these that methods aim to minimize the following loss function of a divergence between the predicted heatmap and the ground truth heatmap:

$$\mathcal{L}_k = D(p(x, y|I, k) \;||\; p^*(x, y|I, k)) \tag{4.1}$$

where $p^*(x, y|I, k)$ is the ground truth distribution for each supervised keypoint k, which in truth, we can consider this be a delta function at exactly $[x^{k*}, y^{k*}]$, but given the inherent human-labeling noise and measurement noise from camera imperfections, we can consider $p^*(x, y|I, k) \sim \mathcal{N}([x^*, y^*], \Sigma)$ with, for example a diagonal covariance with each σ = 1 pixel, as is used in [80] and other works. Various forms of these pixel distribution divergences, which we can also think of as "heat map losses" have been proposed and used in the keypoint detection literature, as summarized in [80]. Options include: the sum over

per-pixel norms, i.e. $\sum_{i,j} ||p(x,y) - p^*(x,y)||$, the KL-divergence over the distributions, $D_{KL}(p(x,y) \,||\, p^*(x,y)) = -\sum_{i,j} p(x,y) log\big(\frac{p^*(x,y)}{p(x,y)}\big)$, or any other divergence metric.

Another divergence metric can be based on the spatial expectation $[\hat{x}^k, \hat{y}^k]$ of the keypoint locations, which can then be compared with the ground-truth keypoint location $[x^{k*}, y^{k*}]$ via for example $||\hat{x}^k - x^{k*}||_p + ||\hat{y}^k - y^{k*}||_p$ for any $p = \{1, 2, ..., \infty\}$. The spatial expectation can be computed by, for example for $\hat{x}^k$:

$$\hat{x}^k = \mathbb{E}(x|I,k) \tag{4.2}$$

$$= \sum_{i,j} p(x_i, y_j|I,k) x_i, \tag{4.3}$$

where $x_i \in \mathbb{R}^1$ is the continuous coordinate in the width of the image, and $p(x_i, y_j|I,k)$ is the probability that the pixel coordinate $[x_i, y_j]$ is the predicted keypoint, i.e. obtained via the spatial softmax:

$$p(x_i, y_j|I,k) = \frac{e^{a_{i,j,k}}}{\sum_{i',j'} e^{a_{i',j',k}}} \tag{4.4}$$

where each $a_{i,j,k} \in \mathbb{R}^1$ is the activation of a row, column ordered neuron, for example in a deep convolutional network, for each keypoint k. The above was for $\hat{x}^k$, and an identical process may be repeated for $\hat{y}^k$ and for each $\hat{x}^{k'}, \hat{y}^{k'}$ additional keypoint. On a computer, this may all be implemented in a highly parallel fashion. Interestingly, a similar insight of converting an image-shaped set of activations into a continuous real-valued number via the spatial expectation was separately proposed in the end-to-end robot learning literature by [39] which has shown usefulness both for reinforcement [39] and imitation learning [78]. These methods can be interpreted as a way to regress a fixed set of keypoints, where the keypoints are learned for the purpose of aiding a downstream task, where in this case the downstream task is robot control.

4.5.2 Learning Correspondence Localization on Distributions

We assume we are given a large set of images and pixel-level correspondences between these images, and would like to learn a model of the form $p(x, y|I, x', y', I', M = 1)$, where $M = 1$ refers to that there is exactly 1 pixel-pixel match between the images. For notational

simplicity in the rest of this chapter, we do not include $M = 1$ as a written condition in each $p(\cdot)$, but it is implied. For each pair of images I_a and I_b, we consider a loss function which is independently summed over each set of pixel-to-pixel correspondences, each set is denoted as coordinate $[x_a, y_a]$ in image I_a corresponding as a match to pixel coordinate $[x_b, y_b]$ in image I_b,

$$\mathcal{L}(I_a, I_b) = \sum_{N_{\text{matches}}} \mathcal{L}_{match}(I_a, x_a, y_a, I_b, x_b, y_b) \tag{4.5}$$

For the loss associated with each matching pair $\mathcal{L}_{match}$, we can consider separately two terms which consider how the pixel in $[x_a, y_a]$ in I_a corresponds to the distribution of pixels in I_b, and vice versa to make the loss symmetric:

$$\mathcal{L}_{match} = \mathcal{L}_{a\to b} + \mathcal{L}_{b\to a} \tag{4.6}$$

For each of these terms, we can consider trying to minimize the divergence between the ground truth pixel distribution and our predicted distributions. We can consider general divergences between these two probability distributions:

$$\mathcal{L}_{a\to b} = D(p_b(x, y|I_b, I_a, x_a, y_a) \;||\; p_b^*(x, y|I_b)) \tag{4.7}$$

$$\mathcal{L}_{b\to a} = D(p_a(x, y|I_a, I_b, x_b, y_b) \;||\; p_a^*(x, y|I_a)) \tag{4.8}$$

Where $p_a^*(x, y|I_a)$ represents a ground truth pixel distribution for the point in image I_a. Note that unlike in the keypoint detection literature, as shown in Eq. 4.3, we cannot interpret $p_a(x, y|I_a, I_b, x_b, y_b)$ as the probability mass for predicting some fixed keypoint, but instead may interpret this as the probability mass that pixel $[x, y]$ in image I_a corresponds to the reference descriptor at pixel location $[x_b, y_b]$ in I_b. Note that for both comparisons, a probability distribution of correspondence has been induced, conditioned on a comparing image and a reference pixel in the image. Specifically, given image I_b and pixel $[x_b, y_b]$, this defines a conditional probability distribution of correspondence in image I_a, namely $p_a(x, y|I_a, I_b, x_b, y_b)$.

We can choose to minimize any divergence metric, for example the norm between the

expectations of the predicted pixel locations and the ground truth locations:

$$\mathcal{L}_{a\to b} = ||x_b - \hat{x}_b||_p + ||y_b - \hat{y}_b||_p \tag{4.9}$$

$$\mathcal{L}_{b\to a} = ||x_a - \hat{x}_a||_p + ||y_a - \hat{y}_a||_p \tag{4.10}$$

where the expectation for each of the four one-dimension pixel coordinates $\hat{x_a}, \hat{y_a}, \hat{x_b}, \hat{y_b}$ in the pair of images can be computed via the expectation, for example shown below for $\hat{x_a}$,

$$\hat{x}_a = \mathbb{E}(x_a|I_a, I_b, x_b, y_b) \tag{4.11}$$

$$= \sum_{i,j} p_a(x_i, y_j|I_a, I_b, x_b, y_b)x_i \tag{4.12}$$

The question then, is how to estimate these distributions p_a and p_b. To estimate them, we can introduce a nonparametric kernel $K\big(f(I)[x, y], f(I')[x', y']\big)$ over the $\mathbb{R}^D$ descriptor space to measure the similarity between descriptors. This nonparametric kernel can be thought of as simply weighting more heavily descriptors that are close together, as opposed to those that are far apart. Any function that maps $\{\mathbb{R}^D, \mathbb{R}^D\} \to \mathbb{R}^1$ should suffice although we would like at least the properties of (i) symmetry, i.e. $K(i, i') = K(i', i)$, (ii) $K(i, i) > K(i, i')$ i.e. two identical descriptors should be weighted more heavily than two different descriptors, and (iii) non-negativity. Note that this is a simple type of weighting function as used in nonparametric estimation, and is similar to the negative of a psuedo-distance in metric learning, although we do not require the additional psuedo-distance property of the triangle inequality [117]. The probability mass for each pixel can be computed via a kernel-weighted descriptor-similarity softmax as

$$p_a(x_i, y_j|I_a, I_b, x_b, y_b) = \frac{K\big(f(I_b)[x_b, y_b], f(I_a)[x_i, y_j]\big)}{\sum_{i',j'} K\big(f(I_b)[x_b, y_b], f(I_a)[x_{i'}, y_{j'}]\big)} \tag{4.13}$$

Note that this kernel is over the descriptor space, not the spatial space. It is the descriptor-space kernel which is used to provide a probability distribution over correspondences, and this estimated distribution is not inherently spatial. We may compute generic divergences

between this distribution and the ground truth distribution, as in Eqs. 4.7, 4.8. The overall loss function may then assume a spatial nature either by spatial assumptions on the ground truth pixel distribution, and/or via computing the spatial expectation as in Eq. 4.11.

For the choice of kernel function $K(\cdot,\cdot)$, various options are available. Perhaps the simplest is a negative norm between descriptors, i.e. $-||f(I)[x,y] - f(I')[x',y']||$. Below we use the Gaussian kernel over the squared l_2 norm, but other kernels could be used (i.e., uniform, triangular, Epanechnikov, Gaussian, etc.). For notation we define the squared l_2 norm between descriptors as $D(I,x,y,I',x',y') \triangleq ||f(I)[x,y] - f(I')[x',y']||_2^2$ and accordingly we obtain the probability mass for each pixel via kernel-weighted descriptor-similarity softmax,

$$p_a(x_i,y_j|I_a,I_b,x_b,y_b) = \\ = \frac{e^{-(D(I_b,x_b,y_b,I_a,x_i,y_i))^2}}{\sum_{i',j'} e^{-(D(I_b,x_b,y_b,I_a,x_{i'},y_{i'}))^2}} \tag{4.14}$$

Note that any normalization terms of the Gaussian kernels drop out when put into the softmax. It is interesting to take note that due to the softmax, then as intended the above equation will normalize any combination of many $\{K(\cdot,\cdot)...\}$ into a normalized probability distribution, and then applying Eq. 4.11 will always compute the expectation of a match for a given reference descriptor.

4.5.3 Extending Image-to-Image Correspondence Learning into 3D Spatial Distributions

A benefit of our dense correspondence localization framework is that it naturally extends into direct 3D inference, occlusion handling, and also naturally lends itself well to multi-sensor fusion and inference. A comparison of the 2D version with 3D versions is provided in Figure 4-2.

Consider the formation of the probability distribution over $p_a(x,y|I_a,I_b,x_b,y_b)$, which again is a probability distribution over the pixel space in image I_a, representing the probability that the pixel coordinates correspond to the some reference coordinate $[x_b,y_b]$ in

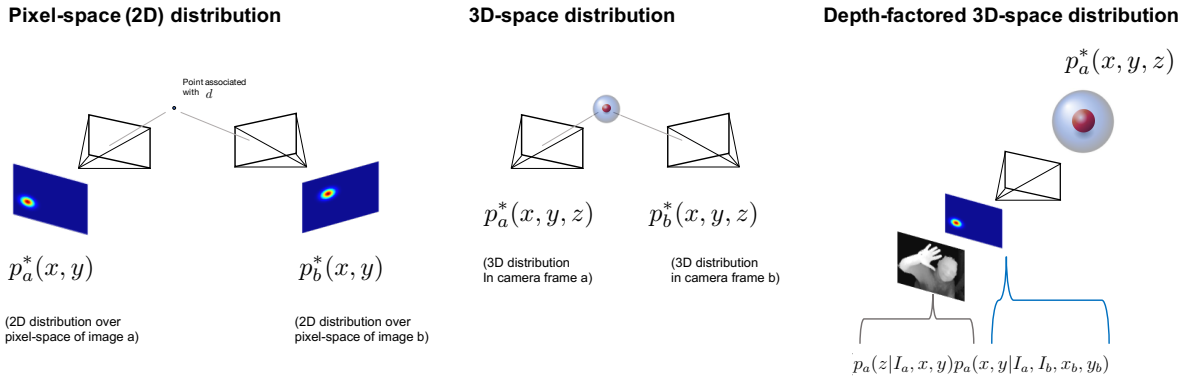


Figure 4-2: Depiction of different spatial formulations for which distributions to learn. In the 2D, pixel-space version (left), for a given point that is viewable from two views, we ask the correspondence model to learn pixel-space distributions centered at the pixel-coordinates to which the point projects. In 3D versions (middle), we ask the correspondence models to learn distributions over 3D space in the frame of the camera, such that the point is centered in these distributions. In our efficient depth-factored 3D version (right), we may use a known depth image, i.e. $p_a(z|I_a, x, y)$, together with a desired 3D distribution $p_a^*(x, y, z)$ to have the model learn the pixel-space distribution $p_a(x, y|\cdot)$ in a way that minimizes the 3D loss.

image I_b. This probability distribution is computed by a kernel-space softmax of the form in Eq. 4.13. As noted previously, the descriptor space kernel is not inherently spatial, but the formulation may become spatial via one of two ways: either assumptions about the ground-truth distribution, or by computing a spatial metric on the distribution such as a spatial expectation as in Eq. 4.11.

Estimation in 3D from 2D-only-trained Correspondence via a Known Depth Image

Due to this separation of the correspondence-kernel and the spatially-formulated-loss, we can naturally make our inference 3D while only having a correspondence model trained on pixel space. If for example we have the depth value of each pixel, as we may acquire from any type of dense depth estimation method, we may acquire the expected $\hat{z}_a$ in camera frame, where $z[x_i, y_j]$ denotes the depth measured at pixel $[x_i, y_j]$

$$\hat{z}_a = \mathbb{E}(z_a|I_a, I_b, x_b, y_b) \tag{4.15}$$

$$= \sum_{i,j} p_a(x_i, y_j|I_a, I_b, x_b, y_b) z[x_i, y_j] \tag{4.16}$$

The above, for example, even if we had only every trained with 2D correspondences between 2D images, would still allow us to compute the expectation of the depth correspond-

ing to a pixel in a reference image.

Training in 3D via a Known 3D Spatial Distribution

If however we also have ground truth depth values for each of the pixels in an image (or, practically, even just a significant fraction) during training time, we can train with with knowledge of the known 3D structure between images. While we will require dense RGBD images $\mathbb{R}^{W\times H\times 4}$ to enable the training, we may either train to do inference from either RGB or RGBD images. Again consider a loss function of the form Eq. 4.5 and Eq. 4.6, except now each of our divergence metrics may instead include probability distributions over the third z dimension as well.

$$\mathcal{L}_{a\to b} = D(p_b(x,y,z|I_b,I_a,x_a,y_a) \;||\; p_b^*(x,y,z|I_b)) \tag{4.17}$$

$$\mathcal{L}_{b\to a} = D(p_a(x,y,z|I_a,I_b,x_b,y_b) \;||\; p_a^*(x,y,z|I_a)) \tag{4.18}$$

If we know the depth value for each pixel $z[x,y]$, and we trust our depth estimation such that these depth values are independent of the reference image, then we can simply factor the inferred distributions to instead be

$$\begin{aligned}\mathcal{L}_{a\to b} = D\big(&p_b(z|I_b,x,y)p_b(x,y|I_b,I_a,x_a,y_a)\\ &||\; p_{I_b}^*(x,y,z)\big)\end{aligned} \tag{4.19}$$

$$\begin{aligned}\mathcal{L}_{b\to a} = D\big(&p_a(z|I_a,x,y)p_a(x,y|I_a,I_b,x_b,y_b)\\ &||\; p_{I_a}^*(x,y,z)\big)\end{aligned} \tag{4.20}$$

where we know $p(z|x,y)$, which we can approximate as a simple Gaussian centered at the measured depth mean, $\mathcal{N}(z[x,y],\sigma^2)$ with σ appropriately chosen for some depth sensor measurement noise. A more complicated distribution may also be assumed, for example one whose noise scales with inverse depth, as this is a more correct notion of uncertainty in multi-view geometry [118].

The above efficiently factorized framework presents a particularly exciting performance opportunity with little tradeoff in compute. Specifically, the $p_a(z|I_a,x,y)$ factor provides a

rigorous way to use our 3rd-dimension spatial data in order to properly weight the learning of the 2D spatial correspondence problem.

Just as we had formulated for pixel-space-only distributions, we may consider a variety of divergence metrics. Again, we may use an identical correspondence kernel in descriptor space as in Eq. 4.13. The learning becomes spatial either by adding a ground truth 3-dimensional distribution $p^*(x, y, z)$ or by computing an expectation over the predicted spatial distribution. Again we may also note that in the above learning scheme we may choose our $f(\cdot)$ to either accept the depth image, i.e. $I \in \mathbb{R}^{W \times H \times 4}$, or learn using only $I \in \mathbb{R}^{W \times H \times 3}$. Similarly in even our 2D pixel-space-only loss formulation we could have also chosen either RGB or RGBD as input, even if the loss was only made spatial in a 2D way.

Training for 3D Inference, Including Occlusion Handling

We may also extend our method into full 3D inference, for which a primary benefit is the ability to handle occlusions. Occlusion handling is a nice capability of 3D keypoint detection techniques [80], and to our knowledge has not been applied to a fully dense self-supervised domain.

By analogy let us consider an example state of the art 3D keypoint estimation framework. For each keypoint $k = 1, 2..., K$, a 3D heatmap may be estimated, $p^k(x, y, z)$. The heatmap is approximated by a 3D tensor, $H_k \in \mathbb{R}^{W \times H \times L}$, where L is the depth discretization. If there are K keypoints, this can be represnted by a $4D$ tensor, $H \in \mathbb{R}^{W \times H \times L \times K}$ Each heatmap H_k is ordered and paired with a ground truth heatmap for the labeled keypoint, for a small number k, and it is over this $4D$ tensor that the divergence metric can be computed.

By comparison, in the previous cases we considered for either 2D-only learning or depth-independence-factorized 3D training, we would have a dense descriptor space which is a 4D tensor $\mathbb{R}^{W \times H \times D \times N_m}$, where N_m is the number of samples.

To extend into the third dimension, the most straightforward way is to make our dense descriptor mapping to extend to creating a 5-dimensional tensor, $f(\cdot) \in \mathbb{R}^{W \times H \times L \times D \times N_m}$, where again L is the depth discretization, and D is the descriptor dimension. This becomes

a valid way to frame the learning problem, and equates to a descriptor-space kernel for which the softmax is applied over all correspondences in both width, height, and depth:

$$p_a(x_i, y_j, z_k | I_a, I_b, x_b, y_b) = \frac{K\big(f(I_b)[x_b, y_b, z_b], f(I_a)[x_i, y_j, z_k^{proj}]\big)}{\sum_{i',j',k'} K\big(f(I_b)[x_b, y_b, z_b], f(I_a)[x_{i'}, y_{j'}, z_{k'}^{proj}]\big)} \tag{4.21}$$

The primary difference is that during training, we do not discard any potential matches which become occluded in an alternative view, and we also note that every z_k^{proj} is the projection of z_b into the image I_a, rather than any depth value in the image I_a.

Multi-Sensor 3D Estimation

Continuing the benefits of our spatial 3D methods further, we can easily consider a multi-sensor fusion framework, where for online prediction of the expected location of a reference descriptor, we can use known physical camera parameters to fuse multiple camera predictions directly. This may be applied to a correspondence model which was trained either over 2D pixel space or 3D space during training time, as long as there is the ability to map to 3D space during test time. This may also be applied either with descriptor mappings trained to handle occlusions, or not. Given some reference descriptor $f(I_b)[x_b, y_b, (z_b)]$, where z_b is optional, in a single image, then if also given a set of images from a calibrated camera or cameras, and known poses $\{T_{\mathcal{W}\leftarrow 1}, T_{\mathcal{W}\leftarrow 2}, ...\}$ each $\in SE(3)$ in some common world frame $\mathcal{W}$, we may compute the expectation of this corresponding point in 3D in some

common world frame via

$$\hat{x}^{\mathcal{W}} = \mathbb{E}(x^{\mathcal{W}}|\{I_1, I_2, ...\}, I_b, x_b, y_b) \tag{4.22}$$

$$= \sum_n \sum_{i,j} p^{\mathcal{W}}_{\{\cdot\}}(x_{i,n}, y_{j,n}, z_n[x_i, y_j]|I_b, x_b, y_b) x^{\mathcal{W}}_{i,n} \tag{4.23}$$

$$\hat{y}^{\mathcal{W}} = \mathbb{E}(y^{\mathcal{W}}|\{I_1, I_2, ...\}, I_b, x_b, y_b) \tag{4.24}$$

$$= \sum_n \sum_{i,j} p^{\mathcal{W}}_{\{\cdot\}}(x_{i,n}, y_{j,n}, z_n[x_i, y_j]|I_b, x_b, y_b) y^{\mathcal{W}}_{j,n} \tag{4.25}$$

$$\hat{z}^{\mathcal{W}} = \mathbb{E}(z^{\mathcal{W}}|\{I_1, I_2, ...\}, I_b, x_b, y_b) \tag{4.26}$$

$$= \sum_n \sum_{i,j} p^{\mathcal{W}}_{\{\cdot\}}(x_{i,n}, y_{j,n}, z_n[x_i, y_j]|I_b, x_b, y_b) z^{\mathcal{W}}_n[x_i, y_j] \tag{4.27}$$

The simplest way to interpret the above is by considering the expectation over a fused 3D point cloud. For simplicity we have written the 3D points $x_{i,n}, y_{i,n}, z_n[x_i, y_j]$ which are the points for each pixel $[x_i, y_j]$ in each image I_n as 3D points which have already been transformed via the camera's intrinsic calibration and known camera pose into a 3D point in world frame, where the superscript $\mathcal{W}$ denotes the frame. For brevity above is written $p_{\{\cdot\}}$, this is really a correspondence distribution over all images, i.e. $p_{\{I_1, I_2, ...\}}$.

The above formulation represents a few exciting properties: for one, for arbitrary numbers of cameras we can still use the same correspondence model and kernel function as we used before during training in order to provide the per-pixel correspondence weighting, and we can incorporate fusion of N cameras in an efficient $\mathcal{O}(N)$ framework which only adds an additional summed term for each camera.

Perhaps the most exciting capability, however, offered by multi-sensor fusion is that our conditional probability distribution is less constrained than in the single-image case: specifically, it is over the full set of fused images, rather than just a given single image. Accordingly when estimating this probability function we may incorporate all of these

terms into the denominator of the softmax:

$$p_{\{I_1,I_2,...,I_N\}}(x_{i,n}, y_{j,n}, z_n[x_i, y_j] \mid I_b, x_b, y_b) = \frac{K\big(f(I_b)[x_b, y_b], f(I_n)[x_i, y_j]\big)}{\sum_{n'} \sum_{i',j'} K\big(f(I_b)[x_b, y_b], f(I_{n'})[x_{i'}, y_{j'}]\big)} \tag{4.28}$$

Where above is written the known 3D version, $f(\cdot) \rightarrow \mathbb{R}^{W \times H \times D}$, but the same can also be written for the 3D inference version, $f(\cdot) \rightarrow \mathbb{R}^{W \times H \times L \times D}$.

4.5.4 Correspondence Learning on Temporal Distributions

Following a similar framework we have developed for spatial correspondence, we can write down our same framework over time. Consider two video sequences V_a and V_b each in $\mathbb{R}^{T \times W \times H \times C}$, where each video is a set of timestamped frames, i.e. $V = \{(I_0, t_0), (I_1, t_1), ...(I_T, t_T)\}$, then we can pose the frame-conditioned video-to-video correspondence problem as a localization problem of the form

$$p_a(t|t_b, V_a, V_b) \tag{4.29}$$

where the time-indexing $V_b[t_b]$ indexes into a specific image at t_b in the video V_b. Note that for a visibly monotonically-changing video-sequence, then we would expect $p_a(t|t_b, V_a, V_b)$ to be a unimodal distribution, but due to the ambiguities in sequences of videos for example due to any redundant motion, the distribution that we would guess as humans corresponding two image sequences will not always be unimodal.

In order to align over times, what we would like to learn is a temporal correspondence mapping $f_t(\cdot)$, which maps a sequence of images $V = (I_0, I_1, I_2, ..., I_T) \in \mathbb{R}^{T \times W \times H \times C}$ to a temporal descriptor space, $\mathbb{R}^{T \times D}$, where for each time T we have a discriminative descriptor.

Using Explicit Temporal Distributions

Similar to the spatial case, we can consider minimizing the divergence between the ground truth alignment distribution and an inferred distribution, for example in the first the frame-

conditioned and then the unconditioned case

$$\mathcal{L} = D(p(t^a|t_b, V_a, V_b) \;||\; p^*(t^a|t_b, V_a, V_b)) \tag{4.30}$$

$$\mathcal{L} = D(p(t^a, t^b|V_a, V_b) \;||\; p^*(t^a, t^b|V_a, V_b)) \tag{4.31}$$

If we would like to choose an explicit temporal distribution, we may for example use a simple univariate Gaussian

$$p^*(t^a|t_b, V_a, V_b) = \mathcal{N}(t_b^*, \sigma) \tag{4.32}$$

where σ can for example be the error in the relative sensor time calibration. Similar to the spatial case, we can choose a variety of divergence metrics, and as in our spatial learning framework, we can estimate our probability distribution via a correspondence kernel:

$$p(t_i^a|V_a, V_b, t_b) = \frac{K\big(f_t(V_b)[t_b], f_t(V_a)[t_i^a]\big)}{\sum_{i'} K\big(f_t(V_b)[t_b], f_t(V_a)[t_{i'}^a]\big)} \tag{4.33}$$

In the temporal learning case during optimization we sample many times and form heatmaps over the other distributions, i.e. during learning our models map to a 3-dimensional tensor $\mathbb{R}^{T\times D\times N_m}$ where N_m is the number of sampled matches per gradient step.

4.5.5 Correspondence Learning Across Space-Time

At this point, since we have a fully general framework for both spatial and temporal generative correspondence learning, it is trivial to extend this to space-time correspondence learning. We can aim to learn a mapping $f_{s-t}(\cdot)$, which maps a sequence of images $V = (I_0, I_1, I_2, ..., I_T) \in \mathbb{R}^{T\times W\times H\times C}$ to a dense spatial-temporal descriptor space. In the pixel-spatial-temporal case the mapping is to a space $\mathbb{R}^{T\times W\times H\times D}$, where for each point in time, and for each pixel we have a D-dimensional descriptor. In the 3D-spatial-temporal case the mapping is to a space $\mathbb{R}^{T\times W\times H\times L\times D}$. During training we accordingly perform descriptor-kernel softmaxes over either a tensor of shape $\mathbb{R}^{T\times W\times H\times D\times N_m}$

or $\mathbb{R}^{T \times W \times H \times D \times N_m}$, i.e. 5-D or 6-D tensors.

Although it is unclear what is the primary application of a spatial-temporal correspondence model, this fits naturally into our framework with essentially no modifications. And after all, spatial-temporal correspondence is a task that humans are able to do, for example this of course is a critical task in forensics.

4.5.6 Dense-to-Sparse Correspondences

For a number of practical reasons though it may be nice to have a visual representation of the world $\boldsymbol{z}$ which is smaller-dimensional that dense fields of descriptors, and instead have a sparse representation of only select points in the world. Reasons include: (i) in particular this can make predictions through occlusion significantly easier, and (ii) for control we may like to have an approximate state space which moves continuously. This does fundamentally change our problem – the difference is that we know what we are looking for. In the dense case, we must assign to each pixel some type of descriptor which is useful for comparing it against every other pixel. In the sparse problem, we just need to decide where is the pixel we are looking for.

Given the ability to perform dense correspondence, it is of course easy to subsample to dense correspondences. The first step is to have some type of initialization procedure. We can then come up with some descriptor set $\{\boldsymbol{d}_i\}_{i=1}^{P}$, which is just a set of P descriptors, with each $d_i \in \mathbb{R}^D$. We can now essentially consider our problem a keypoint problem.. i.e., we would like,

$$p(x_i, y_j | I, k) = \frac{e^{a_{i,j,k}}}{\sum_{i',j'} e^{a_{i',j',k}}} \tag{4.34}$$

To give a fixed number of heatmaps for each point, but where these sparse feature points are defined only as being sampled from a dense correspondence field.

$$p(x_i, y_j | I, d_k) = \frac{e^{K(d_{ij}, d_k)}}{\sum_{i',j'} e^{K(d_{i'j'}, d_k)}} \tag{4.35}$$

We have now transformed our dense descriptor mapping into a sparse mapping, i.e.:

$$f_d : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^{W \times H \times D} \tag{4.36}$$

$$f_{sd} : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^{W \times H \times D} \rightarrow \mathbb{R}^{W \times H \times K} \tag{4.37}$$

Where we now have K heatmaps but we haven't added any parameters to our model.

4.6 Conclusion

We have developed a novel framework for performing dense correspondence learning. Our primary result is presented in the next chapter, which uses this formulation to perform efficient imitation learning.

Compared with the dense pixelwise contrastive loss as discussed in the previous chapter, the dense distributional loss yields learned correspondence models which are more desirably precise and uni-modal. The loss is more aligned with what we actually want, which is that if predicted correspondences are "wrong", they should be close rather than far from the correct correspondence. The distributional framework also enables a number of capabilities leveraging 3D information and multi-sensor fusion.

To summarize briefly, from a broad perspective, what is the capability offered by this approach: the capabilities of this method are similar to the approach presented in Chapter 3, although with a handful of differences. In terms of what is the same: we may take multi-view data, preferably of one object at a time, and learn correspondence models for those objects. The new approach presented in this chapter, however, will produce correspondence models that are more precise, with uni-modal correspondences, may leverage 3D information, and may enable fused inferences from many views. While we did not discuss multi-object model concepts such as cross-object loss, this fits into our new framework at the *detection* stage – if two objects are different, they should not have any detected correspondences between them, and there is no correspondence localization.

Chapter Acknowledgement

This chapter was joint work with Lucas Manuelli, Wei Gao, and Russ Tedrake.

Chapter 5

Self-Supervised Correspondence in Visuomotor Policy Learning

5.1 Introduction

For robots to achieve general-purpose manipulation skills, they will need to employ closed-loop real-time manipulation policies and be able to scalably achieve new manipulation skills without excessive incremental human effort per task. We have seen in our previous work (Chapter 3) that we are able to make a robot entirely self-supervise remarkably consistent dense visual models of objects. In this prior work, however, we did not attempt to take advantage of this ability to form closed-loop policies and address a diversity of tasks, which we now consider. To train closed-loop policies, other prior work has often used visual training methods to such as end-to-end training [39], autoencoding [119], and pose-based losses [39, 78]. These methods, however, have not benefitted from the rich sources of self-supervision that may be provided by dense three-dimensional computer vision techniques [76, 89, 120], for example correspondence learning which robots can automate without human input [1].

Correspondence is fundamental in computer vision, and we believe it has fundamental usefulness for robots learning complex tasks requiring visual feedback. We introduce using self-supervised correspondence for visuomotor policies, and our results suggest this enables policy learning that is surprisingly capable. Our evaluations pair correspon-

dence training with a simple imitation learning objective, and extensive hardware validation shows that learned policies can address challenging scenarios: manipulating deformable objects, generalizing across a class of objects, and visual challenges such as textureless objects, clutter, moderate occlusion, and lighting variation. Learning these policies may take as few as 50 demonstrations. Additionally our simulation-based comparisons empirically suggest that our method offers significant generalization and sample complexity advantages compared to existing methods for training visuomotor policies, while requiring no additional human supervision.

Briefly as well we consider the scope our method – when is it applicable, and when is it not? We believe that the approach of self-supervising correspondence to aid visuomotor policy learning is applicable when there is *informative local spatial visual coherence*. That is to say, if the way something looks (its visual texture and appearance), is at least somewhat coherent in its different configurations and provides information about its state, then correspondence training can be useful. An example that would be difficult for spatial correspondence alone to address would be deciding when to be finished cooking eggs, since there is not local spatial visual coherence. We would propose that whether or not the egg is finished can also be posed as a correspondence problem (to the "perfect time to stop cooking the egg"), but the scale of the correspondence is larger than the dense spatial correspondences that we train in this work.

5.2 Related Work

There are a number of primary related areas of work to the contributions presented in this chapter. Since previous chapters of this thesis have addressed related work in the areas of self-supervised robotics (see Chapter 3) and correspondence learning (see Chapter 4), we instead will focus this chapter's related work around two topics: visual training methods for visuomotor policies, and approaches for providing the policy learning signal.

5.2.1 Visual Training Methods for Visuomotor Policies

There have been three primary methods used in the robot learning literature to train the visual portion of visuomotor policies. Often these methods are used together – for example [39, 78] use pose-based losses together with end-to-end training.

1. *End-to-End training.* This approach can be applied to any learning signal that is formed as a consequence of a robot's actions, for example through imitation learning or reinforcement learning. While often end-to-end training is complemented with other learning signals, other works use purely end-to-end training.

2. *Autoencoders.* Autoencoding can be applied to any data with no supervision and is commonly used to aid visuomotor policy learning [119, 121–125]. Sometimes polices are learned with a frozen encoder [119, 121, 122], other times in conjunction with end-to-end training [125].

3. *Pose-based losses.* In [39], for example, a separate dataset is collected of the robot holding objects, and assuming that the objects are rigid and graspable, then using the robot's encoders and forward kinematics the visual model can be trained to predict the object pose. In [78], pose-based auxiliary losses are used regardless of whether or not objects are held – we wouldn't expect this to learn how to predict object configurations unless they are also rigid and grasped. Simulation-based works [126] have also used auxiliary losses for object and gripper positions.

In our comparison experimentation, we include end-to-end training and autoencoding, but not pose-based losses, since they are not applicable to deformable or un-graspable objects. While the above are three of the most popular, other visual training methods include: training observation dynamics models [19, 127], using time-contrastive learning [77], or using no visual training and instead using only generic pre-trained visual features [128]. Relevant concurrent works include [129] which proposes autoencoder-style visual training but with a reference image and novel architecture, and [130] which proposes a graph-based reward function using a fixed set of correspondences.

5.2.2 Methods for Learning Vision-Based Closed-Loop Policies

While the previous section discussed visual training methods, to acquire policies they must be paired with a policy learning signal. We are particularly interested in approaches that can (i) scalably address a wide variety of tasks with potentially deformable and unknown objects, (ii) use a small incremental amount of human effort (on the order of 1 human-hour) per each new object or task, and (iii) produce real-time vision-based closed-loop policies.

One source of policy learning signal may be from reinforcement learning, which has demonstrated many compelling results. A primary challenge, however, is the difficulty of measuring rewards in the real world. Some tasks such as grasping can be self-supervised [68], and other tasks can leverage assumptions that objects are grasped and rigid in order to compute rewards [39], but this only applies to a subset of tasks. A more generalizable direction may be offered by unsupervised methods of obtaining reward signals [77, 119, 128]. Another direction which has shown promising results is using sim-to-real transfer [126, 131–133], but our interest in a small amount of incremental human effort per new task is challenging for these methods, since they currently require significant engineering effort for each new simulation scenario.

Another powerful source of signal may come from imitation learning from demonstration, which several recent works have shown promise in using to produce real-time vision-based closed-loop policies [24, 77, 78, 121, 122, 125, 128, 134]. We point the reader to a number of existing reviews of learning from demonstration [21, 22]. Another direction may be to learn models from observations and specify goals via observations [19, 119, 127], but these may be limited to tasks for which autonomous exploration has a reasonable chance of success.

In terms of limitations of these prior works, one primary challenge relates to reliability and sample complexity – it is not clear how much data and training would be required in order to achieve any given level of reliability. Relatedly, a second limitation is that little work has characterized the distributions over which these methods should be trained and subsequently expected to generalize. Third, like in many areas of robotics it is difficult to reproduce results and compare approaches on a common set of metrics. While we believe

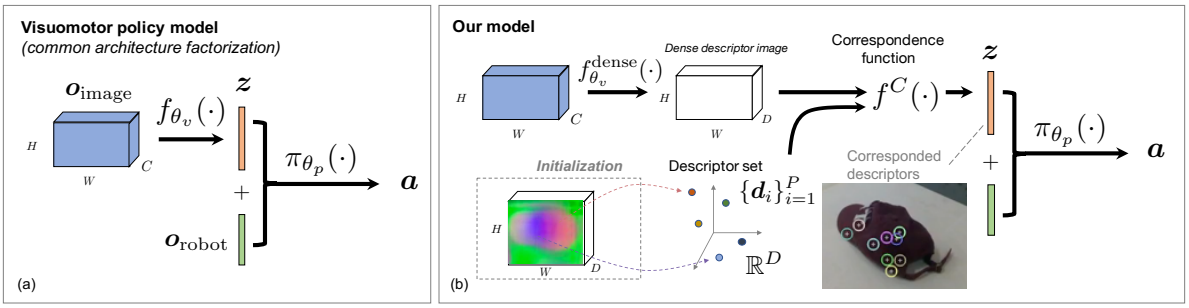


Figure 5-1: Diagram of common visuomotor policy factorization (a), and our proposed model (b) using visual models trained on correspondence.

hardware validation is critical, we also believe that increased effort should be put into simulation-based results that compare methods and can be reproduced.

5.3 Visuomotor Formulation

We propose a formulation of visuomotor policy learning which leverages self-supervised correspondence. First as preliminary we identify some primary attributes of existing visuomotor policy learning approaches, and then present our approach based on self-supervised correspondence.

5.3.1 Preliminary: Visuomotor Policies

We would like to have a policy $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{o}_{0:t})$, where $\boldsymbol{o}_{0:t} = (\boldsymbol{o}_0, \boldsymbol{o}_1, \ldots \boldsymbol{o}_t)$ is the full sequence of the robot's observations during some episode up until time t, with each $\boldsymbol{o}_i \in \mathcal{O}$, the robot's observation space. This sequence of observations is mapped by $\pi_\theta(\cdot)$, the robot's policy parameterized by θ, to the robot's actions $\boldsymbol{a}$, each $\in \mathcal{A}$. In particular, we are interested in visuomotor policies in which the observation space contains high-dimensional images $\mathcal{O}_{\text{image}} \subset \mathcal{O}$, for example $\mathcal{O}_{\text{image}} = \mathbb{R}^{W \times H \times C}$ for a C-channel, width W, and height H image. The visual data is perhaps complemented with additional lower-dimensional measurements $\mathcal{O}_{\text{robot}}$, such as produced from sensors like the robot's encoders, such that $\mathcal{O}_{\text{robot}} \times \mathcal{O}_{\text{image}} = \mathcal{O}$.

It is common for a visuomotor policy to have an architecture that can factored as dis-

played in Fig. 5-1(a),

$$\boldsymbol{z} = f_{\theta_v}(\boldsymbol{o}_{\text{image}}) : \boldsymbol{o}_{\text{image}} \in \mathcal{O}_{\text{image}},\ \boldsymbol{z} \in \mathbb{R}^Z \tag{5.1}$$

$$\boldsymbol{a} = \pi_{\theta_p}(\boldsymbol{z}, \boldsymbol{o}_{\text{robot}}) : \boldsymbol{o}_{\text{robot}} \in \mathcal{O}_{\text{robot}},\ \boldsymbol{z} \in \mathbb{R}^Z,\ \boldsymbol{a} \in \mathcal{A} \tag{5.2}$$

in which a visual model $f_{\theta_v}(\cdot)$, parameterized by θ_v, processes the high-dimensional $\boldsymbol{o}_{\text{image}}$ into a much smaller Z-dimensional representation $\boldsymbol{z}$. The policy model $\pi_{\theta_p}(\cdot)$ then combines the output of the visual model with other observations $\boldsymbol{o}_{\text{robot}}$. This is a practical modeling choice – images are extremely high dimensional, i.e. in this work we use images in $\mathbb{R}^{640\times480\times3} = \mathbb{R}^{921{,}600}$, whereas our $\mathcal{O}_{\text{robot}}$ is at most $\mathbb{R}^{13}$. A wide variety of works have employed a similar architecture to [39], consisting of convolutional networks extracting features from raw images into an approximately $Z = 32$ to 100 bottleneck representation of features, e.g. [24, 78, 119, 123, 125, 135]

5.3.2 Visual Correspondence Models for Visuomotor Policy Learning

The objective of the visual model is to produce a feature vector $\boldsymbol{z}$ which serves as a suitable input for policy learning. In particular, we are interested in deploying policies that can operate directly on RGB images. Given the role that pose estimation has played in traditional manipulation pipelines it seems valuable to encode the configuration of objects of interest in the vector $\boldsymbol{z}$. Pose estimation, however, doesn't extend to the cases of deformable or unknown objects. Some of the prior works discussed in Sec. 5.2.1, for example [39, 119], have interpreted their learned feature points $\boldsymbol{z}$ as encoding useful spatial information for the objects and task. These feature points are learned via the supervisory signals of end-to-end, pose-based, or autoencoding losses, and don't explicitly train for spatial correspondence. In contrast our approach is to directly employ visual correspondence training, building off the approach of [1] which can in a self-supervised manner, learn pixel descriptors of objects that are effective in finding correspondences between RGB images.

We introduce four different methods for how to employ dense correspondence models as the visual basis of visuomotor policy learning. The first three are based on the idea of a set of points on the object(s) that are localized either in image-space or 3D space. We

represent these points as a set $\{\boldsymbol{d}_i\}_{i=1}^{P}$ of P descriptors, with each $\boldsymbol{d}_i \in \mathbb{R}^D$ representing some vector in the D-dimensional descriptor space produced by a dense descriptor model $f_{\theta_v}^{\text{dense}}(\cdot)$. This $f_{\theta_v}^{\text{dense}}(\cdot)$, a deep CNN, maps a full-resolution RGB image, $\mathbb{R}^{W\times H\times 3}$, to a full-resolution descriptor image, $\mathbb{R}^{W\times H\times D}$. Let us term $f^C(\cdot)$ to be the non-parametric correspondence function that, given one or more descriptors and a dense descriptor image $f_{\theta_v}^{\text{dense}}(\boldsymbol{o}_{\text{image}})$, provides the predicted location of the descriptor(s):

$$\boldsymbol{z} = f^C\left(f_{\theta_v}^{\text{dense}}(\boldsymbol{o}_{\text{image}}), \{\boldsymbol{d}_i\}_{i=1}^{P}\right) \tag{5.3}$$

Specifically $f^C : \mathbb{R}^{W\times H\times D} \times \mathbb{R}^{P\times D} \rightarrow \mathbb{R}^{P\times K}$, where $K = 2$ corresponds to: $\boldsymbol{z}$ is the predicted corresponding (u, v) pixel coordinates of each descriptor in the image, while $K = 3$ is their predicted 3D coordinates. All four methods optimize a generic policy-based loss function, shown in Eq. (5.4), and vary only in the set of learnable parameters Θ and how $\boldsymbol{z}$ is acquired (the first three use Eq. 5.3). This loss function $\mathcal{L}$ is generic and could represent any approach for learning the parameters of a visuomotor policy.

$$\min_{\Theta} \; \mathcal{L}\left(\pi_{\theta_p}\left(\boldsymbol{z}, \boldsymbol{o}_{\text{robot}}\right)\right) \tag{5.4}$$

Fixed Descriptor Set. This method only optimizes the policy parameters, $\Theta = \{\theta_p\}$. In this case both the set of descriptors $\{\boldsymbol{d}_i\}_{i=1}^{P}$ and visual model $f_{\theta_v}^{\text{dense}}(\cdot)$ are fixed. We use a simple initialization scheme of sampling $\{\boldsymbol{d}_i\}$ from a single masked reference descriptor image. While we have found this method to be surprisingly effective, it is unsatisfying that the visual model's representation is not optimized after the random initialization process.

Descriptor Set Optimization. This method optimizes the descriptor set $\{\boldsymbol{d}_i\}_{i=1}^{P}$ along with the policy parameters θ_p while keeping the dense descriptor mapping $f_{\theta_v}^{dense}$ fixed. Intuitively $f_{\theta_v}^{dense}$ has already been trained to perform correspondence, and we are simply allowing the policy optimization to choose *what to correspond*. We have observed that Descriptor Set Optimization can improve validation error in some cases over a Fixed De-

The specific form of $f^C(\cdot)$ is defined by how the correspondence model was trained. In our preferred model we compute a spatial-expectation using a correspondence kernel, either in image-space or 3D. See Chapter 4 for details.

scriptor Set, and adds minimal computational cost and parameters.

End-to-End Dense Optimization. The third option is to train the full model architecture end-to-end by including θ_v in the optimization. While we may have expected this approach to allow the visual model to more precisely focus its modeling ability on task-critical parts of images, we so far have not observed a performance advantage of this approach over Descriptor Set Optimization.

End-to-End with Correspondence Pretraining. The fourth option is to directly apply a differentiable operation to a model which was previously trained on dense correspondence. We can apply any differentiable operation $g(\cdot)$ on top of $f_{\theta_v}^{\text{dense}}$ directly to produce a representation $\boldsymbol{z} = g\big(f_{\theta_v}^{dense}(\boldsymbol{o}_{\text{image}})\big)$. For example, we can apply non-parametric channel-wise spatial expectations to each of the D channels of the dense descriptor images. The optimization variables in this case are $\Theta = \{\theta_p, \theta_v\}$.

For our $f_{\theta_v}^{\text{dense}}$ we use a 34-layer ResNet, as in [1], which is a powerful vision backbone. Accordingly, using either a fixed- or optimized- descriptor set will significantly increase policy training speed, since it does not require forward-backward optimizing through a very deep convolutional network in each step of policy training, which in our case is 1 to 2 orders of magnitude faster.

5.4 Visual Imitation Formulation

We now propose how to use the general approach of the previous section for a specific type of imitation learning for robot manipulation.

5.4.1 Imitation Learning Visuomotor Policies

To evaluate visual learning strategies for enabling visuomotor policy learning, we use imitation learning via a simple behavioral cloning [136] strategy, which a few recent works have demonstrated to be viable for learning visuomotor manipulation policies [78, 125]. Optimizing a policy with parameters Θ on the behavioral cloning objective, given a dataset of N_{train} trajectories of observation-sequence-to-action pairs $\{(\boldsymbol{o}_t, \boldsymbol{a}_t^*)\}_{t=0}^{T_i}$ can be written

as:

$$\min_{\Theta} \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \sum_{t=0}^{T_i} \mathcal{L}_{\text{BC}}\big(\boldsymbol{a}_t^*, \pi_\Theta(\boldsymbol{o}_t)\big) \tag{5.5}$$

For our loss function we use a simple weighted sum of l_1 and l_2 loss, $\mathcal{L}_{\text{BC}}(\cdot) = ||\boldsymbol{a}^* - \pi(\cdot)||_2^2 + \lambda||\boldsymbol{a}^* - \pi(\cdot)||_1$ where we use $\lambda = 0.1$. We scale $\boldsymbol{a}^*$ to equalize 1.0m end-effector translation, 0.1 radians end-effector rotation, and 1.0m gripper translation.

5.4.2 Robot Observation Space, Action Space, and Interface to Low-level Control

A primary interest of this work is in how to process the image portion of the observation space $\mathcal{O}_{\text{image}}$, but here we also consider the rest of the observation space $\mathcal{O}_{\text{robot}}$. Our intuition matches the results demonstrated in recent work [137] that, for a robot that has accurate encoders and the ability to perform effective inverse-kinematics based control, end-effector space is a good space for learning policies. For the translational component of our observation space, we can simply use some type of measurement of 3D position of the end-effector in world frame. Rotation, however, is more complicated to represent. We experimented with several different rotational observation options. We settled on a redundant over-parameterized representation, part of which is the location of three 3D points on the hand in world frame as in [39], and the other part of which is an axis-angle representation relative to the end-effector's nominal pose for the task. We found this representation to perform at least as well as any other representation we tried, including only the 3D points, only the axis-angle, and/or some mix of these with quaternions and euler angles. Accordingly, our full non-vision observation comprises:

$$\text{(all non-vision observations)} \;\; \boldsymbol{o}_{\text{robot}} = (\boldsymbol{o}_{\text{end-effector}}, o_{\text{gripper}}) \tag{5.6}$$

$$\text{(end-effector frame observations)} \;\; \boldsymbol{o}_{\text{end-effector}} = ({}^W p_0, \; {}^W p_1, \; {}^W p_2, [a_x, a_y, a_z]) \tag{5.7}$$

where ${}^W p_i \in \mathbb{R}^3$ for $i = \{0, 1, 2\}$ are the three points on the end effector expressed in world frame and $[a_x, a_y, a_z] \in \mathbb{R}^3$ is a vector whose direction and magnitude represents the axis-angle representation relative to the nominal pose. Note that in full $\boldsymbol{o}_{\text{end-effector}}$ comprises a

12-dimensional redundant observation space of what is only 6 degrees of freedom. For the gripper, $o_{\text{gripper}} \in \mathbb{R}^1$ is just the single position measurement of the gripper width.

For the action space, we choose our policy to predict a desired transform of the end-effector. This desired transform of the end-effector is then passed to a lower-level inverse-kinematics controller, which is PD-based and handles computing the joint positions which are sent to the robot. We also predict a continuous desired gripper width $\in \mathbb{R}^1$, which is sent to a lower-level gripper controller. This interface of providing end-effector desired poses and gripper width setpoints matches the teleoperation modality by which we provide demonstrations. This action space can be written as the desired robot state, $\boldsymbol{x}_{\text{robot}}^{desired}$, where "robot" helps to signify that we only have direct control over the fully-actuated robot, and not the full state of the world that we would like our robot to interact with. Specifically, for our robot modeled at the level of end-effector setpoints, the full $\boldsymbol{x}_{\text{robot}}$ state consists of 7 degrees of freedom: the 6-DOF pose of the end-effector, and 1-DOF gripper.

$$\text{(observation, action pair)} \;\; (\boldsymbol{o}_t, \boldsymbol{a}_t) \tag{5.8}$$

$$\text{(action as desired robot state)} \;\; \boldsymbol{a}_t = \boldsymbol{x}_{\text{robot},t}^{desired} \tag{5.9}$$

Interestingly we have observed significant generalization performance increase, in some scenarios, for the robot policy to predict desired setpoints which are relative to its current position, rather than absolute. Specifically, for a given degree of freedom, we observe this performance increase by predicting the relative-desired position only when there is variation for that degree-of-freedom between different trajectories in the demonstration dataset. If that degree-of-freedom's trajectory is identical across all demonstrations, we have observed it is better to predict the absolute rather than relative position. When we use as our action space the desired robot state relative to current state we instead have

$$\text{(action as relative desired robot state)} \;\; \boldsymbol{a}_t = \boldsymbol{x}_{\text{robot},t}^{desired} - \boldsymbol{x}_{\text{robot},t} = \Delta \boldsymbol{x}_{\text{robot},t}^{desired} \tag{5.10}$$

An additional detail is that when using our LSTM sequence model, we follow the recommendation from [125] and train at rates lower than the full image rate (30 Hz), which may

help ease the ability of the sequence model to learn dependencies over time. We prefer 5 Hz, which still provides satisfyingly fast visual feedback for our tasks addressed. When we downsample the trajectory to a lower rate, since we are using position setpoints as the output of our policy, then we accordingly sample action pairs such that the desired action spreads the downsampled gap. For example when we slice a 30 Hz trajectory into six trajectories at 5 Hz, ($N_{slices} = 6$), then we use an "action bias" of $b = N_{slices} - 1 = 5$. If there is no downsampling, as when we train MLP models, then $N_{slices} = 1$ and $b = N_{slices} - 1 = 0$. In general we have

$$\text{(action as rate-adjusted relative desired robot state)} \;\; \boldsymbol{a}_t = \boldsymbol{x}^{desired}_{\text{robot},t+b} - \boldsymbol{x}_{\text{robot},t} = \Delta \boldsymbol{x}^{desired}_{\text{robot},t,b} \tag{5.11}$$

Finally, if we are predicting actions from our model at less-than-image-rate, then we interpolate position setpoints at 100 Hz between the previous and new desired setpoint. This aids in the robot moving smoothly.

5.4.3 Training for Feedback through Data Augmentation

We introduce a simple technique which we have found to be effective in at least partially addressing a primary issue in imitation learning: the issue of cascading errors [138]. While many other works have previously shown injecting noise into the dynamics either during imitation learning [139] or sim-to-real transfer [140] can aid the ability to deploy closed-loop policies that avoid cascading errors, we provide a simple method based only on data augmentation. This method does not address recovering from failures that cause discrete changes in the environment, but we have observed it does effectively enable the robot to stably execute a trajectory from only a single demonstration, whereas without noise augmentation fitting a policy to a single trajectory does not yield a stabilizing controller.

Since the output of our policy is desired position setpoints, and these position setpoints are handled by a lower-level feedback controller, then we can train our policy to perform feedback to the nominal training trajectories by augmenting the robot-state portion of the observations with a small amount of noise. If we consider only the robot-state portion

of the observation space, together with our desired robot-state, then we have, writing our relative-desired action with bias b,

$$\text{(robot-only portion of observation, relative-action pair)} \;\; (\boldsymbol{x}_{\text{robot},t},\; \boldsymbol{x}^{desired}_{\text{robot},t+b} - \boldsymbol{x}_{\text{robot},t}) \tag{5.12}$$

We can then augment the measured robot state with noise $\xi \sim \mathcal{N}(\mathbf{0}, \Sigma)$ with for example Σ as a diagonal covariance. Since our action space is relative to the measured robot state, then we must appropriately account for this noise when forming the relative action:

$$\begin{aligned} &\text{(robot-only portion of noisy-observation, relative-action pair)} \\ &\qquad \big(\boldsymbol{x}_{\text{robot},t} + \xi,\; \boldsymbol{x}^{desired}_{\text{robot},t+b} - (\boldsymbol{x}_{\text{robot},t} + \xi)\big) \end{aligned} \tag{5.13}$$

A remaining question, of course, is what scale of noise is appropriate. In practice we find that, given the scale of our robot its typical speeds, without any downsample slicing then $\sigma = 1$ mm is an appropriate noise scale for the translational components. When we use an action bias $b > 0$, then we scale the noise proportional to the action bias, i.e. for $b = 5$ we use σ of 5 mm. For the rotational components we use $\sigma = 1$ degree when $b = 5$, and for the gripper we use $\sigma = 0.01$. The gripper's domain ranges from 0.0 to 0.1.

$SE(3)$ Formulation of Training for Feedback through Noisy Data Augmentation

A technical detail is that our robot's end-effector pose ${}^{W}\boldsymbol{T}_t$, here noted at time t in world frame W, actually lives in the Special Euclidean group $SE(3)$ for rigid-body transformations in three dimensions. Accordingly we formulate our noise augmentation and relative-desired action space in terms of $SE(3)$. In this section we follow the transform notation of ${}^{to}\boldsymbol{T}_{from}$, so that we can write chained transforms together as ${}^{c}\boldsymbol{T}_a = {}^{c}\boldsymbol{T}_b \, {}^{b}\boldsymbol{T}_a$, for some frames a, b, and c.

Consider again the robot-only subset of the observation-action pair, but this time we will consider both the observation of and the desired action for the end-effector pose $\in SE(3)$,

first with the action in world frame:

$$(\text{end-effector only portion of observation, action pair})\ ({}^{W}\boldsymbol{T}_t,\ {}^{W}\boldsymbol{T}_{t+b}^{desired}) \tag{5.14}$$

to make the action a relative transform, concatenating transforms we have ${}^{W}\boldsymbol{T}_{t+b}^{desired} = {}^{W}\boldsymbol{T}_t\ {}^{t}\boldsymbol{T}_{t+b}^{desired}$. Accordingly our relative-frame action space, with no noise augmentation, is

$$(\text{end-effector only portion of observation, relative-action pair})\ ({}^{W}\boldsymbol{T}_t,\ {}^{t}\boldsymbol{T}_{t+b}^{desired}) \tag{5.15}$$

To now add noise augmentation, we sample a small noisy transform ${}^{t}\boldsymbol{T}_{\tilde{t}}$, which transforms from a noisy end-effector frame to the true end-effector frame at time t. The observed noisy end-effector transform as observed in world frame becomes ${}^{W}\boldsymbol{T}_t\ {}^{t}\boldsymbol{T}_{\tilde{t}}$. For the relative desired transform, we now must have this be relative to the noisy transform, since that is where the robot is told to believe where it is. Concatenated into world frame this action would be ${}^{W}\boldsymbol{T}_t\ {}^{t}\boldsymbol{T}_{\tilde{t}}\ {}^{\tilde{t}}\boldsymbol{T}_{t+b}^{desired}$, but we ask the policy to only predict the relative component:

$$(\text{end-effector only portion of noisy-observation, relative-action pair})\ ({}^{W}\boldsymbol{T}_t\ {}^{t}\boldsymbol{T}_{\tilde{t}},\ {}^{\tilde{t}}\boldsymbol{T}_{t+b}^{desired}) \tag{5.16}$$

This completes the $SE(3)$ noise-augmented and relative-action-space formulation.

5.4.4 Dynamic-Scene, Multi-View Time-Sychronized Correspondence Training

Unlike in our previous work (Chapter 3) in which we trained correspondence models only for static environments, we now would like to train correspondence models of dynamic environments. To address this challenge, we employ a multi-view camera setup. Although the environment is dynamic and we do not perform non-rigid dynamic reconstruction [69] as in [76], we can instead correspond pixels between two camera views if they are approximately synchronized in time. For this we require the cameras to be calibrated both for their intrinsics and extrinsics into a common frame. Like the static-scene case, finding pixel correspondences between images can remain a simple geometric operation involving only

depth images, camera poses, and camera intrinsics.

Compared to our previous, static-scene moving-camera training procedure, the dynamic-scene scenario suffers significantly in (a) not being able to benefit from a wide variety of views from which to fuse highly accurate accurate 3D reconstructions, and (b) not being able to benefit from a wide variety of rotations, viewing angles, and scales in order to provide correspondences between a diversity of views. To help alleviate (a) the reduced correspondence precision due to the lack of a highly accurate 3D reconstructions, we add photometric-error-based rejection of correspondences. Specifically, if the imperfect geometric correspondence provided by noisy depth images predicts two pixels a and b with a photometric error $||[R_a, G_a, B_a]^T - [R_b, G_b, B_b]^T||_2$ to be above a threshold, then we reject this correspondence. To help alleviate (b), we add significant affine transformation augmentation (which includes rotation, scale, and shear augmentations), and for real-data training (but is unnecessary for simulation data) we also augment the hue and brightness of images. Due to the aggressive data augmentation, rather than only sampling correspondences between different views and then augmenting these images, we also with 50% probability just choose one image, pair it with itself, and let the augmentation provide the only correspondence challenge for the model to learn. While these photometric-error-based-rejection (i.e., [141]) and data augmentation methods are standard computer vision techniques, they are critical new components to our correspondence learning practice.

5.4.5 Policy Models

We use two standard classes of policy models, Multi-Layer Perceptrons (MLP) and Long Short-Term Memory (LSTM) recurrent networks, which are familiar model classes to many different types of machine learning problems and in particular have been demonstrated to be viable for real-world visuomotor control [39, 78, 125]. In our evaluations the MLPs are only provided current observations, $\pi(\boldsymbol{o}_t)$, whereas through recurrence the LSTMs use the full observation sequence. For details on the model architectures and training, see the Appendix.

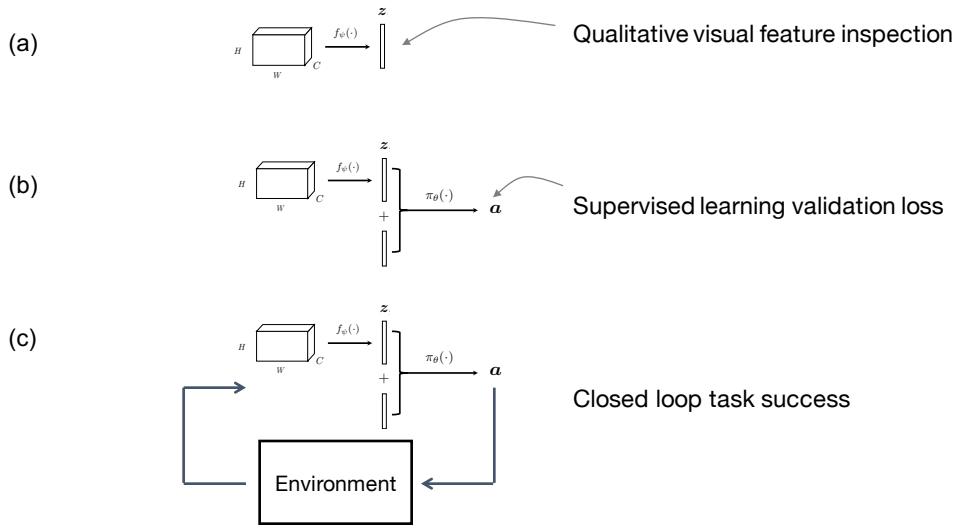


Figure 5-2: Depictions of used methods for evaluating different methods of training the visual portion of visuomotor policies: (a) qualitative visual feature inspection, (b) supervised learning validation loss for behavior-cloned policies, and (c) closed loop task success evaluations.

5.5 Experimental

5.5.1 Evaluating Visual Representations for Visuomotor Policy Learning

We offer three types of analysis of the different visual representations. Since the measure of a representation's usefulness should be based on what it is actually used for, we argue that the last of these methods is ultimately the most important. The others additionally complement the analysis.

1. *Qualitative Visual Feature Inspection.* We can qualitatively inspect the features produced by different visual models to attempt to gain insight. Both the auto-encoder method and end-to-end method we analyze utilize a spatial expectation operation in their model architecture – the two methods actually share an identical model architecture; they are just trained differently. Since the prior works have interpreted the features at the spatial expectation layer, visualized on top of the image, as identifying various features in the robot's environment [39, 119], we can perform this analysis and compare with our own correspondence method as well. Note that not all intermediate representations are meant to be interpretable as features in image-coordinate

space, but this is a design choice for these architectures.

2. *Supervised Learning Validation Error.* In our experimental analysis we learn our policies through supervised learning (imitation learning), and at least in simulation we can create the ingredients of a proper supervised learning analysis. By having the expert use a ground-truth policy for which there is a correct action everywhere in the state space, we can withhold a *validation set* of demonstrations by the expert and use the standard metric of supervised learning: generalization error on the withheld validation set. Note that the expert demonstrator policy may be simply designed using ground-truth state information in simulation, although the learned policy may only be provided observations rather than state in order to learn the policy. There are a couple practical benefits of using a supervised learning validation error metric: (i) it can be computed very quickly, i.e. less than a few seconds to test our models on the entire test set, which is much cheaper computationally than deploying trained models in a closed-loop simulation environment including visual simulation. Additionally, (ii) it becomes even easier for the research community to reproduce our quantitative results – the metric is just computed on static data and does not even require a simulation environment. It is surprising that to our knowledge we do not know of other works at least in the robotic manipulation imitation learning literature that have used supervised learning validation error to evaluate models. We do observe, however, that when comparing trained models, the relative ordering of their performance on supervised learning validation error does not always match their relative ordering of their closed-loop performance. At least one reason for this is the *non-uniqueness* of policies for accomplishing any given task. Although our ultimate goal is closed-loop performance, it is interesting to keep in mind that, when performing behavior cloning, the supervised learning validation error does actually more directly measure the model's success at *doing what it was asked to do* – which is to fit the policy.

3. *Closed-Loop Performance on Task Success.* Since our ultimate goal is to deploy closed-loop policies, we also compute metrics of task success based on the final state of the robot over some finite horizon. This is effectively a reward function, but since

none of our evaluations use reinforcement learning on this signal, we don't call it a reward for clarity. For a given task, the task success metric γ effectively uses the true dynamics $f(\cdot)$, the true state $\boldsymbol{x}$, and the true observation function $g(\cdot)$ (all unknown to the robot) to compute

$$\gamma = \gamma(\boldsymbol{x}_f) \tag{5.17}$$

$$\forall t, \;\; \dot{\boldsymbol{x}}(t) = f\big(\boldsymbol{x}(t), \pi(g(\boldsymbol{x}))\big) \tag{5.18}$$

where the task success metric $\gamma(\cdot)$ is designed for the specific task, and $\boldsymbol{x}_f$ is the final state. We allow a finite amount of time for the robot to attempt the task, or terminate early if the robot reaches the maximum γ.

5.5.2 Simulation Experimentation

We use simulated imitation learning tasks (Fig. 5-3) to compare the generalization performance of behavior-cloned policies where the only difference is how the "visual representation" $\boldsymbol{z}$ is acquired. The first two tasks involve reaching to an object whose configuration varies between trials either in translation only, or rotation as well. An additional two tasks are pushing tasks, which require feedback due to simulated external disturbances. Expert demonstrations use simple hand-designed policies using ground truth object state information.

The compared methods are:

1. *Ground truth 3D points (GT-3D):* $\boldsymbol{z}$ is ground truth world-frame 3D locations of points on the object.

2. *Ground truth 2D image coordinates (GT-2D):* $\boldsymbol{z}$ is similar to the previous baseline, but the points are projected into the camera using the ground truth camera parameters.

3. *Autoencoder (AE):* $\boldsymbol{z}$ is the encoding of a pre-trained autoencoder, similar to the visual training in [119, 123].

4. *End-to-End (E2E):* $\boldsymbol{z}$ is the intermediate representation from end-to-end training.

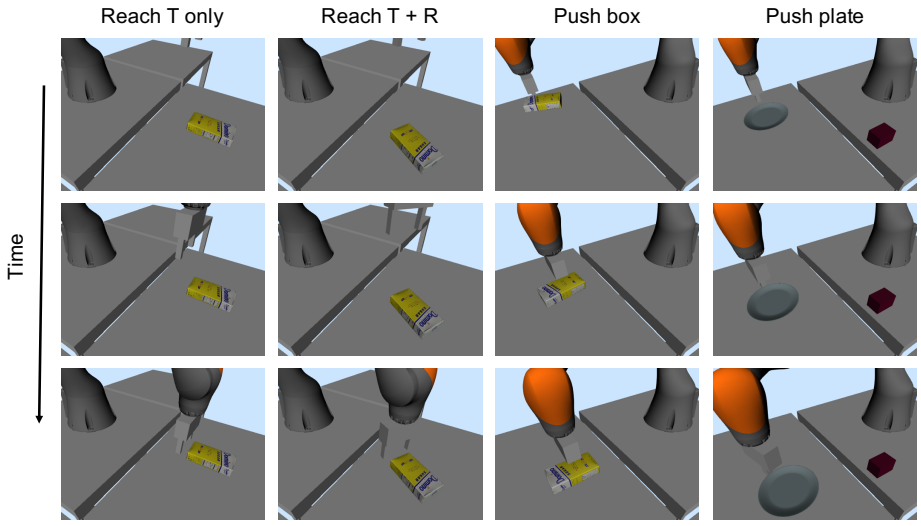


Figure 5-3: RGB images used for visuomotor control in each of the simulation tasks. T=translation, R=rotation, see Appendix for task descriptions.

This closely resembles the visual training and models in [39, 78], but we do not also add pose-based losses, in order to investigate end-to-end learning without these auxiliary losses.

5. *Ours, Dense descriptors (DD):* $\boldsymbol{z}$ is the expected image-space locations (DD-2D) or 3D-space locations (DD-3D) of the descriptor set $\{\boldsymbol{d}\}_i$, where the visual model was trained on dense correspondence.

Note that the two vision-based baselines AE and E2E share an identical model architecture for producing $\boldsymbol{z}$, and differ only in the method used to train the parameters. The model is close to [39, 78, 119] with the key architectural traits of having a few convolutional layers followed by a channel-wise spatial expectation operation, which has been widely used [24, 122, 123, 135, 142–144]. Most methods we compare (AE, E2E, DD-2D) use only one RGB camera stream as input to learned policies; DD-3D additionally uses the depth image. DD methods use descriptor set optimization (Sec. 5.3.2) and use both views for the correspondence training, before policy training. See the Appendix for additional model and task details.

Figure 5-4: Example demonstration, in this case for the shoe flip task, with the robot teleoperated by the demonstrator.

5.5.3 Real-Hardware Experimentation

Our hardware closely resembles our simulation experimentation. We used a Kuka IIWA LBR robot with a Schunk WSG 50 parallel jaw gripper. RGBD sensing was provided by RealSense D415 cameras rigidly mounted offboard the robot and calibrated to the robot's coordinate frame. Note that for effective correspondence learning between views, it is ideal to have views with *some* overlap such that correspondences exist, but still maintain different-enough views. Our hardware setup included three available cameras, although in practice we would choose two of these cameras to log demonstration data for any given task. Our correspondence training leverages multiple views, but note that, for all trained and deployed policies, we only used a single monocular RGB camera (using only the color stream off one of the RealSenses).

We test our method on five different challenging tasks in hardware, each offering a different challenge for the robot. Each of the tasks are human-demonstrated via teleoperation. These tasks are described in the Appendix.

5.5.4 Imitation Learning Demonstration Modality

While many different works in Learning from Demonstration have employed a wide variety of demonstration modalities, we opt for the simple method of teleoperating the robot using a mouse and keyboard, with all of the robot's degrees of freedom mapped to different keys and mousing directions. While this does not allow for as dynamic and fluid demonstrations as with, for example, hand-held 6-DOF tracked controllers as are used for virtual reality

devices [78], this method was sufficient and involves little engineering overhead. For the operator to observe the robot and its environment during demonstrations, we simply have the operator operate "visually", i.e. using their own eyes rather than visualization of the robot's sensors. Note that this introduces a difference in the observation space for the demonstrator and the learner, namely that the robot learner must use its cameras from their viewpoint(s), whereas the demonstrator uses their human eyes from their viewpoint. Note that, as is shared with all remote teleoperation methods, our method of demonstration does not introduce obstructions to the robot's observation space, as kinesthetic demonstrations do, and does not present an embodiment correspondence problem as is present in third-person imitation, for example in [77].

5.6 Results

Our experimentation sought to answer these primary questions: (1) Is it possible to use self-supervised descriptors as successful input to learned visuomotor policies? (2) How does visual correspondence learning compare to the benchmarked methods in terms of enabling effective policy learning, as measured by generalization performance and sample complexity? We also evaluate (3) the effect of noise augmentation, and (4) whether our dynamic-scene visual training technique is capable of effective correspondence learning. Simulation results are detailed in Sec. 5.6.1, hardware results are in Sec. 5.6.2.

5.6.1 Simulation Results

Table 5.1 contains the results of the simulation experiments. Interestingly we find that our method's visual representation is capable of enabling policy learning that is remarkably close in performance to what can be achieved if the policy has access to ground truth world state information. In contrast the performances of the end-to-end (E2E) and autoencoder (AE) methods vary much more across the different tasks. Since our method benefits from object mask information during visual training, we also experimented with letting the autoencoder use this information by applying the reconstruction loss on only the masked image. Additionally we tried training the autoencoder end-to-end during behavior cloning.

Method / Task	Reach T only	Reach T + R	Push box	Push plate
Ground truth 3D points	100.0	100.0	100.0	90.5
Ground truth 2D image coord.	94.1	95.6	100.0	92.0
RGB policy input				
Autoencoder, frozen	8.1	61.1	31.0	53.0
Autoencoder w/ mask, frozen	16.3	10.0	73.0	67.0
Autoencoder, then End-to-End	40.7	38.9	–	16.0
End-to-End	43.0	32.2	**100.0**	5.5
End-to-End (34-layer ResNet)	–	3.3	–	–
DD 2D image coord. (ours)	**94.1**	**97.8**	**100.0**	**87.0**
RGBD policy input				
DD 3D coord. (ours)	**100.0**	**100.0**	–	**98.0**

Table 5.1: Summary of simulation results (success rate, as %). DD = Dense Descriptor. See Appendix for task success criteria and additional details.

Both of these yield mixed results, depending on the task.

Since the vision network in our method is a 34-layer ResNet, we wanted to see if the end-to-end method would benefit from using the same, deeper vision backbone. The deeper network did not improve closed-loop performance (Table 5.1) although it did reduce behavior-cloning validation error. This suggests the advantage of our method comes from the correspondence training rather than the model capacity.

The binary success metrics of Table 5.1, however, do not fully convey the methods' performances. We also experiment with varying the number of demonstrations, and characterize the performance distributions. By plotting the performance for the *"Reach, T + R"* task over a projection of the sampled object configurations (Figure 5-5), we learn that the few failures of our method occur when the box position lies outside the convex hull of the training data. Interestingly the GT-2D baseline also struggles with similar failure modes, while the GT-3D method succeeds in more cases outside the convex hull. This suggests that policies that consume 3D information are better able to extrapolate outside the training distribution; our DD-3D method also provides better generalization than DD-2D. The baseline vision-based methods do not generalize as well; for example, the E2E performance distribution is shown in Figure 5-5. On this task we find that with just 30 demonstrations our method outperforms both AE and E2E with 200 demonstrations.

The pushing tasks are of particular interest since they demand closed-loop visual feedback. Disturbances are applied to the object both while collecting demonstrations and

Noise / Method	*30 demonstrations* GT-2D	DD-2D	*200 demonstrations* GT-2D	DD-2D
No noise	5.6	1.1	5.6	3.4
With noise	**73.3**	**73.3**	**95.6**	**97.8**

Table 5.2: Comparison of using our feedback-training noise augmentation technique or not (success rate, as %) on the *"Reach, T + R"* task. *No noise* uses $\sigma_i = 0.0$, whereas *With noise* uses $\sigma_{\text{translation}} = 1\text{mm}$, $\sigma_{\text{rotation}} = 1$ degree. See Section 5.5.2 for descriptions of GT-2D (ground truth) and DD-2D (our method).

deploying the learned policies. Since the *"Push box"* task used a dynamic state feedback controller to provide demonstrations, we find that we need the sequence model (LSTM) for the policy network to achieve the task, even when the policy has access to ground truth object state. On the other hand, the *"Push plate"* task employed a static feedback controller to provide demonstrations, and so MLP models that consume only the current observation, $\pi_{\theta_p}(\boldsymbol{o}_t)$, are sufficient.

Interestingly a variety of methods performed well on the *"Push box"* task while large differences were evident in the *"Push plate"* task. We speculate that this is because higher precision is required to accurately push the plate as compared to the box. Since the rectangular robot finger experiences a patch contact with the box, while only a point contact with the plate, there is more open loop stability in pushing the box. On the harder *"Push plate"* task we found that our DD-2D method performed almost as well as the GT-2D baseline and significantly outperformed both the AE and E2E approaches, and that DD-3D improved performance even further.

Additionally we find (Table 5.2) our noise augmentation technique (Sec. 5.4.3) has a marked effect on task success for behavior-cloned policies. This applies to ground truth methods and our method, with as few as 30 demonstrations or as many as 200.

More simulation results, including qualitative feature inspection and quantitative comparisons, are provided in the Appendix.

5.6.2 Hardware Results

We validate both our visual learning method and its use in imitation learning in the real world. As in simulation, we *only use demonstration data* for both visual training and policy learning; no additional data collection is needed. While the simulation results provide a

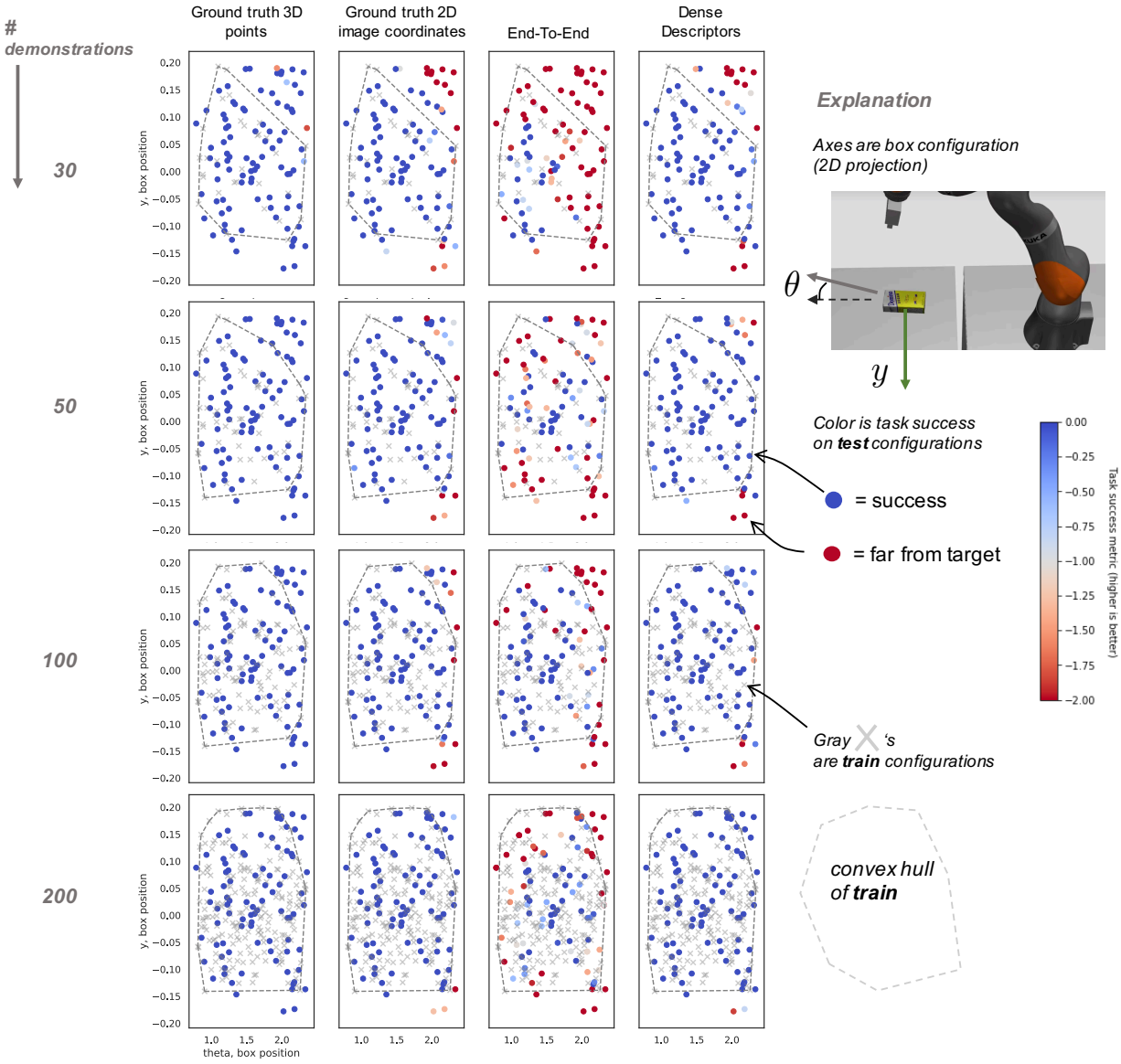


Figure 5-5: Task success distribution plotted over the 2D projection of the varied box configurations for the *"Reach, T + R"* task. The color of each point represents the result of deploying the learned policy with the object at that θ, y. Specifically the color encodes the distance to target threshold: $\min(0, -(\Delta\text{translation} + \Delta\text{rotation}) + \epsilon)$, where ϵ is the success threshold. The x coordinate is not shown in order to plot in 2D. Dark blue corresponds to perfect performance on the task with the object in that configuration, red is poor performance. Note that the color scale cuts off at -2 in order to highlight differences in the range [-2,0]. Each gray "x" in each subplot represents the configurations of the box in the training set, for either (from top to bottom) 30, 50, 100, or 200 demonstrations. The dashed gray line shows the convex hull of the respective training sets.

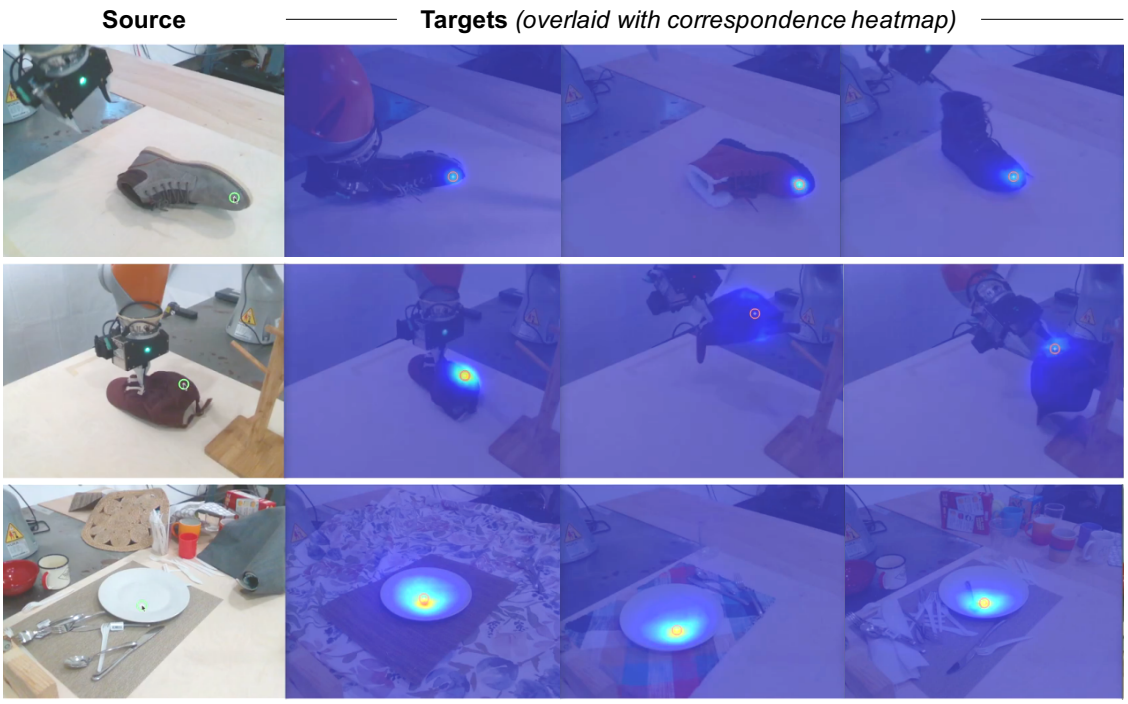


Figure 5-6: Learned correspondences from demonstration data, depicted as correspondence heatmaps between a source pixel (left, with the green reticle) and target scenes (right, with red reticle as best predicted correspondence).

controlled environment for comparisons, there are a number of additional challenges in our real world experiments: (i) visual complexity (textures, lighting, backgrounds, clutter), (ii) use of human demonstrations rather than expert simulation controllers, (iii) real physical contact, and (iv) imperfect correspondence learning due to noisy depth sensors and calibration. Our hardware experiments test all of these aspects. All real hardware experiments use LSTM policy networks, since we suspect our human demonstrators use dynamic internal state.

Learned Correspondences from Dynamic Scenes

Fig. 5-6 displays visualizations of learned correspondences from demonstration data. The results show that the learned visual models, despite imperfect depth sensor noise, calibration, and only time-synchronized image pairs, are capable of identifying correspondences across a class of objects, for an object in different deformable configurations, and for objects in a diversity of backgrounds. Figure 5-7 displays class-general correspondences for a particularly challenging instance with large shape variation.

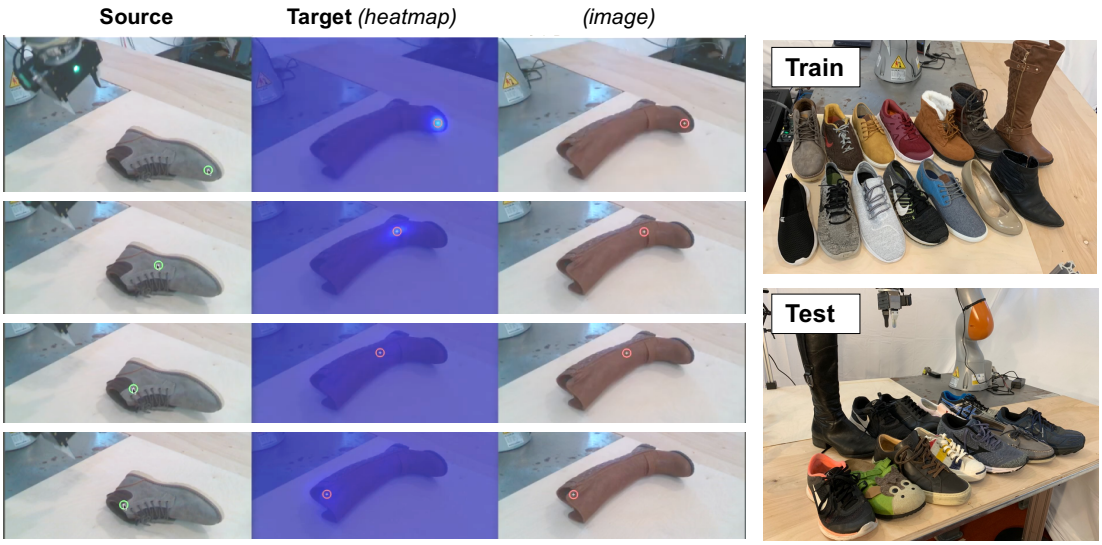


Figure 5-7: Learned correspondences (left) between a standard-sized shoe and an extra-tall boot. A small amount of movement near the top of the ankle on the shoe (far left) corresponds to a "stretched-out" movement on the boot (right). Images cropped for visualization. Also shown are shoe train and test instances (far right).

Real-World Visuomotor Policies

Figure 5-8 displays examples of autonomous hardware results, and Table 5.3 provides a quantitative overview. To highlight a few results, several of the tasks achieve over 95% reliability, including the *"Push sugar box"* task with and without disturbances, and the *"Flip shoe, single instance"* and *"Push-then-grab plate"* tasks without disturbances. Each of the different tasks present significant challenges, best appreciated in our video. Several of the tasks include non-prehensile manipulation, including pushing the box and plate, and flipping the shoes. In the *"Pick-then-hang hat on rack"* task, the robot autonomously reacts to the deformable configuration of the hat after disturbances. The *"Push-then-grab plate"* task as well is highly challenging given the visual clutter, the symmetry and lack of visual texture for the object, and requires using "extrinsic dexterity" [145] via the wood block to enable sliding the gripper into position to grasp the plate.

5.7 Conclusion

Our experiments have shown self-supervised correspondence training to enable efficient policy learning in the real world, and our simulated imitation learning comparisons em-

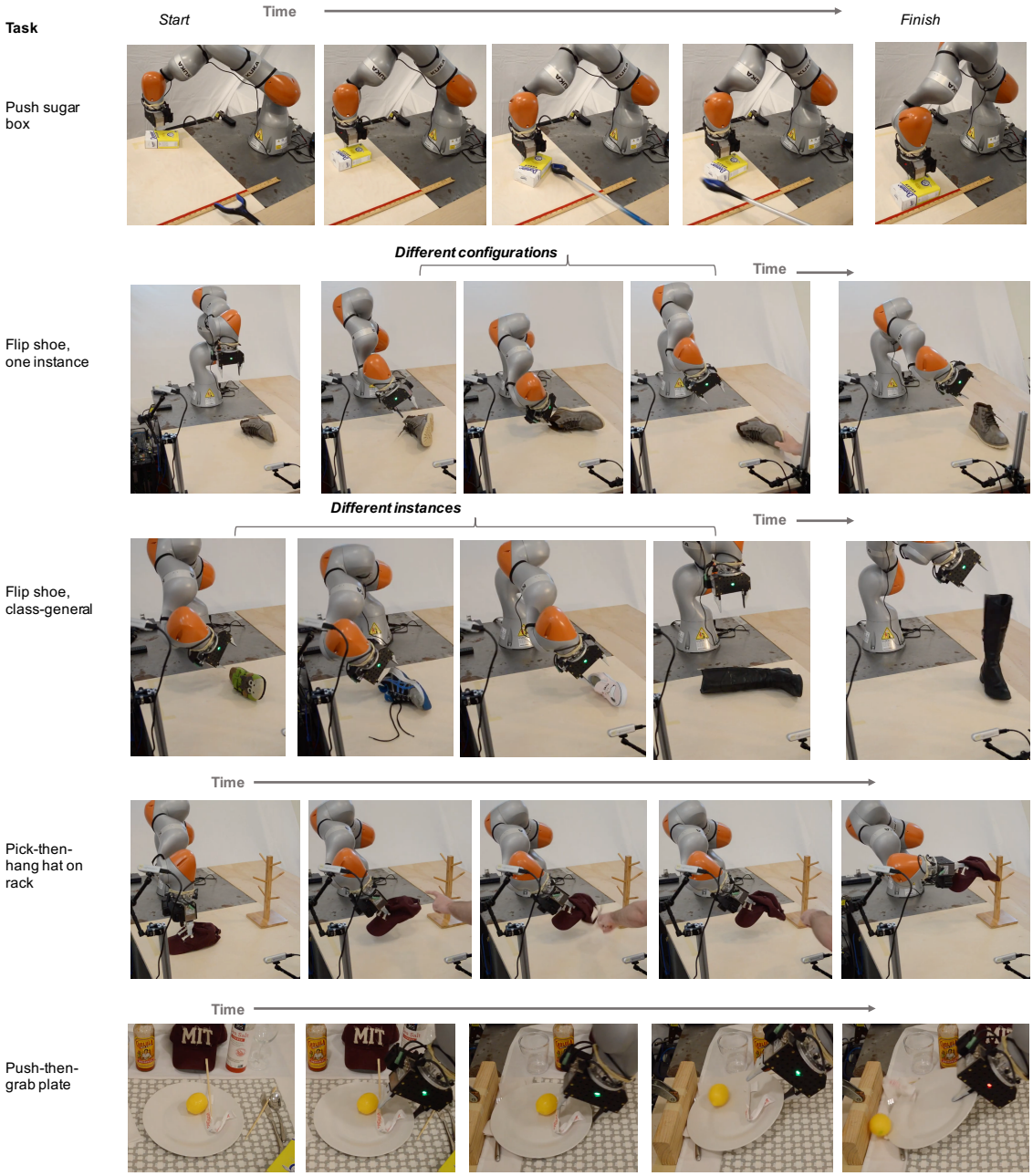


Figure 5-8: Depictions of the five demonstrated hardware tasks.

Task	Success criterion	Trained with manual disturbances	*Without disturbances* # attempts	# success	%	*With disturbances* # attempts	# success	%	*Demonstration data* # total	time (min.)
Push sugar box	box is < 3 cm from finish line	yes	6	6	100.0	70	68	97.1	51	13.9
Flip shoe, single instance	shoe is upright	no	43	42	97.7	40	35	87.5	50	6.5
Flip shoe, class-general										
previously seen shoes (14)	shoe is upright	no	43	38	88.4	–	–	–	146	17.5
novel shoes (12)	shoe is upright	no	22	17	77.3	–	–	–	146	17.5
Pick-then-hang hat on rack	hat is on the rack	yes	50	42	84.0	41	28	68.3	52	11.5
Push-then-grab plate	plate is off the table	yes	22	21	95.5	27	22	81.5	94	27.4
Total			186			178				

Table 5.3: Summary of task attempts and success rates for hardware validation experiments. Autonomous re-tries are counted as successes.

pirically suggest that our method outperforms two vision-based baselines in terms of generalization and sample complexity. While different hyperparameters, model architectures, and other changes to the baselines may increase their performance, our method is already near the upper bound of what can be expected in the used experimental setting: it achieves results comparable to baselines using ground truth information. One reason our approach may outperform the vision-based baselines is that it additionally uses a fundamentally different source of supervision, provided by visual correspondence training. Since our approach is self-supervised, it does not entail additional human supervision.

Dense descriptor learning has shown to be an exciting route for improving visuomotor policy learning. While this has enabled the variety of tasks shown, there are many that are out of scope. One current limitation is that our visual representation does not explicitly address simultaneously viewing multiple object instances of the same class. Future work could, similar to the visual pipeline in [82], combine both instance-level segmentation with intra-instance visual representations. Additionally, returning to the cooking eggs example in the Introduction, it is interesting to consider using spatial correspondence as part, but not the entirety, of the visual representation of the world.

Chapter Acknowledgement

This chapter was joint work with Lucas Manuelli, and Russ Tedrake. This work was supported by National Science Foundation Award No. IIS-1427050, Lockheed Martin Corporation Award No. RPP2016-002, and an Amazon Research Award grant. The views expressed are not endorsed by our funding sponsors.

Chapter 6

Conclusion

In this thesis, we have investigated a variety of directions related to self-supervised robot learning and manipulation. In this conclusion we will synthesize what we have learned from each of these different directions, compare them, and propose future directions.

With LabelFusion, we demonstrated a way of leveraging multi-view geometry and an intuitive human interface to act as a large multiplying factor on the value of information provided by humans in training artificial perception. In hindsight, and in comparison with other more recent work, it has become even more clear how to evaluate its unique benefits and limitations. The primary benefits are in the particular value of the ground-truth data it is able to provide, and the mix of both accuracy, flexibility, ease-of-use, and scalability that is provided. The value of the ground truth data provided by LabelFusion is important to appreciate, and it is a type of value that is not present in the same way in, for example, the Dense Object Nets approach. In particular, when the perception system seeks to estimate a pose and pixelwise instance labels, then we believe there exists ground truth data which is *uniquely correct* – there is one right answer for each of these representations of the data. Additionally, these outputs of the automated system are also the "final answer" that perception systems can either be tested against, trained on, or both. This is in contrast with, for example, Dense Object Nets, in which although the self-supervised underlying pixelwise correspondences are uniquely correct, the final representation that is acquired and used (the dense descriptor representation) has no unique answer. While determining the 6DOF pose of an object is inherently an arbitrary process (what should be the correct origin frame of

my shoe?), deciding an arbitrary one via a single canonical 3D mesh model does make the problem well-defined such that there is a single correct answer. There is some practical additional benefit in that LabelFusion produces standard, widely-used representations (6DOF pose and pixelwise labels), which enables immediate widespread usability by the research community. An additional benefit to appreciate for LabelFusion is that the quality of the data produced can be highly accurate. The generated ground truth data will be accurate within the margin of error of what is acceptable to the human labeler and the error of the dense 3D reconstruction.

While LabelFusion is not fully self-supervised, there are some types of information that are perhaps best for a human to efficiently provide, rather than the robot to fully self-supervise on its own – for example, the semantic class of an object. The goal, however, should be for a robot to take small pieces of efficiently provided information ("this is a shoe") and be able to learn as much as possible about its world from involving that piece of information in the rest of its autonomously-acquired world model. In this case, the dense 3D reconstruction of the world, and each of the camera poses, are autonomously acquired via dense visual SLAM, and this enables taking the human-provided information ("this is an initial guess pose of the object in world frame"), refining it with local optimization, and associating the result with each individual image. As was included in analysis in our paper's supplementary information, we showed that using LabelFusion provided an approximate increase in efficiency of about four orders of magnitude (10,000x) compared to traditional single-image polygon-based labeling. This impressive number, however, must be partially tempered since we also showed in analysis that using thousands of nearby views of a single static scene has diminishing returns – in particular, with data from a slow-moving camera (average 0.03 m/s) within the workspace of a single 7-DOF arm with 1.8 m^3 working envelope volume (Kuka IIWA LBR) we observe diminishing returns in trained segmentation network performance using more than a few hundred views of a scene, whereas for data from a faster human-carried camera (typically 0.05-0.17 m/s) we do observe incremental value in using 1,000 rather than 100 views of a scene. Even after adjusting for subsampling the generated data by one or two orders of magnitude, LabelFusion is still 2-3 orders of magnitude (100x-1,000x) faster than a traditional single-image labeling technique.

The primary limitations of LabelFusion, on the other hand, are simply stated: (1) the method only works for rigid objects with no easy way to extend to deformable objects, (2) acquiring full, watertight 3D object meshes can be time-consuming, and (3) it is not fully self-supervised, requiring human initialization of 3D object poses.

With Dense Object Nets, however, the advantages of moving away from rigid 6DOF poses became readily apparent. By opting for dense descriptor representations of objects rather than 6DOF object poses, and an associated self-supervised learning scheme, we directly address the primary limitations of LabelFusion: we can (i) handle both deformable and rigid objects, (ii) not require complete watertight 3D object meshes, and (iii) require no human in the annotation process for instead a fully self-supervised pipeline. In hindsight after presenting this work to the research community, I believe there are a few elements that resonate well with current researchers in robotics. The primary conceptual contribution is easily understood and widely applicable – we can use dense visual object descriptors as a way to think about manipulation problems. It is a rich intermediate representation and one that may be useful for many types of tasks.

It also has resonated well that our learning process is "sample efficient", in particular in terms of the amount of time that is needed to interact with an object (only about 5 minutes) and then train (only about 15 minutes on a single consumer-grade GPU). Sample efficiency is a key concern of many current learning-based approaches particularly in robotics, both in supervised learning where thousands of human-labeled samples are often required, and in model-free reinforcement learning where the need is for tens of thousands of episodes with reward signals, which may be hard to attain for a real robot and there are challenges in transferring from simulation. A beautiful aspect of training with dense descriptors is that, in addition to them being self-supervised with no human involvement, there is also an enormous amount of individual pieces of information when the labeling happens *densely*. To contrast, labeling a semantic class of one image patch provides one piece of information.

While it is easy to acquire a partial 3D mesh of an object in one configuration, one must be careful in combining multiple scans (for example, to cover the bottom of the object), although other work has shown how to automate this [146].

The original idea for the paper was actually to let global 6DOF registration algorithms solve the initialization process entirely, removing the need for humans to initialize poses. Although we found current global registration methods to not be robust enough in practice, potentially in future they will be, and potentially due to training on data generated from pipelines like LabelFusion.

On the other hand with dense correpondence training, even just from a single scene, if we take N views, there are $\mathcal{O}(N^2)$ pairs of images for which each step of training we take one pair and provide *millions* of match/not-a-match labels. In total, for images that are $W \times H$ pixels, there are $\frac{(WH)(WH-1)(N)(N-1)}{4}$ non-match pixel pairs, which for standard parameters from our work is 3.8×10^{15} constraints for the visual system, from just a single scene and with no human involvement needed. It is a very rich world to live in for visual robots when labeling is both dense and self-supervised. Other work such as self-supervising video prediction of robots interacting with objects [18, 19] also has these two properties of being (i) self-supervised and (ii) *dense* and therefore an enormous amount of individual labels.

The amount of self-supervision is genuinely satisfying, and there is scalability in the promise of "put any object in front of the robot, and it can autonomously learn this representation". It's insightful to contrast the satisfaction of human-independent, fully-autonomous learning with our LabelFusion work: even though we were able to efficiently generate over 1 million labeled object instances with just a handful of hours of labeling data, it was still an excruciatingly boring handful of hours of labeling. The two of us authors that did the labeling unanimously felt that we never wanted to have human labeling be a part of a system again. In contrast, with Dense Object Nets the more scalable, no-human-involvement makes it seem more of a viable component of the long-term "right answer" for robot manipulation. It is both a representation that is autonomously acquirable, but also the representation is sufficiently flexible to handle deformable objects, and also tends to find consistency across entire classes of objects, which is encouraging in terms of its scalability to a wide set of manipulation problems.

The dense descriptor representation of objects for manipulation, however, has the contrast with 6DOF poses and pixelwise class labels (as from LabelFusion) in that there is no unique answer for the representation. Like with anything trained in a metric-learning style, the absolute value of any feature does not matter, instead it is only the pairwise relationships between features that matter. This has the disadvantage that, to the extent systems are built on top of using the dense descriptor intermediate representation, they are dependent on a

$W = 640$, $H = 480$, $N = 400$

feature space that has arbitrary absolute answers. Despite there being no unique answer for the representation, however, it can easily be evaluated for its correctness. It is simple to withhold autonomously-acquired training data as test data and use various quantitative metrics to measure the performance of the dense correspodences between two different views of the same scene. As we also do in the paper, it is also easy to provide a small amount of human labeling (only for evaluation, not for training) on the type of cross-scene and cross-instance correspondences for which we have no way of autonomously acquiring ground-truth labels.

Another consideration that is a limitation of our self-supervised learning method is that there is no guarantee that consistency between different configurations of the object and across the class of objects will arise. The best answer we still have at this point is that although we cannot guarantee consistency will arise, it is easy to check whether or not it happens. There is a difference though between how these two types of consistency (configuration-consistency and class-consistency) may be addressed. For dense correspondence across configurations, this type of training data is acquirable via any method that does non-rigid dynamic reconstruction, as was done in prior work [16], but has challenges in deploying for an autonomous system. For dense correspondence across the semantic class, however, it does not seem possible to guarantee that class consistency will happen in any self-supervised fashion. The counter example would be with certain types of adversarial objects: a shoe could be designed that has an optical illusion such that it looks upside-down when it is right-side up. Another direction would be to involve additional information that could help with class-wide consistency – we could incorporate other sources of information such as observing objects in use by humans: "I know this is the toe of the shoe because the human just put its foot in it this way". This is left, however, to other researchers to pursue.

One question that I often get is whether it makes sense to have descriptor-learning be associated with classes of objects, or whether we should just instead learn “descriptors-for-everything”. Is there any value in training descriptors one class at a time? Although further experimentation would be needed, I have hypotheses of why class information may help: by providing the additional information that objects are the same member of a class,

this may encourage generalization that is more meaningful. Alternatively, if just training descriptors simultaneously on everything in the world, they may end up generalizing just across "yellow rectangular things", rather than generalizing differently across books that happen to be yellow, and cereal boxes that happen to be yellow. It is useful to keep in mind that in the visual correspondence training methods presented in this thesis, the learners do not have access to the broad context and world knowledge that we as humans do. Instead, the learner is just looking at a bunch of pixels and trying to minimize a loss function that has only to do with correspondence. Future work may investigate learning very large-scale descriptor training across many classes of objects, and test whether there are benefits of including class-identify information and how descriptors fare on entire classes that have not been seen before. (For example, a model that has trained descriptors on cats, dogs, foxes, wolves, etc., but hasn't seen a hyena before.)

There is also the ability, although it does not have to be associated with class-generality, of changing the *contract* of what descriptors are chosen to signify. Typically in computer vision, whether stated or not, we assume that a pixel's descriptor should signify something about the first intersection of that pixel's camera ray with the solid geometry of the world – the first thing that ray hits. This contract for a descriptor inherently will not address occlusions. If an object is partially occluded by another object, then the occluding object should overtake that descriptor's value. By instead choosing descriptors to correspond to something about the class, the contract may be changed – the descriptor can correspond to the first ray intersection with an object of the class, rather than just any object. This naturally allows inference through partial occlusions of objects in the class. There may be other benefits to this line of thinking as well.

Finally, with our imitation learning work, we demonstrated that there is indeed a significant benefit of using correspondence self-supervision in order to learn visuomotor policies. This has been an exciting discovery, and to be honest, I have been fully surprised with how well the robot learns to perform the manipulation tasks with such a limited amount of data. I believe there is certainly an exciting future ahead for self-supervision and policy learning.

In terms of limitations with the presented visuomotor policy learning approach and specific used imitation learning methods in Chapter 5, there are several. As mentioned

in the chapter, the current visual approach does not handle multiple instances of the class at the same time. While instance segmentation could address this, it is not obvious this would be sufficiently robust, nor is it obvious this is actually the best route forward. Also as mentioned in the chapter, spatial correspondence simply does not constitute a sufficient state space for some tasks (with cooking eggs as a simple example). Several other obvious limitations are with the behavior cloning strategy that was employed. For one, there is no ability to perform better than the demonstrator. There is also not a great way to use data from anything that is not a successful demonstration. These limitations, however, pertain to behavior cloning in general, and this was not the focus of our work but rather used as a tool to evaluate different visual training strategies.

There are a handful of logical next steps for research to pursue. An inevitably primary one must be thinking about the high-level decision-making at the level above small, short-horizon tasks. Something like an LSTM running a real-time control loop has been shown to be an effective way to produce surprisingly robust skills for challenging short-horizon tasks. For longer-horizon tasks and handling multi-task requests, should the robot's policy be just one big LSTM or comparable model? Or should there be more structure in how different single-task policies are composed? Additionally, there may be many ways to further leverage the visual training methods with policy learning methods that may benefit from non-demonstration data and improve with experience.

Bibliography

[1] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *CoRL*, 2018.

[2] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Labelfusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[3] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 42–49. IEEE, 2015.

[4] Abraham Bachrach. *Trajectory Bundle Estimation For Perception-Driven Planning*. PhD thesis, Massachusetts Institute of Technology, 2013.

[5] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. *Robotics: Science and Systems 2017*, 2017.

[6] Matthew T Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

[7] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[8] Abhinav Gupta, Alexei A Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, pages 482–496. Springer, 2010.

[9] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[10] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[11] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2016.

[13] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[14] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.

[15] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[16] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

[17] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[18] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.

[19] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[20] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[21] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[22] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[23] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[24] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[25] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *arXiv preprint arXiv:1704.06888*, 2017.

[26] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018.

[27] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. S. Johnson, J. Wu, B. Zhou, and A. Torralba. SegICP: Integrated Deep Semantic Segmentation and Pose Estimation. *ArXiv e-prints*, March 2017.

[28] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *arXiv preprint arXiv:1609.09475*, 2016.

[29] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[30] M. Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8, 2017.

[31] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. 2016.

[32] Hironori Hattori, Vishnu Naresh Boddeti, Kris M Kitani, and Takeo Kanade. Learning scene-specific pedestrian detectors without real data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3819–3827, 2015.

[33] Michael Firman. RGBD Datasets: Past, Present and Future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, 2016.

[34] Tomas Hodan, Pavel Haluza, Stepan Obdrzalek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. *arXiv preprint arXiv:1701.05498*, 2017.

[35] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 92–101. IEEE, 2016.

[36] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *arXiv preprint arXiv:1702.04405*, 2017.

[37] Jincheng Yu, Kaijian Weng, Guoyuan Liang, and Guanghan Xie. A vision-based robotic grasping system using deep learning for 3d object recognition and pose estimation. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 1175–1180. IEEE, 2013.

[38] SIXD Challenge.

[39] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[40] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 598–605. IEEE, 2016.

[41] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[43] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems.

[44] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[45] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[46] itSeez3D.

[47] Sense for RealSense.

[48] Skanect.

[49] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[50] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(11):2241–2254, November 2016.

[51] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5):205–215, 2014.

[52] Pat Marion. Director: A robotics interface and visualization framework, 2015.

[53] Colin Rennie, Rahul Shome, Kostas E. Bekris, and Alberto F. De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *CoRR*, abs/1509.01277, 2015.

[54] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

[55] LabelFusion. http://labelfusion.csail.mit.edu.

[56] Max Schwarz, Christian Lenz, Germán Martın Garcıa, Seongyong Koo, Arul Selvam Periyasamy, Michael Schreiber, and Sven Behnke. Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[57] A Milan, T Pham, K Vijay, D Morrison, AW Tow, L Liu, J Erskine, R Grinover, A Gurman, T Hunn, et al. Semantic segmentation from limited training data. *arXiv preprint arXiv:1709.07665*, 2017.

[58] Eric Jang, Sudheendra Vijaynarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[59] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742. IEEE, 2006.

[60] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. In *Advances in Neural Information Processing Systems*, pages 844–855, 2017.

[61] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 103–110. IEEE, 2012.

[62] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2930–2937. IEEE, 2013.

[63] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[64] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 199–208. IEEE, 2017.

[65] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2146–2153. IEEE, 2017.

[66] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *arXiv preprint arXiv:1710.01330*, 2017.

[67] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv preprint arXiv:1803.09956*, 2018.

[68] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[69] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[70] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. Robotics: Science and Systems, 2018.

[71] Ross Finman, Thomas Whelan, Michael Kaess, and John J Leonard. Toward lifelong object segmentation from change detection in dense rgb-d maps. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 178–185. IEEE, 2013.

[72] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[73] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[74] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[75] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. *arXiv preprint arXiv:1903.07593*, 2019.

[76] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

[77] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[78] Tianhao Zhang, Zoe McCarthy, Owen Jowl, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[79] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[80] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.

[81] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[82] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kPAM: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2018.

[83] 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*, pages 1422–1430, 2015.

[84] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[85] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[86] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[87] 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*, pages 270–279, 2017.

[88] Sudeep Pillai, Rares Ambrus, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *arXiv*, page 1810.01849, 2018.

[89] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

[90] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[91] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[92] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[93] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[94] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[95] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[96] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–408, 2018.

[97] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[98] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[99] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[100] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.

[101] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[102] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.

[103] Tanner Schmidt. Model-based self-supervision for fine-grained image understanding. In *PhD Thesis*, 2018.

[104] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, volume 28, page 24. ACM, 2009.

[105] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[106] Rohan Chabra, Julian Straub, Chris Sweeney, Richard Newcombe, and Henry Fuchs. Stereodrnet: Dilated residual stereo net. In *arXiv*, page 1904.02251, 2018.

[107] Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems*, pages 2059–2070, 2018.

[108] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360. ACM, 2007.

[109] AK Ushani and RM Eustice. Feature learning for scene flow estimation from lidar. *Conference on Robot Learning*, 2018.

[110] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[111] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[112] Wei Gao and Russ Tedrake. Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. *CVPR*, 2019.

[113] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[114] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *arXiv*, page 1904.07846, 2018.

[115] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[116] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[117] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[118] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[119] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[120] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9250–9256. IEEE, 2019.

[121] Pin-Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robotics and Automation Letters*, 2(2):397–403, 2016.

[122] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[123] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

[124] Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3928–3934. IEEE, 2016.

[125] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[126] Stephen James, Andrew J Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

[127] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[128] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *Robotics: Science and Systems (RSS)*, 2017.

[129] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *arXiv preprint arXiv:1906.11883*, 2019.

[130] Maximilian Sieb, Zhou Xian, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. *arXiv preprint arXiv:1907.05518*, 2019.

[131] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[132] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

[133] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[134] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *Robotics: Science and Systems (RSS)*, 2018.

[135] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 79–86. IEEE, 2017.

[136] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[137] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *arXiv preprint arXiv:1906.08880*, 2019.

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

[139] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

[140] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[141] Rohan Chabra, Julian Straub, Christopher Sweeney, Richard Newcombe, and Henry Fuchs. Stereodrnet: Dilated residual stereonet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11786–11795, 2019.

[142] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

[143] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. *arXiv preprint arXiv:1902.05542*, 2019.

[144] Avi Singh, Larry Yang, and Sergey Levine. Gplac: Generalizing vision-based robotic skills using weakly labeled images. In *ICCV*, 2017.

[145] Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585. IEEE, 2014.

[146] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5031–5037. IEEE, 2011.

[147] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[148] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[149] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[150] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[151] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016.

Appendices

Appendix A

A.1 Policy Models and Training

For MLP models, we train on only current-observation-to-current-action mappings, i.e. $\boldsymbol{a}_t = \pi(\boldsymbol{o}_t)$. Although several works use a history of low-dimensional robot observations with MLP models [39, 78], we have found LSTMs to be more effective when using a history of observations. With supervised learning validation loss on our simulation tasks as our guiding metric, we performed a significant amount of model tuning. We have found that the general hyperparameters of MLP models used in prior works in visuomotor policy learning [39] are indeed good model choices – at least, we did not find any significant changes that significantly improved model performance. The basic architecture consists of a 2-hidden-layer model with ReLu unit activations – in all shown experiments we use 128 units in each of the hidden layers. Interestingly, we found significant reduction of performance in moving from 2 to 3 hidden layers. We also found Dropout to be effective at reducing overfitting and we use a dropout probability of 0.2. Training of MLP models was performed with the RMSProp optimizer, with $\alpha = 0.9$. We found RMSProp to give better validation error than two other optimizers we tried, Adam and SGD-with-momentum. An advantage of using MLP models, when applicable for tasks, is that we can train them much more quickly than the LSTM models – for a model using ground-truth simulated data (no vision), our MLP models train in approximately 15 minutes, compared with over an hour for the LSTM models. We trained all MLP models for 75,000 steps with a batch size of 16 observation-action pairs, starting at a learning rate of $1e-4$, and decaying by a factor of 0.5 every 10,000 steps.

For LSTM models, we train within each demonstration trajectory on the observation-history-to-current-action mappings, i.e. $\boldsymbol{a}_t = \pi(\boldsymbol{o}_t^{-t:0})$. LSTM models have been validated for example in [125] to be effective for visuomotor robot policy learning. We also performed significant tuning of the LSTM models, using supervised learning test loss as our guiding metric. We arrived at a model architecture similar to [125] with a couple differences. For one, we use only one layer of LSTM, and did not find advantages of using multiple-LSTM-layers at least on our datasets. We also preprocess the input to the LSTM with a 2-hidden-layers of MLP. The architecture we settled on for all presented results uses two 100-ReLu-unit MLP layers followed by one 100-unit LSTM layer. We found Layer Normalization [147] on the activations coming into the LSTM to indeed improve generalization performance. All LSTM models were trained with RMSProp for 200,000 steps, each step optimizing via truncated backpropagation with maximum 50 steps, and with a learning rate starting at $2.0e-3$ and decaying by a factor of 0.75 every 40,000 steps. To aid in stabilizing training, we clip gradient norms as suggested in [148], in our case to have a maximum magnitude of 1.0.

For policy learning we also found, as is standard in machine learning, that normalizing inputs provided significant model performance increase. We performed this both for the MLP and LSTM models, normalizing each observation dimension to have 0-mean and unit-variance. Even when performing Descriptor Set Optimization, we can first normalize the correspondence point space based on the initialization of the descriptor set.

While [125] advocated for multi-modal policy prediction in visuomotor policies, and we agree this is a philosophically desirable property, we did not find our implementation of a Mixture-Density-Network-based multi-modal policy prediction to provide advantage over direct regression of the policy, at least on our tested tasks in simulation. None of our tasks, however, presented any obvious significant needs for multi-modal policy prediction, although other tasks might (i.e. "grab this symmetrical object from *either* the left *or* the right").

The loss function requires a choice of weighting between three components in the action space that, since they are in different units, are not obvious how to weight: translation,

rotational, and gripper components. We can represent the weighting in the loss function as

$$\mathcal{L}(\cdot) = \lambda_{translation}\mathcal{L}_{translation}(\cdot) + \lambda_{rotation}\mathcal{L}_{rotation}(\cdot) + \lambda_{gripper}\mathcal{L}_{gripper}(\cdot)$$

where the overall loss is the sum of losses that operate on only either the end-effector translational, end-effector rotational, or (one-degree-of-freedom) gripper. With these three components respectively in the units of meters, radians, and meters, we found $\lambda_{translation} = 1.0$, $\lambda_{rotation} = 0.1$, $\lambda_{gripper} = 1.0$ to give good performance.

A.2 Vision Networks

Both ***"AE"*** and ***"E2E"*** methods used an identical architecture, with the only difference being the additional decoder used for the AE method during autoencoding. The network is almost exactly as in [39] and [119], but we provided a full-width image, 320×240. We used 16 2D feature points. ***"DD"*** architecture is identical to [1]. DD-2D computes image-space spatial expectation, DD-3D computes 3D-space spatial expectation using the depth image, see Chapter 4 for details; both used 16 descriptors. The ***"E2E (34-layer)"*** network is exactly the DD architecture but with $D = 16$ and channel-wise 2D spatial softmax to obtain $\boldsymbol{z}$.

A.3 Simulation Tasks

Our simulation environment was configured to well match our real hardware experimentation. Using Drake [150], we simulate the 7-DOF robot, the gripper, and environment dynamics, and multi-view RGBD sensing of the robot's workspace. In simulation we use two cameras mounted at a downwards angle towards the table, in a mounting position similar to our hardware experiments. In our simulation tasks we use simple ground truth controllers as the expert demonstrators, which we can design based on ground truth knowledge of the object pose in simulation. Our used simulated tasks are:

Task: "Reach, translation only." Our first task is a simulated reaching task, where the

position of the object (a sugar box) is varied only in 2D translation on the table top. The robot begins at a nominal pose centered above the table and must translate in 3D towards the box. The non-vision observation $\boldsymbol{o}_{\text{robot}}$ for this task is only the 3D position of the end-effector. No rotation is observed nor included in the action space. The ground truth controller for this task simply takes in measured end-effector pose, and a target pose, which is defined relative to the ground truth object pose. At the simulated image rate (30 Hz), the ground truth controller sets the desired end-effector position 1 cm closer to the target than its current position. The varying environment configuration for this task is defined by only the object pose, and is sampled from a truncated gaussian with diagonal covariance, $\sigma_x = 0.05, \sigma_y = 0.1$ and truncated by rejection sampling to be within a rectangle that fits in the robot's workspace, specifically in the range $[[0.6, 0.7], [-0.2, 0.2]]$ meters respectively for x and y. Additionally we reject any positions for which the box is not fully visible by the used camera, which includes a small corner of this region. Samples from this distribution can be seen in Figure A-4.

Task: "Reach, with translation and rotation." Our second task is similar to the first task, except we also include rotations. In addition to sampling the box x and y position from the same truncated gaussian inside the robot's field of view, we also sample the box's rotation, θ from a uniform distribution $[-30, 30]$ in degrees. Samples from a projection of this distribution can be seen in Figure 5-5. Rotations are also added to the robot's observation space (using the full end-effector observation space described previously), and to the action space. The ground truth controller accordingly interpolates between the full measured end-effector pose and the rotation of the target pose, which is the same relative to the object as in the last task but now includes the rotation.

Task: "Push box." Our third task necessitates the need for feedback, and also involves complex frictional contacts. This task is a pusher-slider system [151] in which we constrain the robot's movement to be in the xy plane above the table. The goal of the ground truth controller is push the box across the table in y, without any specified goal in x. We hand-designed the ground truth controller based on a simple state machine. To necessitate feedback in this task, we add random disturbances to the box such that it rotates – this simulates a similar effect as if the box was sliding on an imperfect frictional surface.

The initial box position is sampled over rotation (uniform [-30,30] in degrees) and a small translational region (10 $cm \times 4$ cm) on one side of the table.

Task: "Push plate." Our fourth task is similar to the box-pushing task, but with a few critical differences. For one, the object is circular, and pushing this object yields less open-loop stability than pushing a box, as discussed in Chapter 5. Additionally, the object is fully symmetric rotationally. Another difference is that the ground truth controller pushes the plate to a specific x, y location, rather than just to a line in y as in the box task. Like the box-pushing task, we provide disturbances both during training and during evaluations.

In the first two tasks we use the left camera to train policies, although either could have been used. In the second two tasks, we use the right camera to train policies, since there is significant occlusion for the left camera in both of these tasks. In all tasks both cameras were used for correspondence training.

A.4 Additional Simulation Results

Qualitative Visual Feature Inspection

Inspecting the features learned by the different vision-based methods, we observe that both the spatial autoencoder and end-to-end methods are able to produce features that appear to track consistent locations on or near the object (Figure A-1). Note that not always may we interpret features superimposed on top of the image, but due to the spatial softmax architectures used for these methods, the interpretation is that for at least some of the features they will identify features in the image [39, 119]. While some features appear to be related to the object location, many other features do not. The end-to-end method can produce stable features during the robot's hand movement across the image, but since the spatial autoencoder must be able to use the bottleneck representation to reconstruct different images, the features will move in response to any movement in the image, for example from the hand. Our method using a descriptor set of Dense Descriptors, since it is trained to perform correspondence for the object, will track consistent locations on the object.

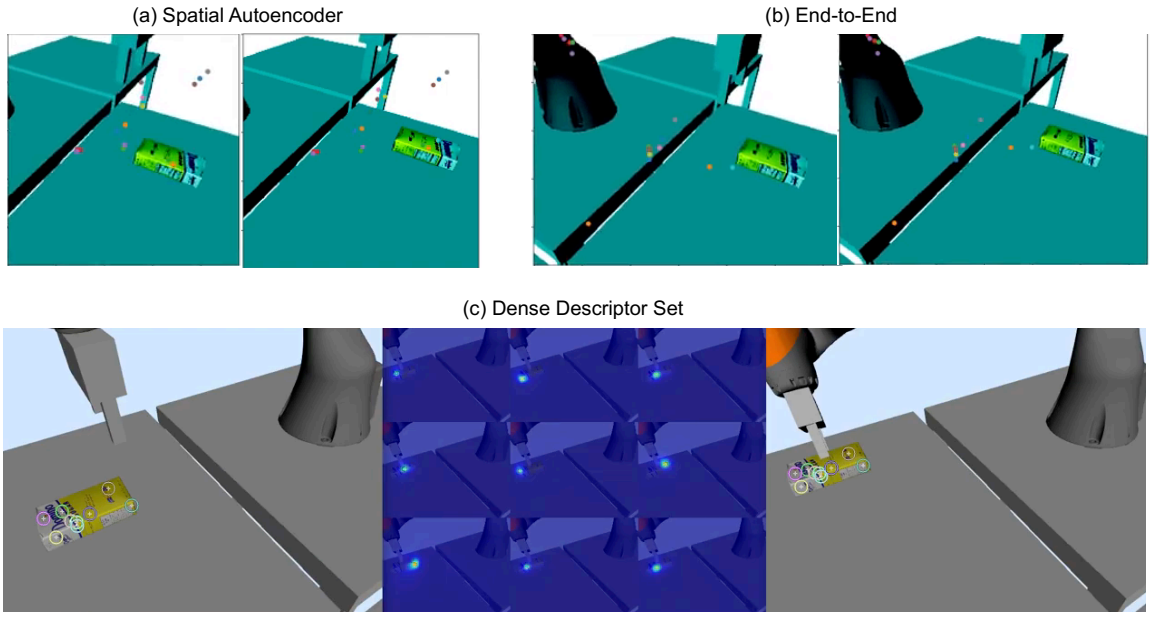


Figure A-1: Feature visualizations of the different methods. For (a) the spatial autoencoder, the orange feature on the box appears to identify the box location, as shown in two different images. Similarly for the end-to-end method, the blue feature near the left corner of the box appears to identify the box location. For the Dense Descriptor method, a set of predicted locations is shown in a reference image (left) and a target image (right), for an example descriptor set of nine descriptors. In the middle is shown the heatmaps of correspondences for these descriptors.

Additional Quantitative Results

On this task, we observe that our proposed method using Dense Descriptors nearly matches the performance of the ground truth methods and provides better performance than the two benchmarked vision-based alternatives. In this case the ordering between the 5 benchmarked options is essentially consistent between both the supervised learning validation loss, and with the closed-loop task success metric γ_p. Although it is difficult to interpret how meaningful the scale of the differences are with the supervised learning validation loss, they are more tangible for the task success metric. In this case γ_p measures the distance between the ground truth target location and the final end effector pose – specifically, $\gamma_p = \min(||p_{target} - p_{end-effector}|| + \epsilon, 0)$ in cm, where we allowed an ϵ of 1.2 cm. All methods for this task were trained on 200 demonstrations of the ground truth controller.

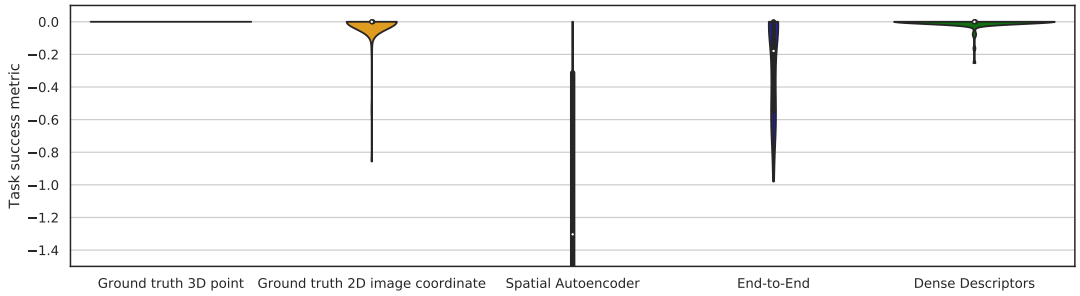


Figure A-3: Distribution of the task success metric γ_p on the "Reach, translation only" task for the different bechmarked methods.

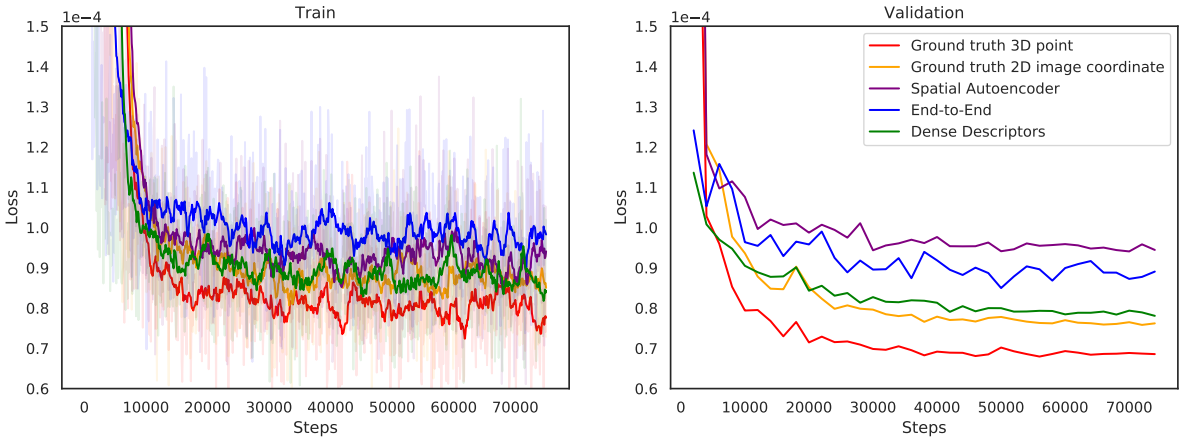


Figure A-2: Comparison of train and validation loss for the different visuomotor models trained on the "Reach, translation only" task. For clearer visualization, the training loss has been smoothed with a simple exponential filter, with $\alpha = 0.9$. The unsmoothed data is plotted semi-transparently.

The ground truth 3D world state estimate, as would be reasonably expected, provides the best ability to learn effective policies. The validation error is notably lowest for this baseline (Figure A-2), and in closed loop performs perfectly on our metric γ_p, terminating within ϵ of the target on 100% of tested object locations (Figure A-3). For the baseline where this ground truth 3D point is projected into the camera and provided as an image-space coordinate, there is a small drop in generalization performance – 94% of tested object locations terminate within ϵ (Figure A-3). As can be seen in Figure A-4, the locations that are imperfectly generalized are in the corner of the distribution, where there were no training examples (which are displayed with gray "x"s overlaid in the plot). Our method using Dense Descriptors provides nearly identical performance as if the policy learning was

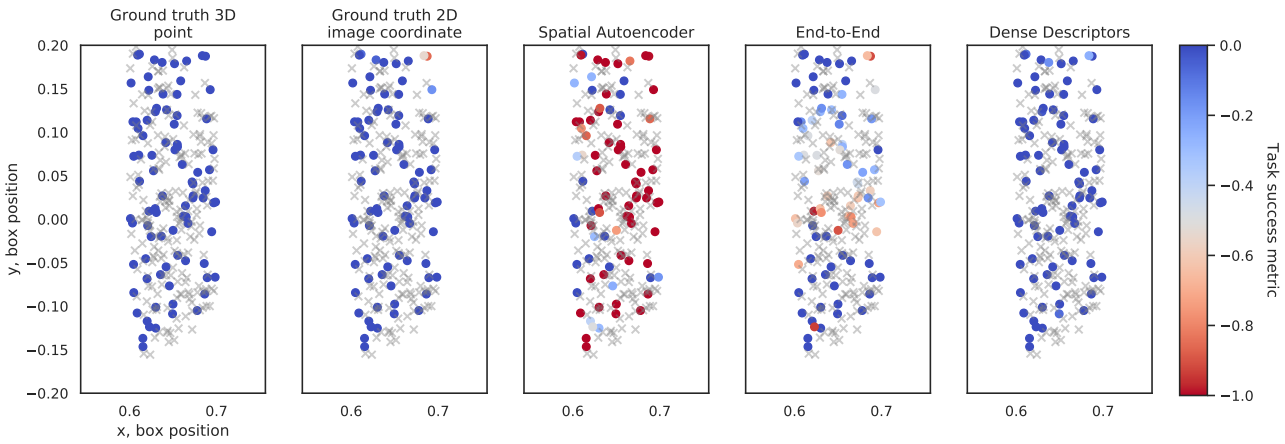


Figure A-4: Spatial distribution of task success metric γ_p on the "Reach, translation only" task for the different bechmarked methods. Each colored point displays the γ_p for deploying the trained policy with the object at that x, y location. The 200 training examples are overlaid as gray "x"s. The bottom right corner of this distribution shows where we did not use points that were outside of the used camera's field of view.

provided the ground truth image coordinate – the validation loss is slightly larger (Figure A-2), while the closed-loop performance is essentially identical, with also 94% of tested object locations terminating within ϵ of the goal (Figure A-3). More specifically, Figure A-4 shows that the same object locations that were imperfectly generalized for the ground truth image coordinate baseline are the same locations imperfectly generalized for the Dense Descriptors.

For the vision-based baselines, end-to-end optimization performs notably well, with every tested object location terminating within 1 cm of the ϵ ball around the target location (Figure A-3). The performance is not as high, however, as the Dense Descriptors method, with only 43% of tested locations terminating inside ϵ of the target. In terms of the spatial distribution of its performance (Figure A-4), the End-to-End method oddly has a region of measurably worse performance in the center of the training distribution, while performing better on some of the edges. We were not able to train policies as effectively using the frozen autoencoder visual features – the average test terminated over 1.2 cm outside the ϵ ball, with a large fraction of the distribution being several cm outside (Figure A-3). Since the End-to-End method provided a much more competitive baseline to our method, we focus on the comparison with End-to-End in our subsequent evaluations.

"Reach, translation and rotation". On the task involving rotations as well the relative

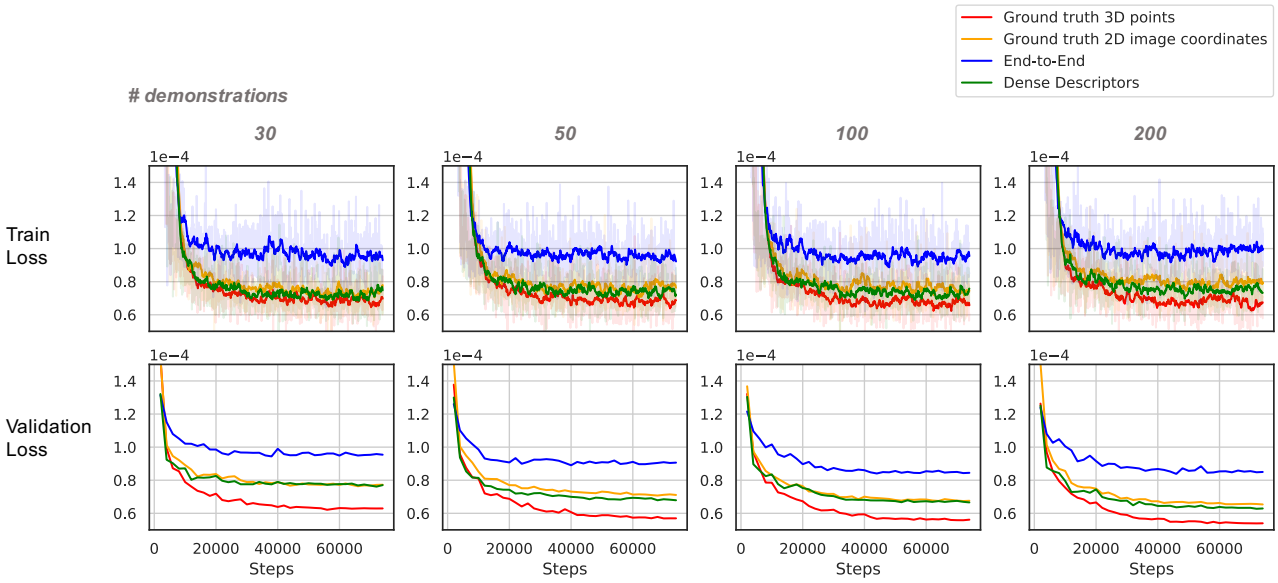


Figure A-5: Comparison of train and validation loss on the "Reach, translation and rotation" task for a sample complexity sweep with $\{30, 50, 100, 200\}$ demonstrations for each of the different four most competitive methods.

ordering of the methods' performances are similar, although there is a notable decrease in performance for the End-to-End method in addressing rotation, whereas the Dense Descriptor method and unsurprisingly the ground truth methods do not struggle to handle rotation. Compared with the shown results for the "Reach, translation only" task, when training on rotations as well and also with 200 demonstrations there is a more pronounced difference for the End-to-End method compared to the others, both measured by supervised learning validation loss (Figure A-5, see for 200 demonstrations) and in closed loop performance (Figure A-6, see for 200 demonstrations). The metric $\gamma_{\boldsymbol{T}}$ for this task now also includes a rotational component such that $\gamma_{\boldsymbol{T}} = \min(||p_{target} - p_{end-effector}|| + \lambda_r(\Delta\alpha) + \epsilon, 0)$, where λ_r scales between translation and rotation, and $\Delta\alpha$ is the angle difference between the target and end-effector rotations. On this task we also perform a sample complexity sweep, training the methods on either 30, 50, 100, or 200 demonstrations and testing their generalization performance. All evaluations used the same test locations for generalization tests, regardless of how many demonstrations were used. In Figure 5-5 the distribution samples in the different training sets can be seen by comparing the gray "x"s for (top to bottom) 30, 50, 100, and 200 demonstrations. The training sets used monotonically increasing data,

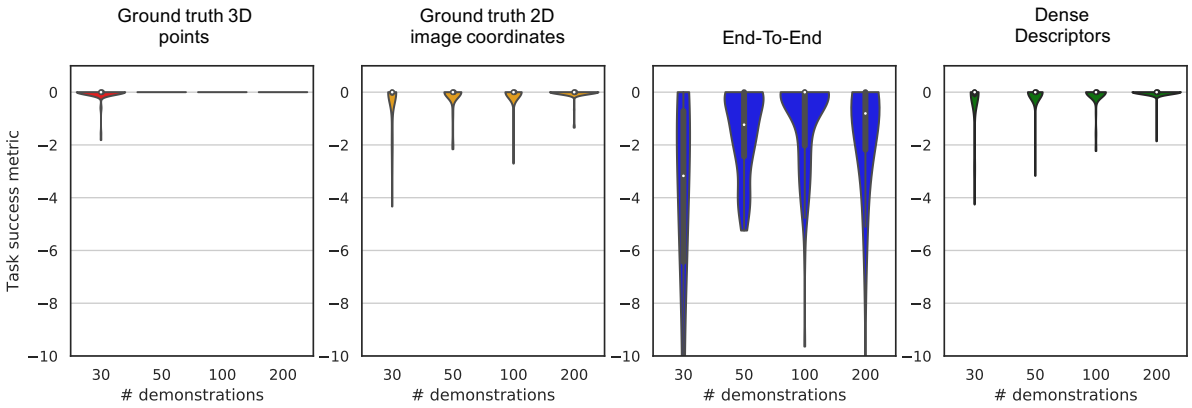


Figure A-6: Comparison of the distribution of task success metric outcomes (rewards) for 30, 50, 100, or 200 demonstrations for the different tested methods.

i.e. the 100-demonstration dataset contained all of the logs in the 50-demonstration dataset plus an additional 50, etc.

The sample complexity sweep shows that, as measured both by supervised learning validation error and by the closed-loop task success metric $\gamma_{\boldsymbol{T}}$, that using Dense Descriptors we are able to train on only 30 demonstrations and achieve better generalization performance than using End-to-End with 200 demonsrations. Figure 5-5 displays the spatial distribution of $\gamma_{\boldsymbol{T}}$ for the tested methods. Again we observe a remarkable similarity between using the Dense Descriptors and using the ground truth image-space 2D coordinates. When observing the convex hull of the used training examples for each of the $\{30, 50, 100, 200\}$ demonstrations, we see that for the most part Dense Descriptors, like the ground truth image-space coordinates, can enable policy learning that generalizes well inside the convex hull of the training examples. Both Dense Descriptors and the ground truth image coordinate methods struggle to perform extrapolation outside of the convex hull. Interestingly the ground truth 3D coordinates enable much better extrapolation.

A critical question, however, is to investigate whether the differences in results are related to (a) the training method, or (b) the model complexity. In particular, the presented dense visual correspondence models use a 34-layer convolutional architecture with 21 million parameters. In contrast, the visual baselines have used only 3-layer convolutional architectures, with approximately 3 orders of magnitude less parameters, at 86,000. To test whether the differences in results are due to (a) or (b), we took the exact architec-

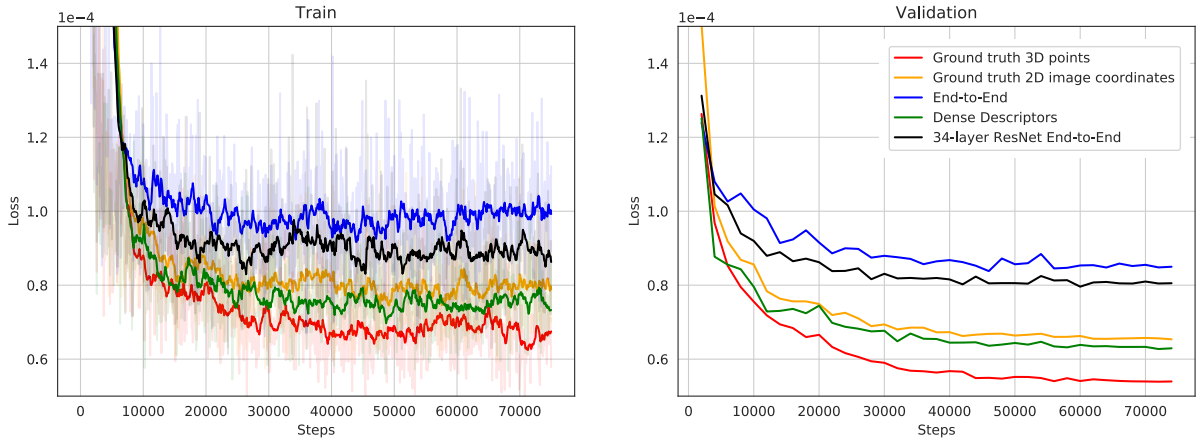


Figure A-7: Comparison, including a 34-layer End-to-End-trained model, of train and validation loss for the different visuomotor models trained on the "Reach, translation and rotation" task. For clearer visualization, the training loss has been smoothed with a simple exponential filter, with $\alpha = 0.9$. The unsmoothed data is plotted semi-transparently.

ture (34-layer ResNet) used for the Dense Descriptor method, and chose a $D = 16$ final full-resolution output channel depth upon which a 2D spatial expectation was computed to produce a $Z = 32$ representation $\boldsymbol{z}$. This is essentially a $>$ 10x deeper version of the End-to-End and Spatial Autoencoder architectures. We trained this 34-layer ResNet end-to-end for 22.5 hours (54x longer than our method) and found that there was a small advantage in validation loss over the smaller end-to-end architecture, but not comparable to the Dense Descriptors method using correspondence training (Figure A-7). This suggests it is (a) correspondence training, and not the deeper architecture, which provides the advantage.

A.5 Hardware Tasks

Task: "Push sugar box." This task is similar to our simulated pushing task, and the goal is to push the box across the table to a finish line. Rather than use a ground truth controller as done in simulation, however, instead the human demonstrator follows a similar strategy as the simulated ground truth controller. The robot's motion is constrained to the xy plane above the table in this task, and to make the system more unstable and hence require more feedback, we leave the gripper closed such that there is a small area of contact. Note that it would be much easier for this task to use the fingers in a widespread configuration. The initial configuration for the box used during the demonstrations is similar to the used

distribution in the simulated task. In this task, in a large fraction of demonstrations a human intervenes with a toy gripper and provides disturbances to the box. This puts reacting to disturbances in the training set.

Task: "Flip shoe, one instance." This task is similar to our used flipping task in simulation. The goal for this task is to dynamically flip the shoe upright. A single shoe is used, and it is placed on the table such that the only variation is its two-dimensional translation on the table top and rotation along one degree of freedom (approximately in the range of +/- 30 degrees). The shoe always starts on the same side. We do not provide disturbances during demonstrations of this task.

Task: "Flip shoe, class-general." This task tests the ability of the learned policies to generalize across a class of objects with some shape and appearance variation among instances of the class. The task is similar to the previous task, except rather than use only one shoe, we demonstrate shoe flips with a training set of 14 shoes.

Task: "Pick-then-hang hat on rack." This task tests the ability of our method to handle deformable objects. The goal of this task is to grab a hat and place it on a short rack on the table. To add to the challenge, in the demonstrations we grab the hat near the bill, such that the rear of the hat will be able to sufficiently deform relative to the rigid grasp of the hat, and hence test our ability to react to meaningful deformable configuration changes of the object. During demonstrations of this task, disturbances are provided to the hat's deformable configuration.

Task: "Push-then-grab plate." This task presents several unique challenges compared to the other tasks. Part of the challenges are visual: for one, the object is rotationally symmetric, textureless, and glossy, and in this task we provide demonstrations with a wide variety of backgrounds and clutter. The robot must first reach for the plate, and then push the plate across the table. Due to physical clutter on the table, the robot must react to how clutter affects the trajectory of the plate. The final portion of the task involves contact-rich manipulation with complicated frictional contacts to grab the plate. The robot pushes the plate up against a wall, lifts one edge of the plate against the wall, slides into a grasping

position, grasps the plate, and then lifts. This task is inspired from a recent demonstration[1] using a robust feedback controller synthesized specifically for this task.

[1]Toyota Research Institute, "Introducing TRI Manipulation for Human-Assist Robots", https://youtu.be/LgaFkWCtSGU?t=56