Learning Visual Representations from Cross-Modal Correspondence

by

Mohamed El Banani

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2024

Doctoral Committee:

Assistant Professor Justin C. Johnson, Chair
Professor Alexei A. Efros, University of California, Berkeley
Assistant Professor David F. Fouhey, New York University
Assistant Professor Andrew Owens
Professor Stella Yu

Mohamed El Banani

mbanani@umich.edu

ORCID iD: 0000-0003-4686-6048

To my grandparents

Acknowledgments

The PhD journey is very long and, like life, is accented with success, failure, and everything in between. Thank you to everyone who shared this journey with me. While it would be difficult to mention everyone, below is my humble attempt.

To Justin Johnson, thank you for providing me with the support, encouragement, and freedom to pursue my ideas. Justin has consistently pushed me to think bigger and aim higher. More importantly, he leads by example. Justin showed me how to be critical and kind and excel in the lab without forgetting the life outside of it. I am grateful for your guidance and support, and I am excited for our next adventure.

To David Fouhey, thank you for showing me how to be a better researcher, teacher, and mentor. Our hallway discussions are some of my most cherished memories from graduate school. I am inspired by your generosity and kindness, and I hope that I can pay it forward.

Thank you to my committee for their insight, flexibility, and time. Stella Yu's clarity of thought and broad view of vision research have been inspiring, and I thank her for always pushing me to broaden my views through engaging debates. I thank Andrew Owens for inspiring me to be more creative and pursue machines that can fully perceive their environment, not just see them. Finally, I thank Alyosha Efros for inspiring much of my thought on self-supervised learning and correspondence. The ideas presented in this dissertation can be traced directly to your papers, and I've learned even more from our discussions.

During my PhD, I spent two summers at FAIR and Google Research. I thank my mentors, Ben Graham and Varun Jampani, for their support and patience. Thank you to Leo Guibas for many truly insightful and enjoyable discussions. Thank you to my collaborators, especially Ignacio Rocco, Amit Raj, and Kevis-Kokitsi Maninis. Finally, I thank Prafull, Sarah, Shivam, Maxwell, Junhwa, Mia, and Tuzi for making my internship much more fun.

I am indebted to my undergraduate mentors who got me started in research when I knew nothing: Omer Inan, Todd Sulchek, and Ashok Goel. Special thanks to Maithilee Kunda, who believed in me and taught me so much about the nuances of cognition, perception, and research. I have also relied on the advice of mentors who were always generous with their time: John Laird, Susan Gelman, Karem Sakallah, Chad Jenkins, and Ahmed Saeed. Thank you!

Research can be a lonely journey, but I've been fortunate to share it with some brilliant researchers. I am thankful to Zakaria Aldeneh and Mahmoud Azab for research arguments between squash matches; Karan Desai for teaching me so much about language, coding, and board games; Richard Higgins for entertaining my philosophical musings; Nilesh Kulkarni for many 3D debugging sessions; Fahad Kamran and Sarah Jabbour for always being there for a good laugh; Preeti Ramaraj for being a perpetual source of support and motivation; Steven Jones for debates about intelligence; Matt Perez for quiet chats and loud laughs; and Max Smith for constantly challenging me to be better.

I have been part of a great community of vision researchers at Michigan: Karan Desai, Nilesh Kulkarni, Ang Cao, Chris Rockwell, Tiange Luo, Gaurav Kaul, Jeongsoo Park, Richard Higgins, Dandan Shan, Sarah Jabbour, Shengyi Qian, Linyi Jin, Daniel Geng, Ayush Shrivastava, Ziyang Chen, Yiming Dou, and many more. I have learned a lot from you, and I can't wait to see what you do next.

Special thanks to ARC-TS and DCO for keeping the clusters running and saving us when things went down: Brock Pallen, Matt Britt, Don Winsor, and Liz Zaenger.

During the pandemic, I was lucky to share Prospect House house with some truly wonderful housemates: Tim, Lauren, Jonathan, Melanie, Ian, Yan-Rou, Alejandro, Matt, Bubs, Max, and Richard. Thank you for all the memories made over meals, parties, and tailgates.

One of the (mis)fortunes of constant travel is having friends all over. I am grateful for all those who were always there despite being far away: Omar El Sayed, Zeena Ammar, Yehia Zalat, Sarah El Houcheimy, Samer Heiba, Cody McCord, Mohamed Shafik.

I couldn't imagine surviving Michigan winters without my friends. Special thanks to the Egyptians, especially Essam Osman, Omar Barky, and Mustafa Zaky. Thank you to Samer and Noura for exploding kittens in Qahwah house. Thank you to my volleyball pals for some great rallies: special thanks to Dan, Natasha, Prince, Alex, Mark, April, Joey, Kixin, Harsha, and SM. I thank Shurjo, Eric, Ariel, Madan, Sharmi, and Ted for laughing together through rough times. Last but not least, I thank Hayley for her unwavering optimism and kindness in the face of everything and for brightening my world.

Finally, I thank my family for their perpetual support and understanding. My parents, Samir and Maha, always put my education first and put me in front of a computer since I was two years old. Thank you for your selfless dedication. I would not be here without your many sacrifices. I thank Menna for always being there and helping me make clearer and prettier figures. I thank Teta and Nano for their almost irrational belief in me and for inspiring me to be a kinder human being. I especially thank my grandfather, Mahmoud El Emary, for instilling in me the importance of good questions, clear thoughts, and simple explanations. I wish you were still with us today.

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Acronyms

CNN Convolutional Neural Network

DOF Degrees of Freedom

GART Geometry-Aware Ratio Test

ICP Iterative Closest Point

kNN k-Nearest Neighbors

LG-SSL Language-Guided Self-Supervised Learning

MLP Multi-layer Perceptron

RANSAC Random Sample Consensus

ResNet Residual Convolutional Neural Networks

RGB-D Red, Green, Blue, Depth

SE(3) Special Euclidean Group in three dimensions

SfM Structure from Motion

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SO(3) Special Orthogonal Group in three dimensions

SSL Self-Supervised Learning

t-SNE t-distributed Stochastic Neighbor Embedding

Abstract

One of the goals of computer vision is to develop visual agents that can learn without human annotation. This is typically done by learning from images and their augmentations. In contrast, humans learn from dynamic and multi-sensory environments without requiring such explicit supervision. My dissertation delves into this contrast, exploring how models can learn visual representations directly from their environments. My core observation is that such environments, despite their complexity, present consistent patterns across modalities. These cross-modal patterns offer a rich training signal as we can leverage similarity in one modality for learning generalizable representations in another without requiring additional supervision.

In this dissertation, I argue that cross-modal correspondence provides a rich signal for learning visual representations and a useful tool for analyzing them. I first discuss how models can learn visual representations by finding 3D correspondence in RGB-D videos. Through estimating geometrically consistent correspondences between video frames, models can learn representations that rival supervised models. I then discuss how the notion of correspondence could be applied to language. I propose language-guided self-supervised learning, where language models are used to find image pairs that depict similar concepts. I show that using language guidance outperforms self-supervised and language-supervised models; further showcasing the utility of learning from correspondence. Finally, I explore how correspondence can also be used to analyze the 3D awareness and consistency of visual representations learned by large-scale vision models. My analysis suggests that while current approaches yield good models for semantics and localization, their 3D awareness remains limited.

Chapter 1

Introduction

While my eyes perceived no interval,
my mind preserved an abyss.

—Marcel Proust, *In Search of Lost Time*

Our minds form coherent and stable representations of the world from noisy and inconsistent sensory inputs. As you walk around a city, your visual input fluctuates widely due to changes in your perspective and the environment. Despite this, your understanding of the city and its dwellers remains consistent. Beyond consistency, you can recognize objects and spaces, estimate their shape, and group them into meaningful categories for efficient communication and decision-making. This process is so effortless that early theories of vision focused solely on the sensory optics of the eye [169], yet decades of computer vision research reveal that the challenge lies in interpreting what the eye senses.

One of the core challenges in computer vision is how to convert images—two-dimensional arrays of light intensity values—to representations that are useful for downstream tasks. Early work developed hand-crafted features based on wavelets [186] and gradient orientations [58] that capture low-level details in the vision. While such features enabled early advances in object detection [58] and structure for motion [243], their low-level nature made them less effective for high-level visual tasks such as recognition and segmentation.

The next advancement came in moving from representation design to representation learning. This approach extends back to early work on associative learning [86, 104, 230], but it regained popularity when AlexNet [147] won the ImageNet competition [63]. The core idea is that the *correct* representation is one that minimizes the task prediction error [234]. Interestingly, such representations were also useful for other visual tasks [69, 250]. Nevertheless, the impressive performance of supervised learning came with the cost of data annotation, which is often expensive and time-consuming. It is also unclear if such extensive annotation is needed as humans exhibit incredible visual abilities despite never being trained with explicit labels, object masks, or 3D meshes.

To relieve the need for explicit annotation, self-supervised approaches that only learn from images were proposed. This idea can be traced back to the Neocognitron [86] which learned "without a teacher." Broadly speaking, there have been two general approaches to self-supervision: generative and discriminative. Generative approaches posit that good representations are ones that capture the distribution of images. The intuition here is straightforward: if you can generate images, you understand them. This has been done through colorization [344], autoregressive modeling [204], auto-encoding [103, 298], and text-conditioned generation [229].

In contrast, discriminative approaches posit that models can learn good representations by learning to discriminate between images. While the idea can be traced back to Hadsell et al. [98], it has seen a recent revival with a range of work that learns representation through instance discrimination [40, 102, 315]. Instance discrimination builds on the success of classification-based pre-training and treats each instance as a separate class. As a result, one can learn through augmenting images and training a model to associate augmented versions of the same image with each other. Given that we can learn good representations just from images, can we extend self-supervision to the kind of multimodal data that any visual agent would have access to?

1.1 Multimodal Consistency and Correspondence

This dissertation is primarily concerned with how to learn visual representations. However, we first might ask: what should visual representations "represent"? This old question goes back to the earliest theories of vision [169]. More recently, it has seen a revival in debates about the veridicality of perception [110] and the nature of mental imagery [290].

One idea that has shown up in both cognitive science and machine learning is the concept of consistent representation. Shepard and Chipman [252] argued that the relationship between external objects and internal representation is best characterized as a *second-order isomorphism*: *i.e.*, the similarity between representations should correlate the similarity between the external objects. The idea also lies at the core of the discriminative visual representation learning and was made explicit by Hadsell et al. [98]. However, the theory remains unclear on what forms of similarity matter between objects, resulting in different forms of discriminative representation learning. Supervised learning matches a classification ontology with models trained to generate consistent representations for members of the same class. Self-supervised approaches eschew a global structure for a much smaller

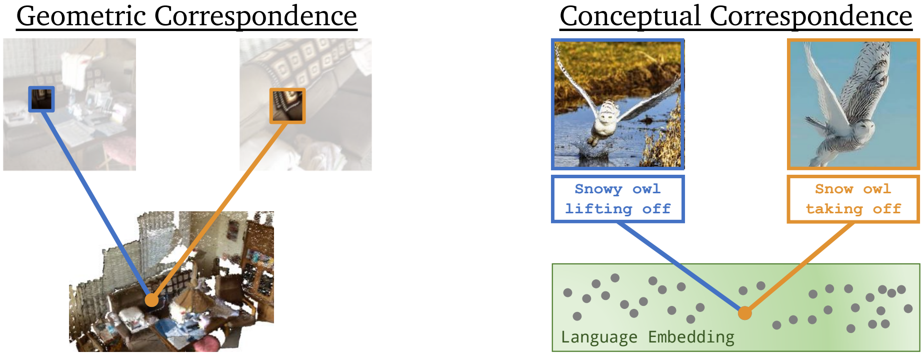


Figure 1.1: This dissertation explores two types of correspondence: geometric correspondence in 3D space and conceptual correspondence in language embedding space.

neighborhood defined by the image and its augmentations. However, this formulation is not as assumption-free as it first seems since the choice of augmentation [318] and curation of the dataset [10] influence the quality of the learned representations. While both formulations have been successful, can we find a better learning signal than a single all-encompassing ontology that aims to explain the relationship between all images or a set of image augmentations that ignore those relationships?

We argue that the answer lies in looking towards our environment and learning from cross-modal consistency. The world manifests itself in different, but *consistent*, ways across sensory modalities. Consider an encounter with a friend's kitten; as you walk into the room, you see him from many different angles and in different contexts as he plays around. Beyond sight, you also get information in multiple other modalities: *e.g.*, haptics, depth, language, etc. As a result, you can also identify the kitten from his purr, fur, or the many references provided by your friend. We posit that such cross-modal relationships, while not perfectly correlated, are a valuable training signal for visual representation learning.

We propose to take advantage of this structure by identifying and learning from cross-modal correspondence. The concept of correspondence arose in research on stereo-vision, where it meant two image patches that depict the same 3D point [129, 188]. However, correspondence need not be limited to 3D. We can conceive of a generalized notion of correspondence that encompasses multisensory inputs that we discussed earlier. In this dissertation, we define correspondence more broadly as two sensory inputs that depict the same entity in a different modality. This provides us with different types of correspondence; *e.g.*, sounds coming from the same instrument or creature, image patches depicting the same point in space, or even images that depict the same animal species or visual concept.

1.2 Dissertation Outline

In this dissertation, we argue that cross-modal correspondence provides a rich consistency signal for learning visual representations and a valuable tool for analyzing them. We explore two forms of correspondence, as shown in Fig. 1.1: **geometric correspondence** are visual inputs that depict that same 3D point, and **conceptual correspondence** are images that depict the same concept. The dissertation is organized as follows:

In Chapters 2 to 4, we explore how we can learn from 3D correspondence in RGB-D video. We observe that close-by video frames capture closely related views of the same scene, which can provide a rich signal for learning visual representations.

In Chapter 2, we explore learning correspondence by relying on photometric consistency. It is based on the idea that if a model predicts accurate correspondence between two views, it should be able to accurately register and render the scene. We propose a differentiable registration and rendering approach for learning point cloud registration which outperforms prior supervised approaches despite only learning from RGB-D video. We find that by integrating principle components into the learning pipeline, we can learn representations that outperform supervised training. This chapter is based on work previously published in [77].

In Chapter 3, we further simplify our approach by only relying on geometric consistency. We observe that randomly initialized convolutional neural networks are already effective feature extractors. While their correspondences can be very noisy, a robust estimator can still estimate a coarsely accurate transformation. As a result, we can train models by constructing a loss function that penalizes correspondences that deviate from the estimated transformation. This means that we can greatly simplify the setting, achieve better performance, and learn both image-based visual features as well as point cloud-based geometric features. This chapter is based on work previously published in [75].

In Chapter 4, we go beyond image pairs and explore learning correspondence via multiview registration. The methods proposed in Chapters 2 and 3 assume partial overlap between view pairs. This is ensured by sampling at a relatively high frequency (1.5 Hz), which also limits the viewpoint change between frame pairs. We overcome this approach by learning multiview synchronized registration, allowing us to train with larger viewpoint changes and further improve performance. This chapter is based on work previously published in [79].

In Chapter 5, we go beyond the traditional formulation of correspondence and explore how language can provide us with conceptual correspondence. Intuitively, when people observe similar scenes, they tend to describe them in similar ways. We can use language models to sample image pairs that correspond to the same caption and use them for visual representation learning. We observe that language-guided pairs provide naturally augmented versions of the same visual concept and yield better visual representations than existing self-supervised and language-supervised approaches. This chapter is based on work previously published in [78].

Over the past five years, the area of visual representation learning has grown at an incredible pace with several large-scale models exhibiting interesting capabilities that go beyond their training objective. Specifically, these models learn representations to perform well on both semantic and localization tasks such as classification and segmentation. However, it remains unclear if such models understand the 3D world or are good models of its 2D projections. In Chapter 6, we analyze the 3D awareness of visual representations through the lens of correspondence. The basic idea here is that the consistency we rely on for correspondence is both a learning signal and a desirable property in representations. Our analysis reveals mixed results showing that large-scale models appear to encode 2.5D properties such as depth and surface normals, but still struggle with 3D consistency.

Finally, we take a step back in Chapter 7 to discuss the contributions of this work, place them in the larger context of modern-day computer vision, and sketch out some future directions.

Chapter 2
Learning from Photometric Consistency

We begin our journey by considering how we can learn directly from video. As we move around, we perceive the same scene from multiple closely related viewpoints. Yet, our understanding of the scene is a single consistent whole, rather than a collection of partial views. The ability to align the partial views is both something we expect a visual agent to be able to do as well as a useful task for learning visual representations. We use this observation to learn dense visual features by estimating 3D correspondence between video frames and aligning the scene.

In this chapter, we propose UnsupervisedR&R: an end-to-end unsupervised approach to learning point cloud registration from raw RGB-D video. Given two RGB-D views of the scene, our model extracts visual features from the images, estimates correspondence, aligns the partial views, and then finally renders the point cloud from the estimated input views. The core intuition is that if the correspondences are accurate, then the resulting renders should match the input views and the correspondences should be consistent with the estimated transformation. Our approach combines representation learning and differentiable rendering with older and more principled techniques such as Lowe's ratio test [179] and Ummeyama's algorithm [292]. Furthermore, through the use of differentiable methods for alignment and rendering, we can ensure that our model learns end-to-end directly from RGB-D video without requiring any pose or correspondence supervision.

We evaluate our approach on indoor scene datasets and find that it can learn good features that enable accurate correspondence estimation and point cloud registration. Interestingly, our approach even outperforms supervised point cloud registration approaches, despite not requiring any supervision. Our results showcase the rich signal available in RGB-D video and support our thesis that cross-modal correspondence allows us to learn without requiring human annotation or additional supervision. This chapter is based on work previously published in [77].

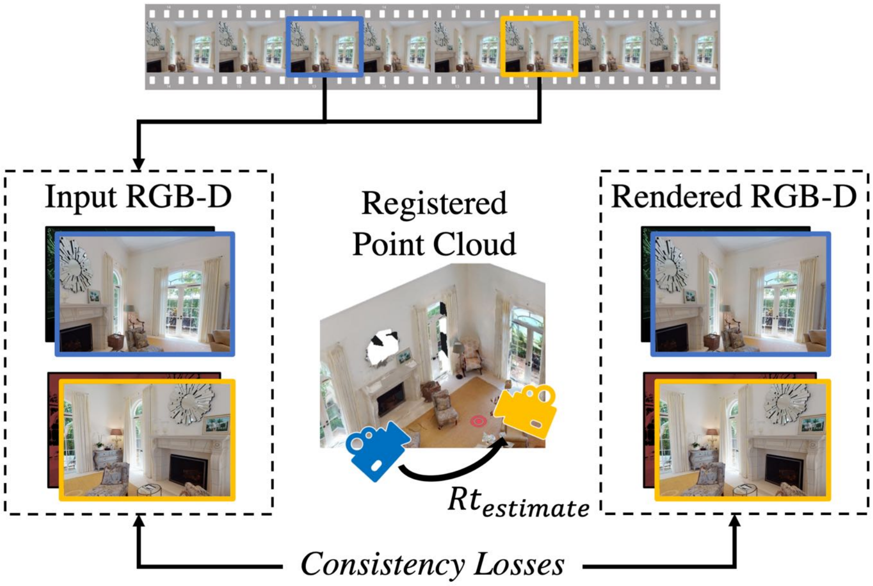


Figure 2.1: What 3D scene do the two views on the left portray? Given 2 RGB-D images, we train a model to estimate the camera motion between them through enforcing photometric and geometric consistency losses on point cloud renderings of the scene.

2.1 Introduction

Consider the two scenes depicted in Figure 2.1. How are they related? What is the layout of the room they depict? Aligning partial views of a scene into a single whole is essential to understanding one's environment and is a key component of numerous robotics tasks such as SLAM and SfM. Recent approaches have leveraged supervised learning to develop end-to-end systems that outperform traditional methods in both accuracy and speed [51, 89]. However, with the rising prevalence of cameras with depth sensors, we can expect a new stream of raw RGB-D data without the annotations needed for supervision. *How can we leverage this data for unsupervised learning of point cloud registration?*

The common approach to point cloud registration relies on correspondence estimation and geometric model fitting. Traditional approaches rely on hand-crafted features [125, 179] and robust estimators such as RANSAC [82]. While those approaches work well, their performance is limited by their inability to flexibly adapt to different data distributions. Recent work leverages supervised learning to address those limitations by learning to extract feature descriptors [50, 66, 329], finding better correspondences [51, 89, 241], and training more efficient robust estimators [26, 27, 221]. However, accurate pose annotation can be challenging to attain automatically due to sensor error or reliance on traditional SfM pipelines with no convergence guarantees [243].

Meanwhile, self-supervised visual learning has made remarkable progress in learning semantic [64, 68, 87, 94, 280] and 3D [120, 148, 289, 293, 354] features. The key idea is to use natural transformations in the data as indirect supervision. RGB-D video provides us with this supervision since successive frames capture different views of the same scene. In this case, aligning two point clouds from nearby frames is not only about achieving good geometric consistency but also showing good photometric consistency between the two views. By achieving both photometric and geometric consistency, we can train our model using RGB-D image pairs without relying on additional supervision.

We propose using view synthesis between RGB-D images as a task for learning point cloud registration. Given two RGB-D video frames, we extract features from each frame to generate a feature point cloud, where each point is represented by both a 3D coordinate and a feature vector. The extracted features serve as descriptors for correspondence estimation. The model is trained end-to-end using photometric and geometric consistency losses between the input and rendered frames. Through using differentiable components, we back-propagate the losses to the feature encoder to learn features that allow us to estimate unique correspondences and accurately register the two views.

We evaluate our model on ScanNet [56], a large indoor scene dataset. Our model outperforms the traditional registration pipeline with visual or geometric descriptors. Furthermore, it performs on par with supervised geometric registration approaches despite being unsupervised; supporting our claim that RGB-D self-supervision can alleviate the need for pose annotation. Finally, we analyze our model through several ablations.

2.2 Related Work

Feature Descriptors. Early work on feature point extraction can be traced back to using corners for stereo matching [197]. This work culminated in patch-based feature 2D descriptors [15, 179, 233] and geometric features based on histograms of local 3D relationships [125, 236]. Those descriptors have been very popular due to being efficient to compute, relatively robust, and data-agnostic. More recently, there has been an interest in leveraging convolutional neural networks to extract good visual descriptors [66, 70, 100, 329, 335] and geometric descriptors [13, 52, 60, 62, 88, 165, 305, 327]. Relevant to our work are approaches that use geometric transformations to learn visual features. This has been commonly done by using known pose or correspondences between large collections of images [66, 70, 329] or point clouds [52, 60, 62, 88, 305]. We extend this work by leveraging the transformations between RGB-D video frames and relying on consistency losses instead of pose supervision.

Correspondence Estimation and Fitting. Early work on image and point cloud registration assumes perfect correspondences [8, 176]. ICP relaxes this assumption for closely aligned points by introducing the simple heuristic of assuming the closest point is the correspondence [347]. However, extending to real-world settings requires the ability to determine such correspondences from the raw input or extracted features. Early work uses feature similarity and heuristic approaches to determine correspondence and robust estimators such as RANSAC to handle noise and outliers in the correspondences [179, 285, 348]. For a review, see [215]. More recent approaches advance this idea by learning differentiable functions for weighting correspondences [26, 27, 51, 89, 115, 180, 221, 241, 328]. Finally, there have been recent self-supervised approaches for registering object point clouds [4, 108, 115, 304, 305, 328, 334]. Those approaches operate on dense point clouds that are either augmented and sampled for partial views with known pose and correspondences. Hence, while the setup might be self-supervised, the methods still require ground-truth annotation. We are inspired by this line of work but differ from it in two key ways: (1) we take RGB-D images as input, not keypoints or 3D scenes; (2) our approach is unsupervised, while those approaches require pose or correspondence supervision.

Differentiable SfM. There has been a large number of recent approaches that replace the traditional SfM pipeline with end-to-end learning approaches [34, 183, 218, 274, 276, 293, 297, 332, 354]. Related to our work are approaches that propose unsupervised learning of depth and camera motion. This is typically done through learning two CNNs, a pose network and a depth network, that are trained to minimize a consistency loss between video frames. While CNN pose estimators have shown a lot of success on outdoor scenes, they have been challenged by cases with larger and more erratic camera motions (*e.g.*video from a hand-held device) [21]. Similar to those approaches, we train an end-to-end system using photometric and geometric consistency losses. Unlike that work, we are interested in pointcloud registration with larger camera motions and learning features for correspondence alignment of RGB-D scans.

View Synthesis. View synthesis is the task of generating views of the scene from image inputs. One line of work focuses on synthesizing views with small camera motions [131, 202, 213, 255, 266, 267]. NeRF and its variants [189, 194, 342] learn a rendering function for a specific scene from a large collection of multiview images. While the goal of that work is highly photorealistic rendering, we are primarily interested in utilizing view synthesis as a training task to enforce photometric consistency. Similar to our goals are approaches that synthesize views for unsupervised 3D learning of object shape [76, 120, 132, 288] and depth [34, 183, 293, 297, 332, 354]. Closest to our work is [310], who train a model for depth estimation and view synthesis with the goal of generating highly photorealistic views of the scene. Our work is complementary to the work of Wiles et al. [310] as we use depth to learn the pose, while they use the pose to learn depth.

2.3 Unsupervised Registration via Differentiable Rendering

Our goal is to build a system that can learn point cloud registration from RGB-D video without any explicit supervision. Our approach, shown in Fig. 2.2, is based on the traditional registration pipeline as it similarly extracts feature descriptors, finds correspondences, and finds the best alignment. We adapt this pipeline by operating directly on the images and learning our own features, as well as using photometric and geometric consistency losses to learn those features. We first present a high-level sketch of our approach before explaining each stage in more detail.

Approach Sketch. Given two RGB-D views of a scene and the camera's intrinsic matrix, we first extract 2D features for each view and lift them into two feature point clouds. We extract correspondences between the two point clouds and rank the correspondences based on their uniqueness. We then use a differentiable optimizer to align the top k correspondences and estimate the 6-DOF transformation between them. Finally, we render the point cloud from the two estimated viewpoints. We use photometric and geometric consistency losses between the RGB-D inputs and outputs and back-propagate through our entire pipeline.

2.3.1 Point Cloud Generation

Given an input RGB-D image, $I \in \mathbb{R}^{4\times H\times W}$, we would like to generate a point cloud $\mathcal{P} \in \mathbb{R}^{(6+F)\times N}$. Each point $p \in \mathcal{P}$ is represented by a 3D coordinate $\mathbf{x}_p \in \mathbb{R}^3$, a color $\mathbf{c}_p \in \mathbb{R}^3$, and a feature vector $\mathbf{f}_p \in \mathbb{R}^F$. We first use a feature encoder to extract a feature map using each image's RGB channels. The extracted feature map has the same spatial resolution as the input image. As a result, one can easily convert the extracted features and input RGB into a point cloud using the input depth and known camera intrinsic matrix. However, given that current depth sensors do not output the depth for every pixel, we omit the pixels with missing depth from our generated point cloud. To avoid heterogeneous batches, we mark points with missing depths so that subsequent operations ignore them.

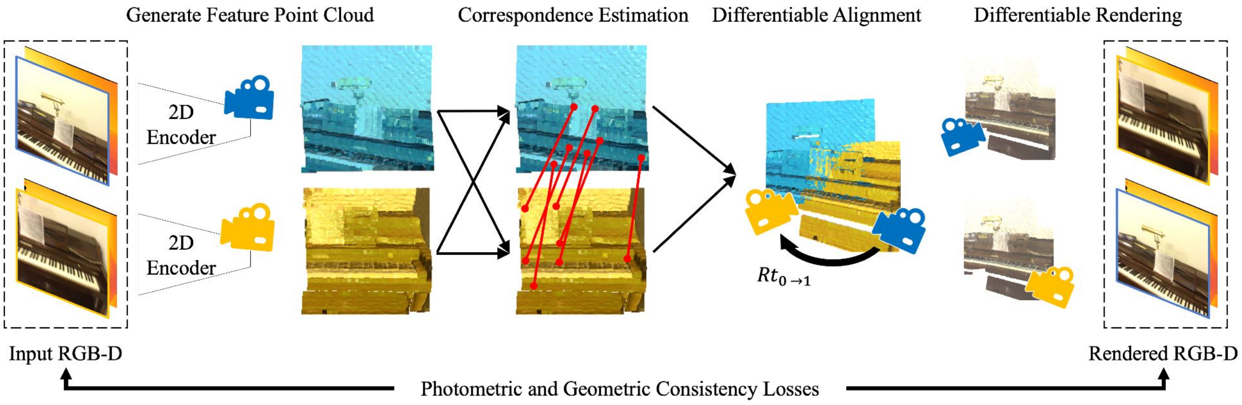


Figure 2.2: **UnsupervisedR&R.** Given two RGB-D images of a scene, we first encode the images into a feature map and project them into a 3D point cloud. We then extract correspondences between the two feature point clouds and use them to estimate $Rt_{0\to1}$; a 6-DOF transformation that aligns the two point clouds. Finally, we differentiably render the points from both point clouds and apply consistency losses.

2.3.2 Correspondence Estimation

Given two feature point clouds[1], $\mathcal{P}$, $\mathcal{Q} \in \mathbb{R}^{(6+F)\times N}$, we would like to find the correspondences between the point clouds. Specifically, for each point in $p \in \mathcal{P}$, we would like to find the point q_p such that

$$q_p = \underset{q\in\mathcal{Q}}{\arg\min}\, D(\mathbf{f}_p, \mathbf{f}_q), \tag{2.1}$$

where $D(p, q)$ is a distance metric defined on the feature space. In our experiments, we use cosine distance to determine the closest features.

We extract such correspondences for all points in both $\mathcal{P}$ and $\mathcal{Q}$ since correspondence is not guaranteed to be bijective. As a result, we have two sets of correspondences, $\mathcal{C}_{\mathcal{P}\to\mathcal{Q}}$ and $\mathcal{C}_{\mathcal{Q}\to\mathcal{P}}$, where each set consists of N pairs.

Ratio Test. Determining the quality of each correspondence is a challenge faced by any correspondence-based geometric fitting approach. Extracting correspondences based on only the nearest neighbor will result in many false positives due to falsely matching repetitive pairs or non-mutually visible portions of the image.

The standard approach is to estimate a weight for each correspondence that captures the quality of this correspondence. Recent approaches estimate a correspondence

[1] As noted in Sec. 2.3.1, point clouds will have different numbers of valid points based on the input depth. While our method deals with this by tracking those points and omitting them from subsequent operations, we assume all the points are valid in our model description to enhance clarity.

weight for each match using self-attention graph networks [241], PointNets [89, 330], and CNNs [51]. In our experiments, we found that a much simpler approach based on Lowe's ratio test [179] works well without requiring any additional parameters in the network. The basic intuition behind the ratio test is that unique correspondences are more likely to be true matches. As a result, the quality of correspondence (p, q_p) is not simply determined by $D(p, q_p)$, but rather between the ratio r which is defined as

$$r = \frac{D(p, q_{p,1})}{D(p, q_{p,2})}, \tag{2.2}$$

where $q_{p,i}$ is the i-th nearest neighbor to point p in $\mathcal{Q}$. Since $0 \leq r_p \leq 1$ and a lower ratio indicates a better match, we weigh each correspondence by $w = 1 - r$.

In the traditional formulation, one would define a distance ratio threshold for inlier vs outliers. Instead, we rank the correspondences by their ratio weight and pick the top k correspondences. We pick an equal number of correspondences from $\mathcal{C}_{\mathcal{P}\rightarrow\mathcal{Q}}$ and $\mathcal{C}_{\mathcal{Q}\rightarrow\mathcal{P}}$. Additionally, we keep the weights for each correspondence to use in the geometric fitting step. Hence, we end up with a correspondence set $\mathcal{M} = \{(p, q, w)_i : 0 \leq i < k\}$ where k=400.

2.3.3 Geometric Fitting

Given a set of correspondences $\mathcal{M}$, we would like to find the transformation, $\mathcal{T}^* \in$ SE(3) that would minimize the error between the correspondences

$$\mathcal{T}^* = \underset{\mathcal{T} \in \text{ SE(3)}}{\arg\min}\, E(\mathcal{M}, \mathcal{T}) \tag{2.3}$$

where the error $E(\mathcal{M}, \mathcal{T})$ is defined as:

$$E(\mathcal{M}, \mathcal{T}) = |\mathcal{M}|^{-1} \sum_{(p,q,w) \in \mathcal{M}} w \left(\mathbf{x}_p - \mathcal{T}(\mathbf{x}_q)\right)^2 \tag{2.4}$$

This can be framed as a weighted Procrustes problem and solved using a weighted variant of Kabsch's algorithm [130].

While the original Procrustes problem minimizes the distance between a set of unweighted correspondences [93], Choy et al. [51] have shown that one can integrate weights into this optimization. This is done by calculating the covariance matrix between the centered and weighted point clouds, followed by calculating the SVD of the covariance matrix. For more details, see [51, 130].

Integrating weights into the optimization is important for two reasons. First, it allows us to build robust estimators that weigh correspondences based on our confidence in their uniqueness. More importantly, it makes the optimization differentiable with respect to the weights, allowing us to backpropagate the losses to the encoder for feature learning.

Randomized Optimization. While this approach is capable of integrating the weights into the optimization, it can still be sensitive to outliers with nonzero weights. We take inspiration from RANSAC and use random sampling to mitigate the problem of outliers. More specifically, we sample t subsets of $\mathcal{M}$, and use Eq. (2.3) to find t candidate transformations. We then choose the candidate that minimizes the weighted error on the full correspondence set. Since the t optimizations on the correspondence subsets are all independent, we are able to run them in parallel to make the optimization more efficient. We deviate from classic RANSAC pipelines in that we choose the transformation that minimizes the weighted error instead of maximizing the inlier count to avoid having to define an arbitrary inlier threshold.

It is worth noting that the model can be trained and tested with a different number of random subsets. In our experiments, we train the model with 10 randomly sampled subsets of 80 correspondences each. At test time, we use 100 subsets with 20 correspondences each. We evaluate the impact of those choices on performance and run time in Sec. 2.4.2.

2.3.4 Point Cloud Rendering

The final step is to render the RGB-D images of the aligned point clouds. This provides us with our primary learning signals: photometric and depth consistency. The core idea is that if the camera locations are estimated correctly, the point cloud render will be consistent with the input images. We use differentiable rendering to project the colored point clouds onto an image using the estimated camera pose and known intrinsics. Our rendering pipeline is based on Wiles et al. [310].

A naive approach of simply rendering both point clouds suffers from a degenerate solution: the rendering will be accurate even if the alignment is incorrect. An extreme case of this would be to always estimate cameras looking in opposite directions. In that case, each image is projected in a different location of space, and the output will be consistent without alignment. We address this issue by forcing the network to render each view using only the other image's point cloud, as shown in Fig. 2.3. This forces the network to learn consistent alignment, as a correct reconstruction requires the mutually visible parts of the scene to be correctly aligned. This introduces another challenge: *how to handle the non-mutually visible surfaces of the scene?*

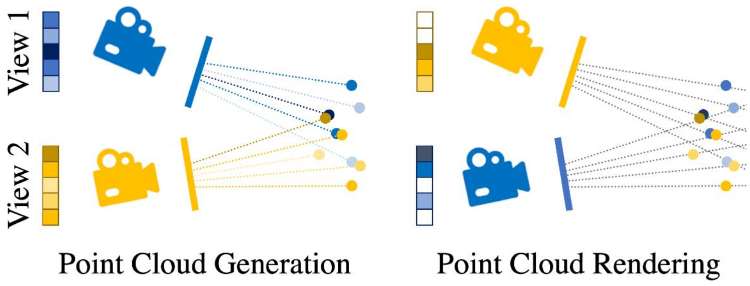


Figure 2.3: **Point Cloud Rendering.** We project the views from both views, but only render from the alternative view; *e.g.*we render the points projected from view 2 in the perspective of view 1. This can result in invalid pixels, visualized in white. *(For clarity, we show 1D projections in 2D space.)*

While view synthesis approaches hallucinate the missing regions to output photo-realistic imagery [310], earlier work in differentiable SfM observed that the gradients coming from the hallucinated region negatively impact the learning [354]. Our solution to this problem is to evaluate the loss for valid pixels only. Valid pixels, as shown in Fig. 2.3, are ones for which rendering was possible; *i.e.*, there were points along the viewing ray for those pixels. This is important in this work since invalid pixels can occur due to two reasons: non-mutually visible surfaces and pixels with missing depth. While the first reason is due to our approach, the second reason for invalid pixels is governed by the current depth sensors, which do not produce a depth value for each pixel.

In our experiments, we found that pose networks are very susceptible to the issues above; the network starts estimating very large poses within the first hundred iterations and never recovers. We also experimented with rendering the features and decoding them, similar to [310], but found that this resulted in worse alignment performance.

2.3.5 Losses

We use three consistency losses to train our model: photometric, depth, and correspondence. The photometric and depth losses are the L1 losses applied between the rendered and input RGB-D frames. Those losses are masked to only apply to valid pixels, as discussed in Sec. 2.3.4. Additionally, we use the correspondence error calculated in Eq. (2.4) as our correspondence loss. We weight the photometric and depth losses with a weighting of 1, while the correspondence loss receives a weighting of 0.1.

2.4 Experiments

We now empirically evaluate our model on pairwise point cloud registration. Our experiments aim to answer several questions:

1. Does unsupervised training provide us with useful features for alignment?
2. Does training with RGB-D frames alleviate the need for pose supervision?
3. How do the different components of the model contribute to its performance?

We address those questions by evaluating our approach on two datasets of indoor scenes: ScanNet [56] and 3DMatch [337]. We find that our approach achieves better registration accuracy than off-the-shelf visual and geometric feature descriptors (Sec. 2.4.1). We also find that our approach performs on par with supervised geometric registration approaches despite using significantly simpler correspondence matching and alignment algorithms; supporting our claim that RGB-D video can alleviate the need for pose supervision. Finally, we analyze our model components through several key ablations (Sec. 2.4.2).

Datasets. We evaluate our approach using ScanNet [56] and 3D Match [337]. ScanNet contains RGB-D images and ground-truth camera poses for 1513 scenes, while 3D Match is a much smaller dataset with a total of 101 scenes. We use the official data split of 1045/156/312 scenes for train/val/test for ScanNet. 3D Match only provides a train/test split, so we further divide the train split into train and validation; resulting in 71/11/19 RGB-D sequences for train/val/test split. We generate view pairs by sampling image pairs that are 20 frames apart. We sample the training scenes more densely by sampling all pairs that are 20 frames apart. This results in 1594k/12.6k/26k ScanNet pairs and 122k/1.5k/1.5k 3D Match pairs.

Baselines. We compare our model to several learned and non-learned point cloud registration approaches. Since we are interested in the unsupervised setting, we first compare with methods that do not require pose supervision. Our first set of baselines uses off-the-shelf keypoint detectors and descriptors with RANSAC [82] as the robust estimator. For all these baselines, we use Open3D's RANSAC implementation [353]. Despite being proposed over a decade ago, SIFT features are still used and serve as a strong baseline for a non-learned method. SuperPoint [66] is a recently proposed approach for keypoint detection and description and has achieved state-of-the-art performance in correspondence matching on several benchmarks. Finally, FCGF [50] is a recently proposed geometric feature descriptor that has also achieved state-of-the-art performance on several 3D correspondence benchmarks. Furthermore, FCGF features have been used by several recent approaches for point cloud registration without further fine-tuning [51, 89].

We also compare with two supervised geometric registration approaches: DGR [51] and 3D MV Registration [89]. Both of these approaches operate on FCGF point cloud embeddings as their input and learn how to extract good correspondences between pairs. There are two salient differences between our approaches: First, our approach is unsupervised, while those approaches rely on pose supervision. Second, our approach operates on RGB-D, while those approaches use the FCGF embedding of the point cloud without relying on the images. This comparison demonstrates how leveraging the currently ignored RGB modality could alleviate the need for pose supervision and pre-trained descriptors. We emphasize that we use the weights provided by the authors, which were trained on the 3D Match Geometric Registration benchmark.

Feature Encoder. The feature encoder is the only trainable component in our pipeline. We use a small ResNet model consisting of 6 layers. The first layer is a 2D convolution layer with a kernel size of 3 and an output channel dimension of 64. This is followed by two ResNet basic blocks that retain the spatial and feature dimensions of the activations. Finally, we use a 2D convolution layer to map the feature dimension from 64 to 32. We reduce the feature dimension to 32 as it allows us to use the fast kNN CUDA kernel defined in PyTorch3D [224]. All convolution layers are followed by BatchNorm and ReLU activation, except for the last layer.

Training Details. We train our model with the Adam [137] optimizer with a learning rate of 10^{-4} and momentum parameters of (0.9, 0.99). We train each model for 200K iterations. We implement our approach in PyTorch [224], while making extensive use of PyTorch3D [224] and Open3D [353].

2.4.1 Pairwise Registration

We first evaluate our approach on point cloud registration. Given two RGB-D images, we estimate the 6-DOF pose that would best align the first input image with the second. The transformation is represented by a rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$.

Evaluation Metrics. We evaluate pairwise registration by evaluating the pose prediction as well as the chamfer distance between the estimated and ground-truth alignments. We compute the angular and translation errors as follows:

$$E_{\text{rotation}} = \arccos(\frac{Tr(\mathbf{R}_{pr}\mathbf{R}_{gt}^{\top}) - 1}{2}), \tag{2.5}$$

$$E_{\text{translation}} = ||\mathbf{t}_{pr} - \mathbf{t}_{gt}||_2. \tag{2.6}$$

We report the translation error in centimeters and the rotation errors in degrees.

While pose gives us a good measure of performance, some scenes are inherently ambiguous and multiple alignments can explain the scene appearance; *e.g.*, walls, floors, and symmetric objects. To address these cases, we compute the chamfer distance between the scene and our reconstruction. Given two point clouds where $\mathcal{P}$ represents the correct alignment of the scene and $\mathcal{Q}$ represents our reconstruction of the scene, we can define the closest pairs between the point clouds as set $\Lambda_{\mathcal{P},\mathcal{Q}} = \{(p, \arg\min_{q\in\mathcal{Q}} ||p - q||) : p \in \mathcal{P})$. We then compute the chamfer error as follows:

$$E_{\text{cham}} = |\mathcal{P}|^{-1} \sum_{(p,q)\in\Lambda_{\mathcal{P},\mathcal{Q}}} ||\mathbf{x}_p - \mathbf{x}_q|| + |\mathcal{Q}|^{-1} \sum_{(q,p)\in\Lambda_{\mathcal{Q},\mathcal{P}}} ||\mathbf{x}_q - \mathbf{x}_p||. \tag{2.7}$$

For each of these error metrics, we report the mean and median errors over the dataset as well as the accuracy for different thresholds.

We conduct our experiments on ScanNet and report the results in Tab. 2.1. We find that our model learns accurate point cloud registration, outperforming prior feature descriptors and performing on par with supervised geometric registration approaches. We next analyze our results through the questions posed at the start of this section.

Does unsupervised learning improve over off-the-shelf descriptors? Yes. We evaluate our approach against the traditional pipeline for registration: feature extraction using an off-the-shelf keypoint descriptor and alignment via RANSAC. We show large performance gains over both traditional and learned descriptors. It is important to note that FCGF and SuperPoint currently represent the state-of-the-art for feature descriptors. Furthermore, both methods have been used directly, without further fine-tuning, to achieve the highest performance on image registration benchmarks [241] and geometric registration benchmarks [51, 89]. We also find that our approach learns features that can generalize to similar datasets. As shown in Tab. 2.1, our model trained on 3D Match outperforms the off-the-shelf descriptors while being competitive with supervised geometric registration approaches.

Table 2.1: **Pairwise Registration on ScanNet.** We outperform existing registration pipelines that use traditional or learned feature descriptors with RANSAC. Furthermore, we perform on-par with supervised geometric matching methods that were trained on 3D Match, demonstrating the utility of unsupervised training in this domain.

			Rotation					Translation					Chamfer	
	Features		Accuracy ↑			Error ↓		Accuracy ↑			Error ↓		Error ↓	
	Vis	3D	5°	10°	45°	Avg	Med	5	10	25	Avg	Med	Avg	Med
RANSAC + Feature Descriptors														
SIFT	✓	✗	55.5	75.8	89.2	18.6	4.4	17.8	45.0	80.0	26.2	11.2	40.8	1.7
SuperPoint [66]	✓	✗	65.7	86.7	96.7	8.8	3.5	21.0	51.7	88.0	16.2	9.7	19.3	1.2
FCGF [50]	✗	✓	69.5	87.6	96.1	9.4	3.2	28.1	58.2	82.8	23.6	8.3	23.8	0.9
Supervised Geometric Approaches														
DGR [51]	✗	✓	81.1	89.3	94.8	9.4	1.8	54.5	76.2	88.7	18.4	4.5	13.7	0.4
3D MV Reg [89]	✗	✓	87.7	93.2	97.0	6.0	1.2	69.0	83.1	91.8	11.7	2.9	10.2	0.2
Ours (3D Match)	✓	✗	87.6	93.1	98.3	4.3	1.0	69.2	84.0	93.8	9.5	2.8	7.2	0.2
Ours (ScanNet)	✓	✗	92.7	95.8	98.5	3.4	0.8	77.2	89.6	96.1	7.3	2.3	5.9	0.1

Does RGB-D training alleviate the need for pose supervision? Yes. We compare our approach to two recently proposed supervised point cloud registration approaches: DGR [51] and 3D Multi-view Registration [89]. Since their model was trained on 3D Match, we also train our model on 3D Match and report the numbers. We find that our model is competitive with supervised approaches when trained on their dataset and can outperform them when trained on ScanNet. However, a direct comparison is more nuanced since those two classes of methods differ in two key ways: training supervision and input modality.

We argue that the recent rise in RGB-D cameras on both hand-held devices and robotic systems supports our setup. First, the rise in devices suggests a corresponding increase in RGB-D raw data that will not necessarily be annotated with pose information. This increase provides a great opportunity for unsupervised learning to leverage this data stream. Second, while there are cases where depth sensing might be the better or only option (*e.g.*, dark environment or highly reflective surfaces.), there are many cases where one has access to both RGB and depth information. The ability to leverage both can increase the effectiveness and robustness of a registration system. Finally, while we only learn visual features in this work, we note that our approach is easily extensible to learning both geometric and visual features since it is agnostic to how the features are calculated.

Table 2.2: **Ablation Results.** Our ablation experiments demonstrate the utility of the ratio test for correspondence filtering. Furthermore, we find that some ablations can improve model performance when used for training, but not for testing.

	Ablation		**Rotation**		**Translation**		**Chamfer**	
	Train	**Test**	**Mean**	**Med.**	**Mean**	**Med.**	**Mean**	**Med.**
Full Model			3.4	0.8	7.3	2.3	5.9	0.1
Model with Joint Rendering	✓		5.5	1.1	12.0	3.1	9.0	0.2
- Randomized Optimization	✓	✓	4.7	1.3	10.8	3.9	8.2	0.3
- Ratio Test	✓	✓	6.9	1.9	15.1	5.2	10.5	0.5
- Randomized Optimization	✓	✗	2.8	0.8	6.1	2.1	5.0	0.1
- Ratio Test	✓	✗	4.1	1.1	9.3	3.2	6.8	0.2
- Randomized Optimization	✗	✓	5.8	1.6	13.3	4.7	9.7	0.4
- Ratio Test	✗	✓	16.6	5.5	35.5	13.4	27.8	2.8

2.4.2 Ablations

We perform an ablation study to understand the model's performance and its various components. In particular, we are interested in better understanding the impact of the optimization and rendering parameters on the overall model performance. While some ablations can only be applied during training (*e.g.*, rendering choice), ablations that affect the correspondence estimation and fitting can be selectively applied during training, inference, or both. Hence, we consider all variants.

Joint Rendering. We first investigate the impact of our rendering choices. As we discuss in Sec. 2.3.4, we render alternate views to force the model to align the point clouds to produce accurate renders. As shown in Tab. 2.2, we find that naively rendering the joint point cloud results in a significant performance drop. This supports our claim that joint rendering negatively impacts feature learning as the model achieves good photometric consistency even if the point clouds are not accurately aligned.

Ratio Test. In our approach, we use Lowe's ratio test to estimate the weight for each correspondence. We ablate this component by instead using the feature distance between the corresponding points to rank the correspondences. Since this ablation can be applied to training or inference independently, we apply it to training, inference, or both. Our results indicate that the ratio test is critical to our model's performance, as ablating it results in the largest performance drop. This supports our initial claims about the utility of the ratio test as a powerful heuristic for filtering correspondences. It is worth noting that

Table 2.3: **Run-time Analysis.** We find that using a larger number of random subsets improves our performance while also increasing the inference time. This trade-off between performance and run-time could be used to tune the model based on the use case.

	Rotation		**Translation**		**Chamfer**		
Number of Subsets	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**	**Time (ms)**
5	4.8	1.2	10.5	3.4	7.8	0.2	50.4 ± 0.3
10	4.2	1.0	9.2	2.9	7.1	0.2	60.2 ± 0.3
20	3.8	0.9	8.4	2.6	6.7	0.2	79.2 ± 0.5
50	3.5	0.9	7.7	2.4	6.0	0.1	135.4 ± 1.1
100	3.4	0.8	7.3	2.3	5.9	0.1	239.6 ± 1.2
200	3.3	0.8	7.2	2.2	5.9	0.1	425.2 ± 6.1

Lowe's ratio test [179] shows incredible efficacy in determining correspondence weights, a function often undertaken by far more complex models in recent work [51, 89, 221, 241]. Our approach is able to perform well using such a simple filtering heuristic since it is also learning the features, not just matching them.

Randomized Subsets. In our model, we estimate t transformations based on t randomly sampled subsets. This is inspired by RANSAC [82] as it allows us to better handle outliers. We ablate this module by estimating a single transformation based on all correspondences. Similar to the ratio test, this ablation can be applied to training or inference independently. As shown in Tab. 2.2, ablating this component at test time results in a significant drop in performance. Interestingly, we find that applying it during training and relieving it during testing improves performance. We posit that this ablation acts similarly to DropOut [268] which forces the model to predict using a subset of features and is only applied during training. As a result, the model is forced to learn better features during training, while gaining the benefits of randomized optimization during inference.

Number of subsets. We find that the number of subsets significantly impacts run-time and performance. During training, we sample 10 subsets of 80 correspondences each. During testing, we sample 100 subsets of 80 correspondences each. For this experiment, we use the same pre-trained weights and only vary the number of subsets used. As shown in Tab. 2.3, a larger number of subsets improves the performance while also increasing the run-time, but the gains saturate at 100 subsets.

Table 2.4: **Impact of feature and alignment algorithm on Pairwise Registration.** Using Lowe's Ratio test to filter correspondences improves performance as expected, while using it to further weight the correspondences in the Weighted Procrustes algorithm further improves performance. Furthermore, our features outperform the baselines regardless of choice of alignment algorithm.

		Rotation		**Translation**		**Chamfer**	
Features	**Estimator**	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**
SIFT	RANSAC	25.4	5.4	33.8	12.5	54.3	2.5
SuperPoint	RANSAC	13.1	4.0	20.9	10.6	30.0	1.4
FCGF	RANSAC	10.9	4.9	20.3	11.8	15.9	2.2
Ours	RANSAC	10.6	4.6	19.9	11.0	9.9	1.6
SIFT	RANSAC-Corr	18.6	4.3	26.5	11.2	42.6	1.7
SuperPoint	RANSAC-Corr	8.9	3.6	16.1	9.7	19.2	1.2
FCGF	RANSAC-Corr	9.5	3.3	23.6	8.3	24.4	0.9
Ours	RANSAC-Corr	3.5	1.8	8.5	5.6	4.4	0.5
SIFT	Ours	14.5	2.0	26.5	5.7	20.8	0.5
SuperPoint	Ours	4.8	1.6	8.5	4.1	7.4	0.3
FCGF	Ours	15.3	4.3	34.8	11.6	28.9	2.0
Ours	Ours	3.4	0.8	7.3	2.3	5.9	0.1

Alignment Algorithm. During training, we use the Weighted Procrustes algorithm for registration to maintain the differentiability of the pipeline. However, during testing, we can use our pretrained features with any alignment method; *e.g.*, RANSAC. Furthermore, we could use the Weighted Procrustes algorithm with other feature descriptors to better understand its performance. we evaluate the pairwise registration performance for different pairs of feature descriptors and alignment algorithms.

Given that the choice of correspondence set can be critical to the performance of an alignment algorithm, we compare two variants of RANSAC. The first variant, RANSAC, considers a large correspondence set that includes all nearest neighbors in the feature space. The second variant, RANSAC-Corr, filters the correspondences using Lowe's ratio test and only keeps the top 400. This is similar to our approach, and the comparison allows us to understand the impact of Lowe's ratio test for filtering as opposed to both filtering and weighing the correspondence as done by the Weighted Procrustes algorithm. We use Open3D's RANSAC implementation [353] with a limit of 10^5 iterations.

We present the results in Tab. 2.4. We first observe that filtering the correspondences improves the performance of all feature descriptors, as shown by the improved performance of RANSAC-Corr and Weighted Procrustes. We note that some of those differences would decrease by allowing RANSAC to run for a longer number of iterations. However, even with 100,000 iterations, we find the inference with RANSAC can be up to 10x slower than using Weighted Procrustes. Intuitively, filtering correspondences should improve the performance since it makes it easier for RANSAC to find the best alignment by decreasing the number of outliers. This is especially true for methods that output a large number of feature descriptors, like FCGF and our method. Finally, we observe that using Weighted Procrustes further improves the performance of all visual features.

2.5 Qualitative Analysis

We visualize our correspondence registration and rendering in Figs. 2.4 and 2.5 to provide a clear picture of our model's performance and limitations. First, we show the additional registration results in Fig. 2.4. Our model can extract dense correspondences between images, allowing it to accurately estimate the camera motion in the scene and register the images. Similar to registration methods, our approach struggles when there is limited visual and geometric texture in the image. This results in the failure mode shown in the bottom row of Fig. 2.4. However, as shown in the second to last row, the dense correspondences can still help it identify the correct correspondences.

Second, we also show the RGB-D renderings produced by the model in Fig. 2.5. As can be seen, our model's reconstruction from the rendered features closely matches the appearance of the input images. Furthermore, as discussed in Sec. 2.3.4, our model only renders the overlapping sections of the scene, resulting in the sections of the image not being rendered. As presented in the image, our image pairs do not have significant overlap, demonstrating our model's ability to perform wide-baseline correspondence matching and point cloud registration.

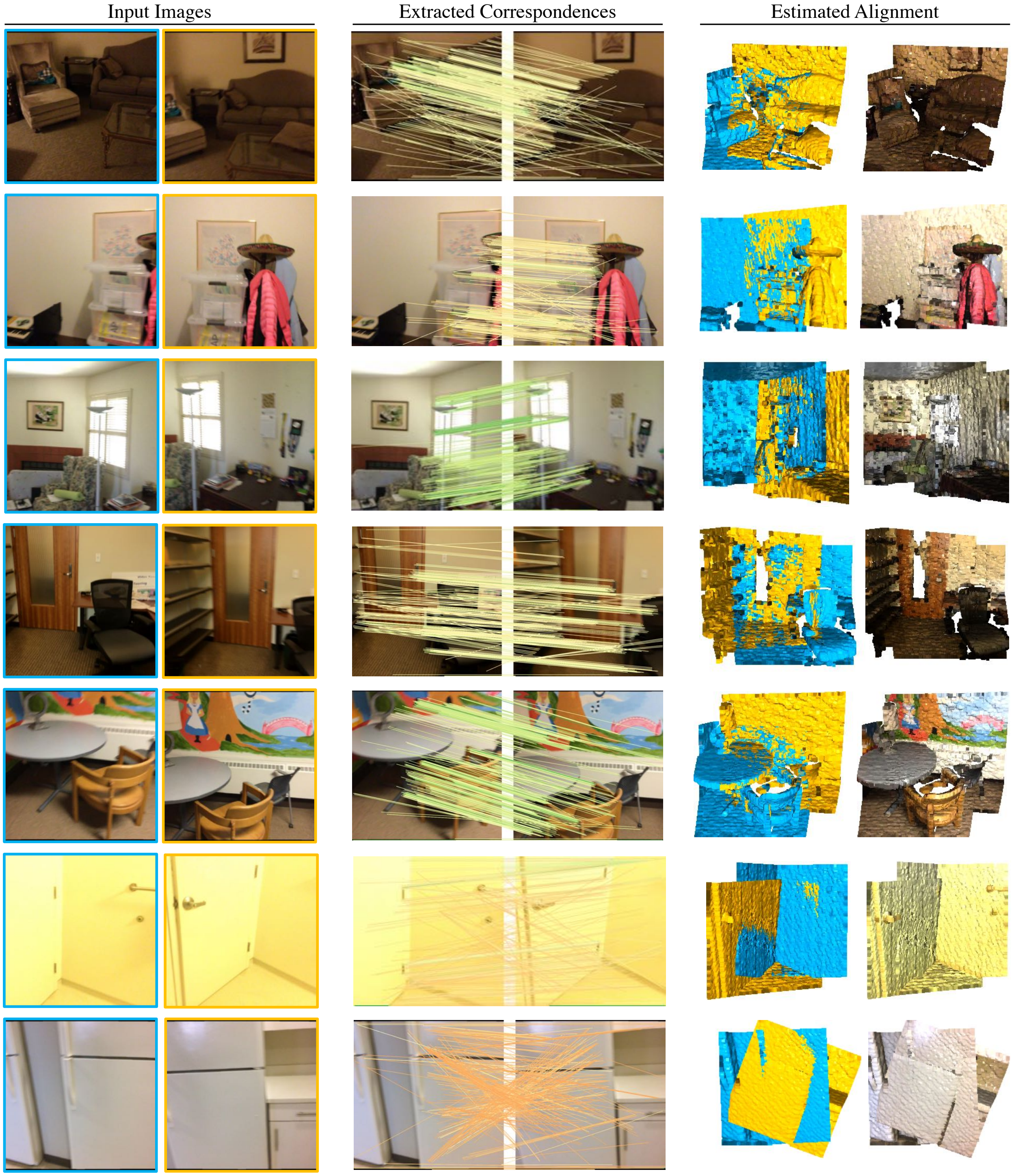


Figure 2.4: **Pairwise Registration Results.** Our model extracts can extract dense correspondences that enable accurate registration. However, we observe that the estimates can be very noisy for scenes with limited visual texture. Nevertheless, the correspondences estimated for such scenes have lower confidence, as indicated by the correspondence colors, which range from green for high confidence to red for low confidence.

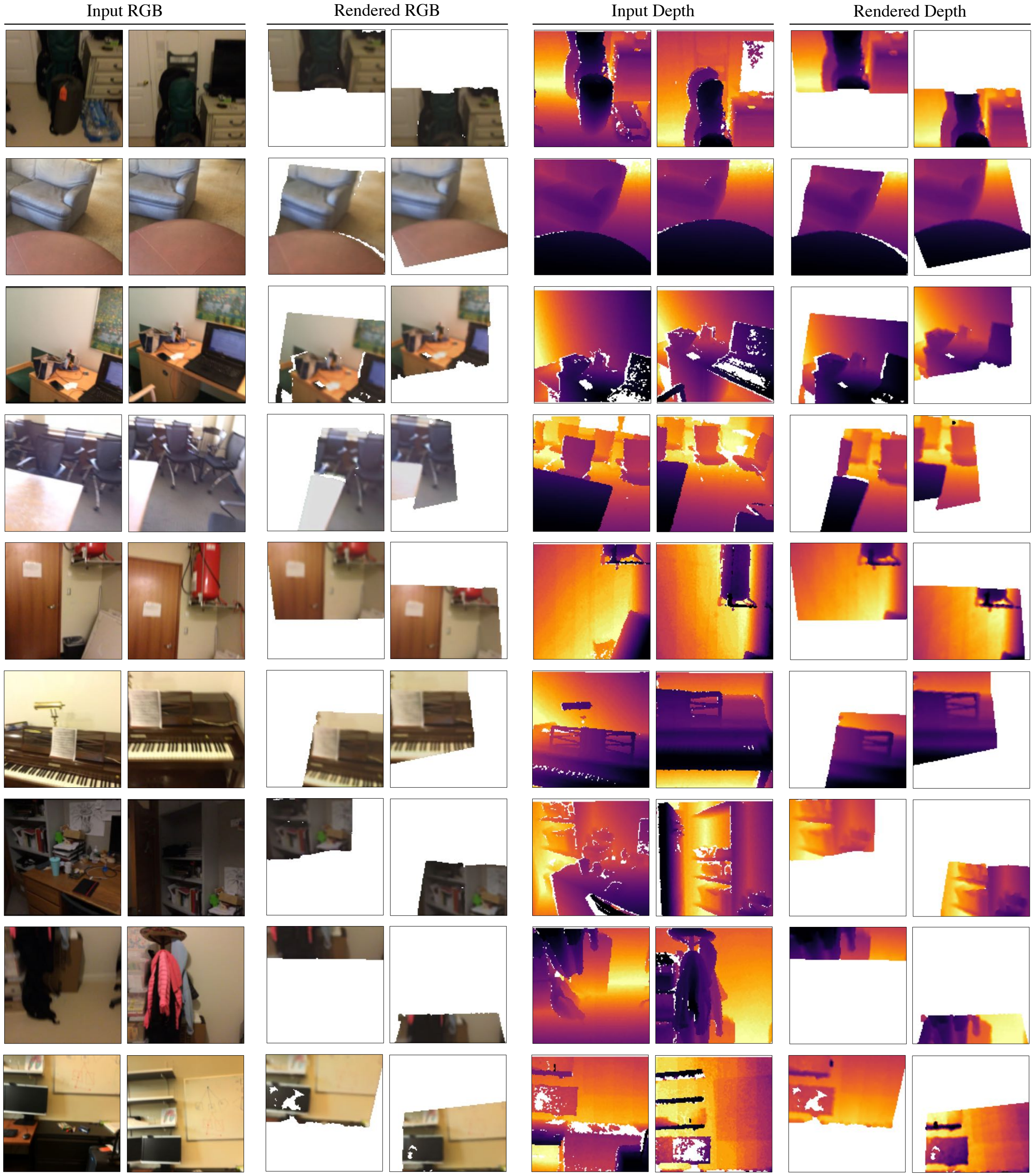


Figure 2.5: **RGB-D Rendering Results.** Our model accurately aligns and renders the scene. While missing depth pixels in the input are due to sensor issues, missing pixels in our renders can be due to non-mutually visible surfaces either due to a change in camera view, occlusion, or missing depth.

2.6 Discussion

In this chapter, we present an unsupervised, end-to-end approach to pairwise RGB-D point cloud registration. We observe that existing approaches to point cloud registration rely on pose supervision for learning geometric point cloud alignment. However, with the increase in cameras with depth sensors, we expect a large stream of raw RGB-D data. This provides us with the opportunity to develop approaches that learn directly from their sensor data without requiring any supervision.

Our key insight is that an accurate alignment entails geometrically consistency correspondence and the ability to accurately re-render the scene. Hence, we can train a model that learns good visual features by relying on those two signals, which it gets directly from its inputs and intermediate estimates. At the time of publication, our approach outperformed all existing traditional and learned feature descriptors, showcasing the utility and promise of relying on those simple consistency signals.

Furthermore, we observed that the use of principled components that are integrated into the learning can greatly simplify the approach. One clear example of this was the use of the ratio test which was proposed by Lowe [179] two decades ago. Despite its relative simplicity, we were still able to outperform approaches that trained neural networks to do the filtering of the correspondence.

Nevertheless, we observe several limitations in our work. First, our reliance on photometric consistency makes our approach sensitive to lighting changes that are inherent to many indoor environments. The rendering pipeline can also be slow and complicates our overall pipeline. Another limitation is that we assume that the transformation between the two views can be explained by a single rigid-body transform. This limits our ability to handle dynamics and is further exacerbated by photometric consistency which would struggle with any dynamics in the scene. However, those limitations are due to our modeling choices and are not inherent in the learning from correspondence. Furthermore, we alleviate some of those limitations in our next chapter by only relying on geometric consistency for learning.

Chapter 3

Learning from Geometric Consistency

In this previous chapter, we showed how models can learn good visual representations from RGB-D video by relying on photometric and geometric consistency. We now challenge this assumption and show that geometric consistency is actually sufficient for learning good representation learning. The intuition here is very simple: if our estimated correspondence is accurate, they should all agree on a single global consistency. If we assume that our estimates will contain a small set of consistent correspondences, then we can train models by penalizing any deviation from a robust estimate of the transformation. This observation allows us to both simplify our approach and extend to learning both visual and geometric representations.

In this chapter, we propose BYOC: a self-supervised approach that learns visual and geometric features from RGB-D video. Our approach relies on two key observations: First, randomly initialized convolutional neural networks are still effective feature extractors for correspondence estimation. Second, the noisy correspondences estimated by randomly initialized features can estimate coarsely accurate transformations when combined with robust estimators. BYOC builds on UnsupervisedR&R, presented in Chapter 2, by simplifying its setup, proposing a more robust geometric consistency loss, and incorporating geometric feature encoders. We are particularly interested in learning geometric features based on point cloud data, and we show how correspondences based on visual features provide good pseudo-labels for geometric feature learning. BYOC matches the performance of prior traditional and learned geometric feature descriptors and exhibits strong cross-dataset generalization. Those results are surprising when considering the training signal: how well the model can align its own *predicted* correspondences. This chapter is based on work previously published in [75].

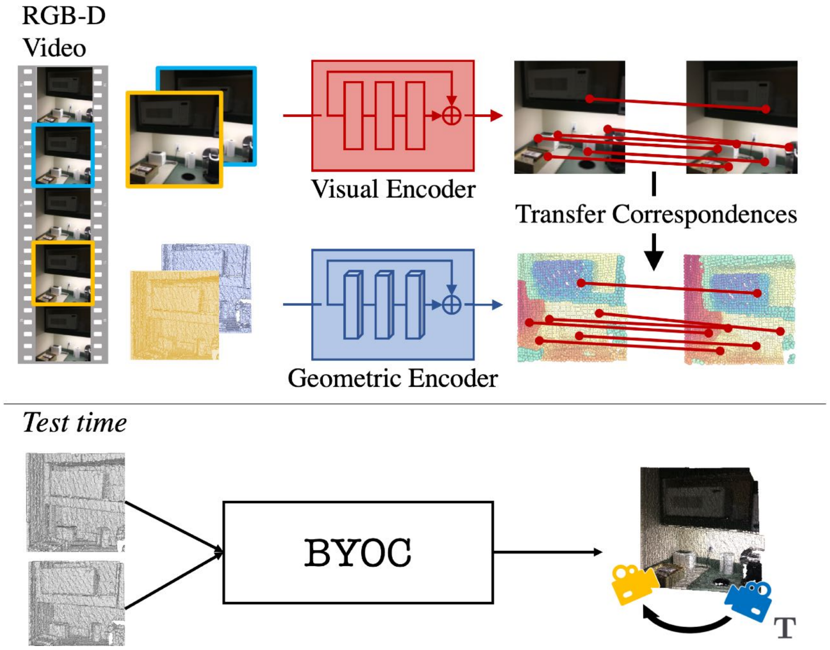


Figure 3.1: BYOC estimates visual correspondences and uses them to train a visual and a geometric encoder on RGB-D video frames. At test time, we only use the geometric encoder to register uncolored point clouds.

3.1 Introduction

One's ability to align two views of the same scene is closely intertwined with their ability to identify corresponding points between the two views. The duality between correspondence estimation and point cloud registration has long been recognized and serves as the basis for many approaches to both problems. Given an accurate registration of a scene, one can easily extract correspondences between the two views. Conversely, given point correspondences, one can easily register two views of a scene. *Can we leverage this cycle to jointly learn both correspondence estimation and point cloud registration from scratch?*

At the core of this cycle is the ability to generate good feature descriptors for points in the scene. The prevailing approach to 3D feature learning relies on preregistered scenes to sample ground-truth correspondences for the supervised training of a feature encoder. This is done by sampling positive and negative feature pairs and applying triplet [50, 135, 165, 327] or contrastive [13, 50, 319] losses. While very successful, these approaches require us to have already registered the raw depth or RGB-D scans to generate the training data. This limits this approach to data that can be successfully registered with automated approaches like COLMAP [243]. Ideally, we would leverage the success of supervised approaches without relying on ground-truth correspondence labels.

To this end, we propose Bootstrap Your Own Correspondences (BYOC): a self-supervised end-to-end approach that learns point cloud registration by leveraging pseudo-correspondence labels. Our approach extracts pseudo-correspondences using the features of a randomly initialized feature encoder. We use the sampled correspondences to register the point clouds and apply losses based on the quality of the registration to train the feature encoders. This allows us to slowly bootstrap[1] the feature learning process and learn from RGB-D scans without relying on any pose or correspondence supervision.

This approach works well for registering RGB-D frames, but it is less effective for raw point clouds. This is primarily due to the fact that randomly initialized 2D CNNs produce more distinctive features than current point cloud encoders, as shown in Fig. 3.3. We leverage this observation and propose bootstrapping the geometric feature learning using visual correspondences. We do this by using the estimated visual correspondences, as opposed to ground-truth correspondences [13, 50, 135, 165, 319, 327], to train the geometric encoder. We train the geometric encoder by adapting SimSiam [42], a non-contrastive self-supervised approach, for 3D representation learning. Unlike typical contrastive self-supervised approaches, SimSiam allows us to train the model using only positive pairs without requiring negative sampling or momentum encoders.

Our work draws inspiration from two sources: iterative closest point algorithm (ICP) [17, 45, 347] and self-supervised learning with pseudo-labels [31, 97, 154]. While seemingly different, the same intuition lies at the core of both lines of work. ICP is a registration algorithm that assumes that the closest points between two point clouds correspond to each other. Through iterative refinement and resampling, it can register roughly aligned point clouds. Meanwhile, self-supervised learning with pseudo-labels learns to predict pseudo-labels in the form of the current top prediction [154], feature clusters [31], or even a previous prediction [97]. Through redefining the labels over time, the model can progressively learn better representations. Both rely on the observation that pseudo-labels in a well-structured space (*i.e.*, similar entities already lie close to each other) can provide a valuable learning signal. This is particularly relevant for learning due to the observation that CNNs, *even* when randomly initialized, are good feature extractors [231, 291].

We evaluate our approach on two indoor scene datasets: ScanNet [56] and 3D Match [337]. Despite the simplicity of our approach, it outperforms hand-crafted features as well as several supervised baselines while being competitive with current state-of-the-art supervised approaches.

In summary, we propose a self-supervised approach that uses sampled correspondences from randomly initialized feature encoders to learn point-wise features for point cloud

[1] We use *bootstrap* in its idiomatic rather than its statistical sense.

registration. We further demonstrate how visual correspondences can further improve geometric feature learning. We demonstrate the efficacy of this approach on point cloud registration and correspondence estimation.

3.2 Related Work

3D Feature Descriptors. Early work on feature point extraction can be traced back to using corners for stereo matching [197]. The core intuition of extracting features based on histograms of gradients was later extended to 3D features [124, 125, 236, 239, 284]. More recently, the focus has shifted towards leveraging supervised learning for 3D feature learning [13, 50, 60–62, 88, 135, 165, 305, 327, 350]. The common approach is to sample positive and negative pairs between two views and then use them in triplet [50, 135, 165, 327] or contrastive [13, 50, 61, 319] losses. Other approaches propose applying unsupervised learning on reconstructed scenes [60, 319, 350]. While those approaches do not explicitly use ground-truth pose, they rely on reconstructed scenes that are generated using ground-truth pose. Unlike prior work, our approach learns directly from RGB-D scans without relying on ground-truth pose.

Point Cloud Registration. Early work on point cloud registration assumed perfect correspondence between the point clouds [8, 176]. This assumption was later relaxed by ICP by assuming that the closest point is the correspondence [17, 45, 347]. While this assumption holds for several applications (*e.g.*, registering scans from a high frame-rate scanner or fine-tuning alignment), it is challenged by large transformations and partially overlapping point clouds. Later work focused on designing feature descriptors for establishing correspondence and using robust estimators such as RANSAC to handle noise and outliers [285, 348]. For a review, see [215]. This has been extended further by incorporating learning into the registration process [26, 27, 51, 77, 89, 115, 221, 328]. Finally, recent work has proposed self-supervised approaches for registering objects [4, 108, 115, 304, 305, 328, 334] or reconstructed scenes [60, 135, 350]. Those approaches operate on dense point clouds that are constructed from aligned partial views. Hence, while the method might be self-supervised, the overall approach still requires ground-truth annotation. We are inspired by this line of work and extend it by learning directly from RGB-D scans instead of reconstructed scenes.

Self-supervised learning. Self-supervised learning refers to approaches that apply supervised learning to tasks where the data itself serves as the supervision. This idea has

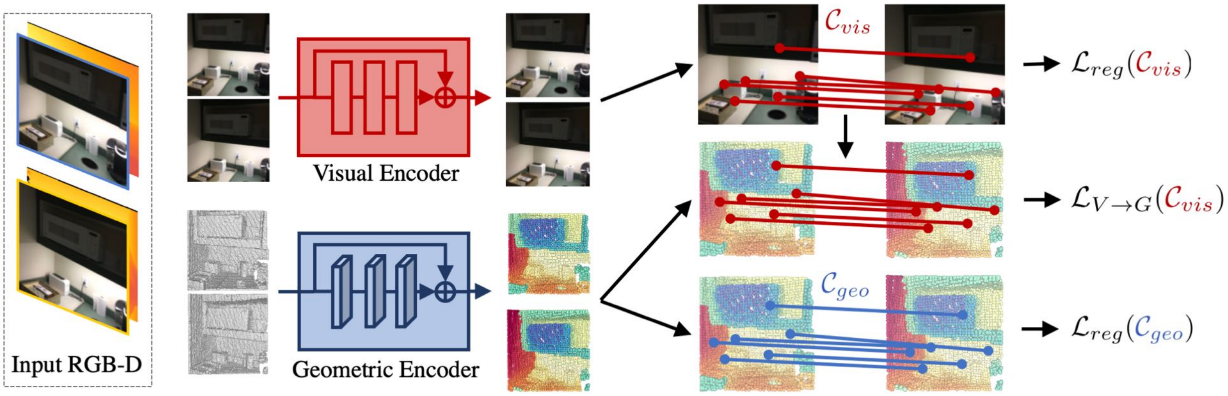

Input RGB-D
Visual Encoder
Geometric Encoder
$\mathcal{C}_{vis}$
$\mathcal{C}_{geo}$
$\mathcal{L}_{reg}(\mathcal{C}_{vis})$
$\mathcal{L}_{V\to G}(\mathcal{C}_{vis})$
$\mathcal{L}_{reg}(\mathcal{C}_{geo})$

Figure 3.2: **BYOC**. Our model takes as input two RGB-D images of a scene. First, we extract visual features from the images and geometric features from the point clouds. This results in two point clouds where each point has a 3D location, visual feature, and geometric feature. We then extract correspondences from the visual and geometric features. Those correspondences are used to estimate a transformation and compute a registration loss. We also apply a feature similarity loss on geometric features sampled using the visual correspondences.

been very popular for 2D representation learning with the goal of learning representations that generalize to downstream tasks [42, 64, 68, 87, 94, 97, 280]. Recently, PointContrast [319] and DepthContrast [346] demonstrated how to extend this formulation to 3D representation learning. We are inspired by this line of work but differ from it in several ways. First, our goal is to learn good features for registration, not for different downstream tasks. Second, we learn from RGB-D videos, not reconstructed scenes like [319]. Also, we learn point-level representations, not scene-level representations like [346]. Finally, while prior work has focused on using contrastive learning, we show that non-contrastive learning [42, 97] can be very effective for 3D feature learning despite being far simpler.

3.3 Bootstrap Your Own Correspondence

The goal of this work is to learn geometric point cloud registration from RGB-D video without relying on pose or correspondence supervision. Our approach, shown in Fig. 3.2, has three major components: visual registration, geometric registration, and correspondence transfer. The first two components are based on the traditional registration pipeline of feature extraction, correspondence estimation, and geometric fitting. The only difference between them is whether the features are extracted using a visual encoder from the image or a geometric encoder from the point cloud. The third component is based on

SimSiam [42] and applies a feature similarity loss on pairs of geometric features that are sampled using visual correspondences. Our key insight is that randomly initialized CNNs produce features that allow for coarse correspondence estimation and registration. This allows us to bootstrap the learning of both visual and geometric encoders by using estimated correspondences with registration and feature similarity losses.

3.3.1 Point Cloud Registration

Given two point clouds, $\mathcal{P}_0$ and $\mathcal{P}_1$, point cloud registration is the task of finding the transformation $\mathbf{T} \in \mathrm{SE}(3)$ that aligns them. Registration approaches commonly consist of three stages: feature extraction, correspondence estimation, and geometric fitting. In our approach, we register the point cloud pair using either visual or geometric features. Below we discuss each of these steps in detail.

Geometric Feature Extraction. The geometric encoder extracts features based on the geometry of the point cloud. We first generate a point cloud for each view using the input depth and known camera intrinsic matrix. We then encode each point cloud using a sparse 3D convolutional network [49, 96]. We use this network due to its success as a back-end for supervised registration approaches [50, 51, 89] and 3D representation learning [319, 346]. This network applies sparse convolution to a voxelized pointcloud, allowing it to extract features based on local geometry while maintaining a quick run-time. Similar to prior work [50, 319, 346], we find that a voxel size of 2.5 cm works well for indoor scenes. This step maps our input RGB-D image, $I_0, I_1 \in \mathbb{R}^{4\times H\times W}$ to $\mathcal{P}_0, \mathcal{P}_1 \in \mathbb{R}^{N\times(3+F)}$ where each point cloud has N points, and each point p is represented by a 3D coordinate $\mathbf{x}_p$ and a F-dimensional geometric feature vector $\mathbf{g}_p$.[2] We use a feature dimension of 32.

Visual Feature Extraction. The visual encoder extracts features based on the image. We use a ResNet encoder with two residual blocks as our image encoder and map each pixel to a feature vector of size 32. We use the projected 3D coordinates of the voxelized point cloud from the geometric encoder to index into the 2D feature map. This allows us to generate a point cloud for each input RGB-D image, where each point $p \in \mathcal{P}$ has a 3D coordinate $\mathbf{x}_p$, a visual feature $\mathbf{v}_p$, and a geometric feature $\mathbf{g}_p$. Since each point can be represented by a visual or a geometric feature, we can easily transfer the correspondences between the different feature modalities as shown in Sec. 3.3.2. We note that we only use the visual encoder during training to bootstrap the geometric feature learning. At test time, we register point clouds using only the geometric encoder.

[2] Voxelization will result in point clouds of varying dimension. We use heterogeneous batching to handle this in our implementation but assume that point clouds have the same size in our discussion for clarity.

Correspondence estimation. We estimate the correspondences between the two input views for each feature modality to output two sets of correspondences: $\mathcal{C}_{vis}$ and $\mathcal{C}_{geo}$. We first generate a list of correspondences by finding the nearest neighbor to each point in the appropriate feature space. Since each point cloud has N points, we end up with 2N candidate correspondences for each modality.

The candidate correspondences will likely contain a lot of false positives due to poor matching, repetitive features, and occluded or non-overlapping portions of the image. The common approach is to filter the correspondences based on some criteria of uniqueness or correctness. Recent approaches propose learning networks that estimate a weight for each correspondence [51, 89, 221]. In this work, we leverage the method proposed by [77] of using a weight based on Lowe's ratio [179]. Given two point clouds, $\mathcal{P}_0$ and $\mathcal{P}_1$, we find the correspondences of point $p \in \mathcal{P}_0$ by finding the two nearest neighbors q_p and q_{p,nn_2} to p in $\mathcal{P}_1$ in feature space. We can calculate the Lowe's ratio weight as follows:

$$w_{p,q_p} = 1 - \frac{D(\mathbf{f}_p, \mathbf{f}_{q_p})}{D(\mathbf{f}_p, \mathbf{f}_{q_{p,nn_2}})} \tag{3.1}$$

where D is cosine distance, and $\mathbf{f}_p$ is either the visual or the geometric feature descriptor depending on the feature modality used. It is worth noting that this formulation is similar to the triplet loss often used in contrastive learning, where q_p is the positive sample and q_{p,nn_2} is the hardest negative sample. We use the resulting weights to rank the correspondences and only include the top k correspondences. We use $k = 400$ in our experiments. Each element of our correspondence set $\mathcal{C}$ consists of the two corresponding points and their weight $(p, q, w_{p,q})$.

Geometric Fitting. For each set of correspondences, we estimate the transformation, $\mathbf{T}^* \in$ SE(3) that would minimize the mean-squared error between the aligned correspondences:

$$E(\mathcal{C}, \mathbf{T}) = \sum_{(p,q_p,w) \in \mathcal{C}} \frac{w}{\sum_{\mathcal{C}} w} ||\mathbf{x}_{q_p} - \mathbf{T}(\mathbf{x}_p)|| \tag{3.2}$$

This problem can be reformulated as a weighted Procrustes problem [93, 130, 263, 292] allowing for weights to be integrated into the operation to improve the optimization process while maintaining differentiability with respect to the weights [51]. We adopt this formulation due to its relative simplicity and ease of incorporation within an end-to-end trainable system.

Despite having filtered the correspondences, the correspondence set might still include some outliers that would result in an incorrect geometric fitting. We adopt the randomized

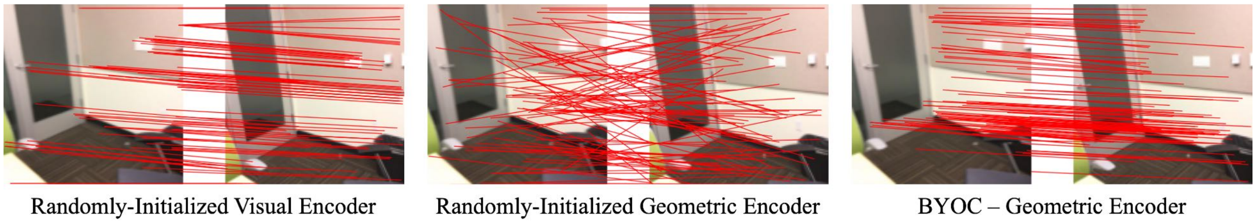

Randomly-Initialized Visual Encoder
Randomly-Initialized Geometric Encoder
BYOC – Geometric Encoder

Figure 3.3: **Randomly-initialized CNNs are good feature extractors.** Random visual features provide much better correspondence than random geometric features. We use this to bootstrap the learning of geometric features using estimated visual correspondences.

optimization used in [77] and similarly find that we get the best performance by only using it at test time.

Registration Loss. Our registration loss is defined with respect to our correspondence set and the estimated transformation as follows:

$$\mathcal{L}_{reg}(\mathcal{C}) = \underset{\mathbf{T} \in \text{SE}(3)}{\arg\min}\, E(\mathcal{C}, \mathbf{T}) \tag{3.3}$$

There are a few interesting things about this loss. First, the gradients are back-propagated to the feature encoder through both the weights, w, *and* the transformation, $\mathbf{T}$. Hence, the loss can be formulated without using the weights. We find that using the weight improved the performance of visual registration while deteriorating the performance of geometric registration. Therefore, in our model, we only apply the weighting to the visual registration branch while removing it from the geometric branch.

Second, the loss operates as a weighted sum over the residuals. Specifically, the loss is minimized if the correspondence with the lowest residual error has the highest weight. Since the weights are L1 normalized, the relative weighing of the correspondences matters. Removing the normalization results in an obvious degeneracy since the loss can be minimized by driving the weights to 0, which can be achieved by mode collapse. Finally, the weighted loss closely resembles a triplet loss since we estimate both a positive (first nearest neighbor) and a hardest negative (second nearest neighbor) sample. However, unlike the commonly used margin triplet loss, this formulation does not require defining a margin as it operates on the ratio of distances rather than their absolute value.

3.3.2 Visual To Geometric

The approach outlined in Sec. 3.3.1 works well with visual features, but it is less effective with geometric features. The reason for this becomes apparent once we consider the registration performance using features from randomly initialized encoders. As shown in Fig. 3.3, we observe that the features extracted from a randomly initialized visual encoder provide some distinctive output, while a random geometric encoder's outputs are more random. This has a strong impact on registration as shown in Tab. 3.2. Additionally, as we find in Sec. 3.4.1, our approach using visual features performs much better than our approach with geometric features.

Ideally, we would leverage good visual correspondence to improve geometric feature learning. We observe that geometric feature learning approaches typically rely on ground-truth correspondence sampled from the 3D reconstructed scene. [13, 50, 88, 165, 327]. We adapt this approach to the unsupervised setting by sampling feature pairs using visual correspondences. This is simple in our approach since each point has both a visual feature and a geometric feature, so transferring correspondences is simply indexing into another tensor. Since the correspondences act as indices, the loss is only back-propagated to the geometric encoder.

Current 3D feature learning approaches rely on both positive and negative pairs to define triplet [50, 135, 165, 327] or contrastive [13, 50, 319] losses. However, as noted in the literature, those losses can be difficult to apply due to their susceptibility to mode collapse and sensitivity to hyperparameter choices and negative sampling strategy [50, 319, 346]. Those issues are amplified in our setting since the visual correspondences only provide us with estimated, not ground-truth, positive samples. Instead of the typical contrastive setup, we adapt the recently proposed non-contrastive self-supervised learning approaches [42, 97] to the point cloud setting. We use SimSiam [42] due to its simplicity and strong performance: it does not require negative sampling or a momentum encoder.

We adapt SimSiam by applying it to the geometric features of visually corresponding points instead of different augmentations of the same image. Given a correspondence $(p, q) \in \mathcal{C}_{vis}$, we first project the features using a two-layer MLP projection head and apply a stop-gradient operator on the features:

$$\mathbf{z}_p = project(\mathbf{g}_p). \tag{3.4}$$

$$\overline{\mathbf{g}_p} = \texttt{stopgradient}(\mathbf{g}_p). \tag{3.5}$$

We then compute the loss based on the cosine distance between each geometric feature and the projection of its correspondence:

$$\mathcal{L}_{V\to G}(\mathcal{C}_{vis}) = \frac{1}{|\mathcal{C}_{vis}|}\sum_{(p,q)\in\mathcal{C}_{vis}} D(\overline{\mathbf{g}_p}, \mathbf{z}_q) + D(\mathbf{z}_p, \overline{\mathbf{g}_q}) \tag{3.6}$$

where D is the cosine distance function and $\mathcal{C}_{vis}$ is the set of visual correspondences.

3.4 Experiments

We evaluate our approach on point cloud registration of indoor scenes. We train our model on ScanNet, a large dataset of indoor scenes, and evaluate it on ScanNet and the 3D Match registration benchmark. Our experiments aim to answer two questions: (1) can we learn accurate point cloud registration from bootstrapped correspondences?; (2) can we leverage RGB-D video at training time to train better geometric encoders?

BYOC variants. We consider two variants of our model: BYOC-Geo and BYOC. BYOC-Geo is trained only on depth pairs using the geometric registration loss. This variant applies the bootstrapping idea without leveraging the visual correspondence. BYOC, shown in Fig. 3.2, is trained using RGB-D pairs but only uses the geometric encoder for registration at test time. Since BYOC uses visual correspondences to train the geometric features, we use data augmentation to further improve the geometric feature learning. We sample random rotations and apply them to the point cloud before the geometric encoder. This is a common augmentation in 3D feature learning [50, 319] and is intended to improve the learned feature's rotational equivariance. We note that training BYOC-Geo with rotation augmentation greatly deteriorates its performance.

RANSAC Baselines. We use the Open3D [353] RANSAC implementation and use the same parameters for all experiments, including our features. We run RANSAC for 10^5 iterations with estimates being scored on the number of inliers (within a threshold of 3 cm). It is worth emphasizing that this work is focused on learning better geometric features. As a result, our comparison here is against other features (*e.g.*, FPFH and FCGF), not against RANSAC. A more accurate or robust estimator would improve the performance of all methods.

Datasets. We evaluate our approach on two datasets of indoor scenes: ScanNet [56] and 3D Match [337]. While both datasets provide RGB-D video annotated with ground-truth camera poses, 3D Match provides an additional geometric registration benchmark that is more challenging due to the larger viewpoint changes. ScanNet provides pose annotated RGB-D video for 1513 scenes, while 3D Match only has 101 scenes. We emphasize that we only use RGB-D video and camera intrinsics for training our model. We use the official scene splits for both datasets and generate view pairs by sampling image pairs that are 20 frames apart. This results in 1594k/12.6k/26k RGB-D pairs for ScanNet and 122k/1.5k/1.5k RGB-D pairs for 3D Match.

Training Details. We use the Adam [137] optimizer with a learning rate of 10^{-4} and momentum parameters of (0.9, 0.99). We train each model for 200K iterations with a batch size of 8. We implement our models in PyTorch [224], with extensive use of PyTorch3D [224], Open3D [353], and Minkowski Engine [49]. The code is available at https://github.com/mbanani/byoc.

3.4.1 Point Cloud Registration

We first evaluate our approach on point cloud registration on ScanNet and report our results in Tab. 3.1. Given two point clouds, we estimate the transformation $\mathbf{T} \in \text{SE}(3)$ that would align the point clouds. We emphasize that we discard the visual encoder at the test time and only use the geometric encoder on the point cloud input.

Baselines. We compare our approach to both classical hand-crafted and supervised learning approaches. We first compare our approach against two variants of ICP [235]. ICP is an important baseline since it is both an inspiration of this work and a classical point cloud registration algorithm. We also compare against a RANSAC-based aligner using FPFH [236] or FCGF [50] 3D feature descriptors. FPFH [236] is a hand-crafted 3D feature descriptor that represents a point by a histogram of the spatial relationships to its nearest neighbors. FPFH is one of the best non-learned 3D feature descriptors and is representative of the performance of hand-crafted 3D features. FCGF [50] is a recently proposed learned 3D feature descriptor that combines sparse 3D convolutional networks with contrastive losses trained with ground-truth correspondences.

We also compare against Deep Global Registration [51] and 3D Multiview Registration[3] [88]: two supervised approaches that learn to estimate correspondences on top of FCGF features. Those approaches use supervision for both feature learning and correspondence estimation, while our approach is unsupervised for both.

[3] It is worth noting that 3D Multi-view Registration [89] proposes both a method for pairwise registration and synchronizing multiple views at the same time. We only compare against their pairwise registration module.

Table 3.1: **Pairwise Registration on ScanNet.** We outperform existing registration pipelines that use traditional or learned geometric feature descriptors with a RANSAC or Weighted Procrustes estimator. Furthermore, we perform on-par with supervised approaches that were trained on 3D Match, demonstrating the utility of unsupervised training in this domain.

		Rotation					Translation					Chamfer	
		Recall ↑			Error ↓		Recall ↑			Error ↓		Error ↓	
	Train Set	5°	10°	45°	Mean	Med.	5	10	25	Mean	Med.	Mean	Med.
ICP (Point-to-Point)	-	31.7	55.6	99.6	10.4	8.8	7.5	19.4	74.6	22.4	20.0	32.9	14.1
ICP (Point-to-Plane)	-	54.4	68.0	98.6	8.6	3.6	30.0	36.7	70.4	23.6	18.0	29.5	8.2
FPFH + W. Procrustes	-	22.2	48.2	84.9	27.8	10.4	7.4	19.6	56.3	54.1	25.3	26.5	5.8
FPFH + RANSAC	-	34.1	64.0	90.3	20.6	7.2	8.8	26.7	66.8	42.6	18.6	23.3	2.9
Pose-Supervised Approaches													
FCGF + W. Procrustes	3DMatch	54.1	73.3	92.2	15.3	4.3	30.8	46.2	73.0	35.0	11.6	21.5	1.4
FCGF + RANSAC	3DMatch	75.3	87.7	95.6	9.7	2.5	39.7	64.9	86.5	20.8	6.4	13.0	0.6
DGR [51]	3DMatch	83.6	90.5	95.2	9.0	1.7	57.6	78.8	91.3	17.1	4.2	10.7	0.3
3D MV Reg [89]	3DMatch	87.7	93.2	97.0	6.0	1.2	69.0	83.1	91.8	11.7	2.9	10.2	0.2
BYOC	3DMatch	66.5	85.2	97.8	7.4	3.3	30.7	57.6	88.9	16.0	8.2	9.5	0.9
BYOC-Geo	ScanNet	80.3	92.8	98.8	4.8	2.3	46.5	74.6	94.6	10.6	5.4	7.2	0.5
BYOC + RANSAC	ScanNet	81.3	92.8	98.4	5.6	2.4	37.8	69.7	92.1	13.3	6.4	7.7	0.5
BYOC	ScanNet	86.5	95.2	99.1	3.8	1.7	56.4	80.6	96.3	8.7	4.3	5.6	0.3

Evaluation Metrics. We evaluate the pairwise registration by calculating the rotation and translation error between the predicted and ground-truth transformation as follows:

$$E_{\text{rotation}} = \arccos(\frac{Tr(\mathbf{R}_{pr}\mathbf{R}_{gt}^{\top}) - 1}{2}), \tag{3.7}$$

$$E_{\text{translation}} = ||\mathbf{t}_{pr} - \mathbf{t}_{gt}||_2. \tag{3.8}$$

We report the translation error in cms and the rotation errors in degrees. We also report the chamfer distance between predicted and ground-truth aligned point clouds. For each metric, we report the mean and median errors and recall at different thresholds.

Results. We note that ICP approaches fail on this task. ICP assumes that the point clouds are prealigned and can be very effective at fine-tuning such alignment by minimizing a chamfer distance. However, our view pairs have a relatively large camera motion with the mean relative transformation being 11.4 degrees and 19.4 cm. As a result, ICP struggles with the large transformations and partial overlap between the point cloud pairs. Similarly, FPFH also fails on this task as its output descriptors are not distinctive enough, resulting in many false correspondences that greatly deteriorate the registration performance.

Table 3.2: **Random Visual Features are surprisingly good for registration.** Visual features outperform geometric features with and without training.

	Rotation (deg)		Translation (cm)		Chamfer	
	Mean	Median	Mean	Median	Mean	Median
Random Visual	6.4	2.7	14.9	7.0	9.8	0.6
Random Geometric	21.3	13.0	46.5	28.5	26.0	8.6
BYOC (Visual)	2.7	0.9	6.4	2.6	3.3	0.1
BYOC-Geo	4.8	2.3	10.6	5.4	7.2	0.5
BYOC	3.8	1.7	8.7	4.3	5.6	0.3

On the other hand, learned approaches show a clear advantage in this domain as they are able to learn features that are well-tuned for the task and data domain. Our model is able to outperform FCGF despite FCGF being trained with ground-truth correspondences on an indoor scene dataset. This is true regardless of whether our model is trained using RGB-D or depth pairs. While we find that our model trained on 3D Match performs worse than FCGF, this is expected since 3DMatch is a much smaller dataset making it less suitable for a self-supervised approach.

Finally, our approach is competitive with approaches that use supervision for both feature learning and correspondence estimation [51, 89]. This comparison represents the difference between full supervision on a small dataset vs. self-supervision on a large dataset. Our competitive performance demonstrates the promise of self-supervision and its ability to leverage a very simple learning signal: consistency between video frames.

What is the impact of the transformation estimator? While RANSAC improves the performance of FPFH and FCGF compared to the Weighted Procrustes, we see the opposite pattern with our approach. This is due to our model being trained specifically with a registration loss on filtered correspondence. As a result, Lowe's ratio becomes a very effective method of filtering our correspondences while being less effective for other approaches.

How good are random features? We find that random visual features can serve as a strong baseline for point cloud registration on ScanNet, as shown in Fig. 3.3 and Tab. 3.2. This is surprising since random visual features perform on par with FCGF. This explains why our method is capable of achieving this performance without any supervision. We also find that after training, our visual features achieve the highest registration performance. Those results suggest that visual features are better descriptors for registration, but it is unclear if this a fundamental advantage or if the performance gap can be be resolved through better architectures or training schemes for geometric feature learning.

3.4.2 Correspondence Estimation

We now examine the quality of the correspondences estimated by our method. We evaluate our approach on the 3D Match geometric registration benchmark and follow the evaluation protocol proposed by Deng et al. [61] for evaluating the correspondence recall. Intuitively, feature-match recall measures the percentage of point cloud pairs that would be registered accurately using a RANSAC estimator by guaranteeing a minimum percentage of inliers.

Baselines. We compare our approach against three sets of baselines. The first set is hand-crafted features based on the local geometry around each point [236, 239, 284]. The second set is supervised approaches that use known pose to sample ground-truth correspondences and apply a metric learning loss to learn features for geometric registration. Finally, the third set is unsupervised approaches trained on reconstructed scenes. While those approaches do not directly use ground-truth pose during training, their training data (reconstructed scenes) is generated by aligning 50 depth maps into a single point cloud. Hence, while those approaches do not use pose supervision explicitly, pose information is *needed* to generate their data. We refer to those approaches as *scene-supervised*.

Evaluation Metrics. Given a set of correspondences $\mathcal{C}$, $FM(\mathcal{C})$ evaluates whether the percentage of inliers exceeds τ_2, where an inlier correspondence is defined as having a residual error less than τ_1 given the ground-truth transformation $\mathbf{T}^*$. Feature-match recall is the percentage of point cloud pairs that have a successful feature matching.

$$FM(\mathcal{C}) = \left[\frac{1}{|\mathcal{C}|} \sum_{(p,q)\in\mathcal{C}} \mathbb{1}\big(||\mathbf{x}_p - \mathbf{T}^* \mathbf{x}_q|| < \tau_1\big) \right] > \tau_2 \tag{3.9}$$

Similar to [50, 60, 61], we calculate feature-match recall over all view pairs using $\tau_1 = 10$ cm and $\tau_2 = 5\%$. Prior approaches often generate feature sets without any specified means of filtering them. As a result, they define the correspondence set as the set of all nearest neighbors. Unlike prior work, we output a small set of correspondences after ranking them using Lowe's ratio test.

Results. BYOC achieves high feature match recall, outperforming traditional and scene-supervised approaches while being competitive with supervised approaches. This performance is achieved by only training on the raw RGB-D or depth scans without requiring any additional annotation or postprocessing of the data. This across-dataset generalization is interesting since ScanNet and 3DMatch differ in two key ways. First, 3D Match point clouds are generated by integrating 50 depth frames. As a result, they are denser than single-frame ScanNet point clouds. Second, point cloud pairs in 3D Match have larger

Table 3.3: **Feature-Match Recall on 3D Match.** Our approach achieves better recall than hand-crafted and scene-supervised approaches while being competitive with supervised approaches.

	Training Data		FMR	
	Dataset	Data Format	Recall	St. Dev.
SHOT [239]	-	-	0.238	0.109
USC [284]	-	-	0.400	0.125
FPFH [236]	-	-	0.481	0.150
FPFH [236] (corr)	-	-	0.462	0.198
3D Match [337]	3D Match	Posed Depth	0.596	0.088
PPFNet [61]	3D Match	Posed Depth	0.623	0.108
PerfectMatch [88]	3D Match	Posed Depth	0.947	0.027
FCGF [50]	3D Match	Posed Depth	0.952	0.066
FCGF [50] (corr)	3D Match	Posed Depth	0.932	0.104
CGF [221]	SceneNN	Scenes	0.582	0.142
PPF-FoldNet [60]	3D Match	Scenes	0.718	0.105
3D Point Capsule Networks [350]	3D Match	Scenes	0.787	0.062
BYOC (no filtering)	ScanNet	RGB-D Video	0.662	0.225
BYOC	3D Match	RGB-D Video	0.690	0.172
BYOC	ScanNet	RGB-D Video	0.766	0.181
BYOC-Geo	ScanNet	Depth Video	0.786	0.195

viewpoint changes. Despite those differences, our model can still generalize from ScanNet to 3D Match. This can be attributed to both the voxelization performed by the geometric encoder and the augmentation, which gives the model some degree of equivariance with respect to point cloud density and rotation.

We also observe that BYOC-Geo, which is only trained with geometric correspondence, generalizes better to 3D Match despite doing worse on ScanNet. One explanation for this discrepancy is that bootstrapping with visual correspondences biases the model towards representing features that are meaningful in both modalities. Such representations might be more dataset-specific, hindering across-dataset generalization. This finding opens up the possibility of using datasets that only have depth video; *e.g.*, lidar.

While our best configuration performs on par with the best scene-supervised approach, they outperform us if we do not filter our correspondences. We observe that when we attempt to filter the correspondences for FPFH or FCGF, their performance deteriorates. This is consistent with some of the reported results by [60] where using a larger number of features improved their performance. Hence, it is unclear how correspondence filtering would affect the performance of self-supervised methods. Due to the lack of publicly available implementations of those methods and the complexity of their approach, we were unable to run additional experiments to better understand the impact of the training data and correspondence filtering on the learning process.

3.5 Qualitative Analysis

We visualize the extract features, correspondence, and registration to better understand the model performance. We emphasize that the RGB images are *only* used for visualization; our model operates on the uncolored point cloud computed from depth images. We visualize the feature point clouds by using t-SNE [294] to map each 32-dimensional feature vector to a scalar. We then normalize the values onto the Spectral color map [118] following Choy et al. [50]. We use the features from both input pairs when computing the t-SNE to map similar features to similar colors.

Feature Quality. BYOC extracts features that appear to consistently map the same parts of the scene to similar features as shown in Fig. 3.4. This allows the model to estimate accurate correspondences and register the point clouds. Beyond consistent mapping, we observe that the t-SNE mapping of the features displays sharp discrete jumps around object boundaries. It would be interesting to explore the efficacy of our learned representations for more semantic tasks such as 3D detection or segmentation.

Correspondence estimation. We also find that our model is able to extract accurate correspondences for different scenes and different degrees of overlap. As a result, our model is able to accurately register point clouds under a variety of setups. Furthermore, the density of extracted correspondences allows us to accurately register the object even if some of the correspondences are incorrect.

Failure modes. Finally, we also observe some failure modes in our model that are shown in the bottom 3 rows. In the seventh row, the model's correspondences appear to be mapping points on the sink's surface across the images, resulting in the registration being slightly off. We observe this pattern for images with large, fairly flat surfaces. This can be seen to a larger extent in the eighth row where the model extracts correspondences between random points on the carpet, resulting in very poor registration performance. Finally, the bottom row shows a case where the incorrect depth estimation results in erroneous registration. In this case, the input incorrectly specified the open window as a flat surface, resulting in a large number of incorrect correspondences.

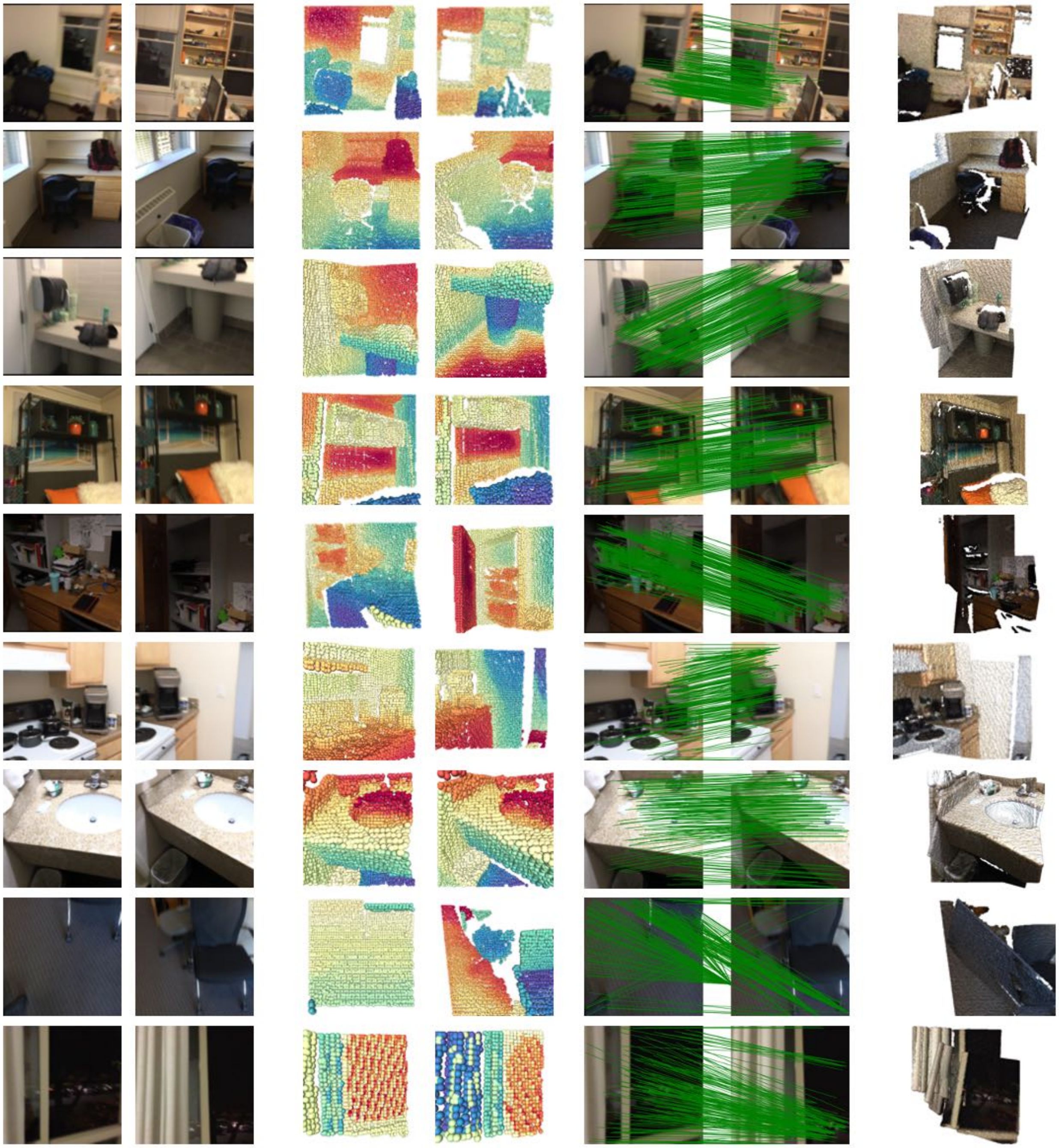


Figure 3.4: **BYOC's geometric features allow for accurate registration by mapping corresponding points to similar feature vectors.** Our approach takes uncolored point clouds as input; images and colored point clouds are presented to aid visualization. We find that the features appear to delineate objects such as chairs and floor edges, resulting in accurate registration. We observe several failure modes that can be caused by relatively flat geometry (rows 7 and 8) or incorrect depth input (row 9).

3.6 Discussion

In this chapter, we presented another way of learning from 3D correspondence in RGB-D video. BYOC simplifies the setup of UnsupervisedR&R while achieving strong performance. Our key insight is that geometric consistency provides a strong enough signal to learn visual representations and support geometric representation learning. This is enabled by the surprisingly strong performance of randomly initialized convolutional neural networks that estimate noisy but relatively accurate correspondences without any training. Our approach takes advantage of visual encoders to provide pseudo-correspondence labels to enable geometric feature learning.

This approach shares some of the limitations in Chapter 2. Namely, it assumes that the transformation between scenes is a rigid body transformation, which makes it difficult to handle learning from dynamic scenes or deformable objects. Furthermore, all our experiments focused on indoor scenes which greatly benefit from being generally static and structured. Finally, both approaches rely on close-by video frames, which limits the viewpoint change. This limitation is quite fundamental to the current approach as it relies on the assumption that overlap exists between frames. However, as discussed in the following chapter, we can alleviate this concern by extending from pairwise to multiview registration. While this does not completely alleviate the issue, it is a step towards better leveraging video data and learning from longer-range consistency.

Chapter 4

Learning from Multiview Consistency

In Chapters 2 and 3, we explored how we can learn 3D correspondence from RGB-D video. In both cases, we use frame pairs with the primary signal provided by video is spatio-temporal consistency and view overlap between close-by frames. However, close-by frame pairs depict small viewpoint changes and miss out on richer long-range consistency between distant overlapping frames. We overcome this challenge by simply sampling more frames for multiview registration. Our key insight is that multiview registration allows us to obtain correspondences over longer time frames; increasing both the diversity and difficulty of our training pairs.

In this chapter, we propose SyncMatch: a self-supervised approach for correspondence estimation that learns from multiview consistency from short RGB-D sequences. We build on the visual stream discussed in Chapter 3 and enhance it with a geometry-aware correspondence estimation algorithm and a RANSAC-inspired transformation fitting algorithm. We estimate pairwise transformations for all pairs and propose a novel SE(3) transformation synchronization algorithm to convert pairwise estimates into a single canonical frame. This allows us to improve the representation learning and handle more frames during training and testing. This chapter is based on work previously published in [79].

4.1 Introduction

Consider the couch in Fig. 4.1. While the start and end frames depict overlapping regions in space, the large viewpoint change makes them appear significantly different. The ability to establish correspondence across views lies at the core of scene understanding and visual tasks such as SLAM and structure-from-motion. The common approach to learning correspondence estimation relies on correspondence supervision; *i.e.*, telling the model which pixels belong to the same point in space. However, we commonly learn about the world by moving and observing how appearance changes without such explicit supervision. *Could we learn correspondence estimation directly from video?*

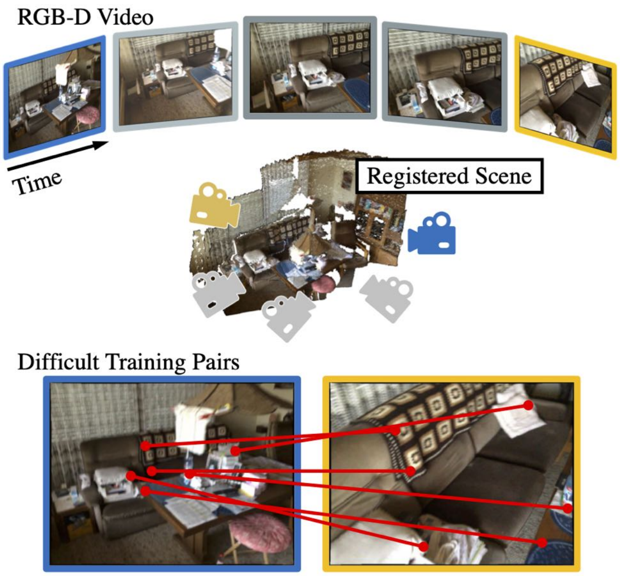


Figure 4.1: Multiview RGB-D registration allows us to learn from wide-baseline view pairs which exhibit more appearance change that adjacent frames.

Modern correspondence estimation approaches rely heavily on supervision. This is typically obtained by applying classical 3D reconstruction algorithms on large image collections [167, 260, 311] or carefully captured indoor scans [35, 56, 57], and then sampling overlapping view pairs for training. This has been widely successful, as it provides correspondence supervision with large viewpoint and lighting changes. While current methods benefit from supervision, they are limited to learning from carefully captured videos that can already be constructed using standard algorithms. Recently, there has been a rise in self-supervised correspondence approaches that rely on close-by frames in video [75, 77, 121, 152]. This sampling strategy limits the appearance change, as shown in Fig. 4.1, resulting in poor performance on image pairs with large viewpoint changes. Ideally, we would leverage the temporal consistency within the video to learn from distant overlapping frames while ignoring non-overlapping pairs.

To this end, we propose SyncMatch: a self-supervised approach for learning correspondence estimation through synchronized multiview pointcloud registration. Our approach bootstraps itself, generating wide-baseline view pairs by registering and synchronizing all pairwise transformations within short RGB-D video clips. Our core insight is that through synchronizing transformations across longer time frames, we can detect and learn from difficult pairs with large viewpoint changes. Despite only relying on geometric consistency within RGB-D videos, we achieve comparable performance to fully-supervised approaches.

Our approach is inspired by self-supervised pointcloud registration [75, 77] and transformation synchronization [7, 89]. The core insight in self-supervised pointcloud registration is that randomly initialized networks provide sufficiently good features for narrow-baseline pointcloud registration. This allows them to provide good pseudo labels for self-supervised training. Meanwhile, transformation synchronization allows us to estimate an accurate camera trajectory from a potentially noisy set of pairwise relative camera poses. Our approach combines both ideas for self-supervised multiview registration, allowing us to learn correspondence estimation across large viewpoint changes.

We evaluate our approach on RGB-D indoor scene videos. We train our model on RGB-D videos from ScanNet and ETH-3D, and evaluate it on correspondence estimation and RGB-D pointcloud registration. Despite only learning from RGB-D video, our approach achieves a similar performance to supervised approaches with more sophisticated matching algorithms. Furthermore, we provide a comprehensive analysis of our approach to understand the impact of the training data and the model components.

4.2 Related Work

Correspondence Estimation. Correspondence estimation is the task of identifying points in two images that correspond to the same physical location. The standard approach for establishing correspondence has two distinct steps: feature extraction and feature matching. Early work exploited hand-crafted feature detectors [179, 192] to extract normalized image patches across repeatable image points, combined with hand-crafted descriptors based on local statistics to obtain features with some robustness to illumination changes and small translations [5, 15, 179]. These features are matched via nearest neighbor search and filtered using heuristics; *e.g.*, ratio test [179] or neighborhood consensus [242]. With the advent of deep learning, learnt keypoint detectors [151, 157], descriptors [14, 283], and correspondence estimators [227, 241, 330] have been proposed. These models are trained using correspondence supervision from traditional 3D reconstruction algorithms [57, 243] on image collections of tourist landmarks [167, 260, 311] or indoor scene video scans [35, 56, 337]. Other approaches have explored self-supervision using synthetic data [66, 191, 226, 354], traditional descriptors [325], or RGB-D pairs from video [75, 77]. Our work shares the motivation of self-supervised approaches and extends it to learning from multiview consistency to better exploit the rich signal in video.

Pointcloud Registration. Pointcloud registration is the task of finding a transformation that aligns two sets of 3D points. Early work assumes access to perfect correspondence and devised algorithms to estimate the rigid body transformation that would best align them [8, 130, 292]. Later work proposed robust estimators that can handle correspondence errors and outliers that arise from feature matching [82, 348]. More recently, learned counterparts have been proposed for 3D keypoint descriptors [50, 62, 96], correspondence estimation [26, 27, 51, 89, 115, 221, 328], as well as models for directly registering a pair of pointclouds [62, 108, 153, 180]. Closest to our work are approaches that use RGB-D video to learn correspondence-based pointcloud registration [75, 77, 306]. Similar to our approach, they learn from the geometric consistency between RGB-D frames in a video. However, unlike those approaches, we train on short sequences of videos instead of frame pairs, allowing us to train on view pairs with larger camera changes.

SE(3) Transformation Synchronization Given a set of pairwise estimates, synchronization estimates that set of latent values that explain them. Transformation synchronization refers to this problem applied to SO(3) and SE(3) transformations, as it commonly arises in SLAM settings [30, 149, 200, 238]. For video, one could naively only consider adjacent pairs to construct a minimum spanning tree and aggregate the transformations. However, this only works if all pairwise estimates are accurate since a single bad estimate can be detrimental. More robust approaches have been proposed that can leverage the information from multiple (or all) edges in the graph [22, 113, 114, 116, 149, 228]. Most relevant to our work are Arrigoni et al. [6, 7] and Gojcic et al. [89]. Arrigoni et al. [6, 7] propose a closed-form solution to SO(3) and SE(3) synchronization based on the eigendecomposition of a pairwise transformation matrix. Gojcic et al. [89] builds on those ideas and integrates transformation synchronization with a supervised end-to-end pipeline for multiview registration. We are inspired by this work and propose a different approach to SE(3) synchronization based on iterative matrix multiplication, which allows for accurate synchronization while being more numerically stable. Furthermore, unlike prior work [89, 114, 228], we use transformation synchronization for learning without supervision.

4.3 Correspondence Estimation via Multiview Registration

We learn correspondence estimation from multiview registration of short RGB-D sequences without relying on pose or correspondence supervision. We first provide a high-level sketch of our approach, shown in Fig. 4.2, before discussing each component in detail.

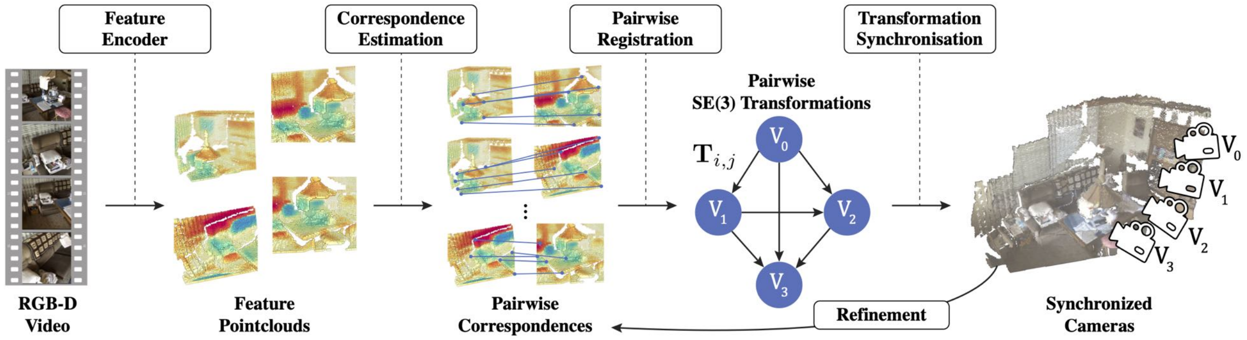


Figure 4.2: **SyncMatch.** Given a sequence of N RGB-D video frames, we extract features for each image and project them to a 3D pointcloud using input depth. We extract all pairwise correspondences to estimate pairwise SE(3) transformations. We then synchronize the pairwise transformations to register the scene. Finally, we use the estimated registration to refine our correspondence and transformation estimates. We compute correspondences losses for both the initial and refined registrations, and backpropagate them to the feature encoder.

Approach Sketch. Given N RGB-D frames, we extract features for each RGB image and project them onto a 3D pointcloud using input depth and camera intrinsics. We then extract correspondences between all pointcloud pairs and estimate pairwise SE(3) transformations. Given $\binom{N}{2}$ pairwise transformations, we apply transformation synchronization to find the N camera extrinsic parameters in a shared global frame. Given this coarse alignment, we resample correspondences based on both feature and spatial proximity. We repeat the registration using the updated correspondence. Finally, we compute the loss using the estimated correspondences and SE(3) transformations and backpropagate it to the feature encoder.

4.3.1 Feature Pointcloud

We use a randomly initialized ResNet-18 for feature extraction. While correspondence estimation methods often rely on keypoint detection, our approach is detector-free and generates dense features uniformly across the image. Similar to Sun et al. [272], we generate the feature grid at a lower resolution ($^1/_4$) than the input image. For each frame i, we use the input depth map and camera intrinsics to project the features into a pointcloud $\mathcal{P}_i$ where each point $p \in \mathcal{P}_i$ has a feature $\mathbf{f}_p$ and a 3D coordinate $\mathbf{x}_p$.

4.3.2 Correspondence Estimation

We estimate feature correspondences for all view pairs (i, j). We first generate an initial set of correspondences by finding for each point in image i the point in image j with the closest matching feature vector. The initial correspondence set will include mismatches due to poor matching, repetitive textures, or occlusion.

Correspondences can be filtered using measures of unique matches or geometric consistency. Heuristics such as Lowe's ratio test [179] prefer unique matches and have been extended to self-supervised pointcloud registration [75, 77] and attention-based matching [112, 241, 272]. Geometric consistency relies on the idea a geometrically consistent set of correspondences is likely correct. This can be done by estimating the transformation similar to RANSAC [82] or directly estimating the inlier scores [51, 221, 330]. We use the ratio test for initial alignment and leverage geometric consistency for refinement (Sec. 4.3.5). Specifically, we compute a ratio between the cosine distances in feature space as follows:

$$w_{p,q} = 1 - \frac{D(p,q)}{D(p,q')}, \tag{4.1}$$

where $D(p, q)$ is the cosine distance between the features, and q and q' are the first and second nearest neighbors to point p in feature space. We use the weights to rank the correspondences and only keep the top k correspondences. This results in a correspondence set $\mathcal{C}_{i,j}$ for each pair of frames. The correspondences $(p, q, w_{p,q}) \in \mathcal{C}_{i,j}$ consists of the two matched points and the match weight.

4.3.3 Pairwise Alignment

For each pair of frames, we can identify a transformation $\mathbf{T}_{i,j} \in \text{SE}(3)$ that minimizes the weighted mean-squared error between the aligned correspondences across the images:

$$\mathbf{T}_{i,j} = \mathop{\arg\min}_{\mathbf{T}\in\text{SE}(3)} \sum_{(p,q,w)\in\mathcal{C}_{i,j}} w||\mathbf{x}_q - \mathbf{T}(\mathbf{x}_p)||_2^2. \tag{4.2}$$

A differentiable solution is given by the Weighted Procrustes (WP) algorithm [51], which adapts the classic Umeyama pointcloud alignment algorithm [130, 292].

WP-RANSAC. While the WP algorithm can handle small errors, it is not robust against outliers. El Banani et al. [77] propose combining the alignment algorithm with random sampling to increase robustness. However, a single large outlier can still perturb the solution since solutions are still ranked according to the average residual error on all matches.

We modify the WP algorithm to more closely resemble classic RANSAC [82]. We randomly sample k correspondence subsets, estimate k transformations, and compute an inlier score based on each transformation. We choose the transformation that maximizes the inlier score instead of minimizing the weighted residual error [77], making us more robust to large outliers. Finally, we update the correspondence weights with the inlier scores, which can zero out large outliers. We compute the final registration using the WP algorithms with the updated weights, maintaining differentiability with respect to the correspondence weights.

4.3.4 SE(3) Transformation Synchronization

Given estimates for the $\binom{N}{2}$ camera-to-camera transformations, we want to find the N world-to-camera transformations that best explain them. Arrigoni et al. [6, 7] propose a closed-form solution to SE(3) synchronization using spectral decomposition. This approach was later extended to end-to-end learning pipelines [89, 114]. The approach operates by constructing a block matrix of pairwise transformations where block (i, j) corresponds to the transformation between camera i to camera j. The core insight in that line of work is that the absolute transformations constitute the basis of the pairwise transformations matrix and, hence, can be recovered using eigendecomposition.

While those approaches are successful for inference, they suffer from numerical instabilities during training. This is caused by the backward gradient of the eigendecomposition scaling with $\frac{1}{\min_{i \neq j} \lambda_i - \lambda_j}$ where λ are the eigenvalues. Given that the rank of a perfect SE(3) pairwise matrix is 4, $\min_{i \neq j} \lambda_i - \lambda_j$ approaches 0 for an accurate pairwise matrix which results in exploding gradients. We observed this instability during training. To avoid this instability, we compute the relevant part of the eigendecomposition by power iteration, similar to PageRank [207]. We observe that this converges quickly while being stable during training. We refer the reader to the appendix for more details.

Pairwise confidence. Synchronization is more effective when provided with confidence weights for each of the pairwise estimates. While prior approaches train separate networks to estimate pairwise confidence [88, 114], we instead opted for a simpler approach. We observe that the mean confidence weight is well correlated with view overlap as shown in Fig. 4.3, where pairwise confidence is computed as $\hat{c}_{i,j} = \frac{1}{|\mathcal{C}_{i,j}|} \sum\limits_{w \in \mathcal{C}_{i,j}} w$.

While confidence is correlated with view overlap, non-overlapping frames still receive non-zero confidence. Incorrect transformations from non-overlapping pairs can negatively affect both synchronization and learning. We address this by rescaling the confidence

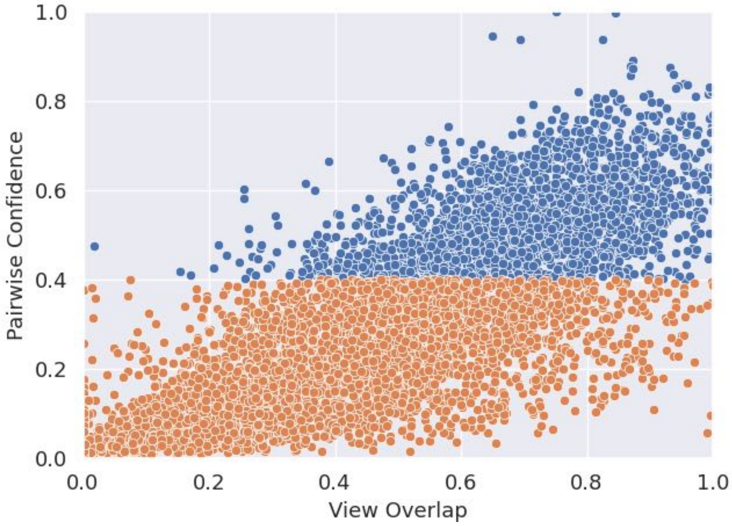


Figure 4.3: **Mean correspondence confidence is an effective overlap filter.** While confidence is not perfectly correlated with view overlap, simple thresholding can accurately filter out low- and no-overlap view pairs.

values based on a threshold to ignore any non-overlapping pairs. While this criterion is simple, it has many false negatives, as shown in Fig. 4.3. To ensure that synchronization is always possible, we exclude adjacent pairs from rescaling since we know they most likely overlap. The pairwise confidence terms are adjusted as follows:

$$c_{i,j} = \begin{cases} \max(0,\ \hat{c}_{i,j} - \gamma)/(1-\gamma) & \text{if } |i-j| > 1, \\ \hat{c}_{i,j} & \text{otherwise.} \end{cases} \tag{4.3}$$

where γ is the confidence threshold. We only estimate pairwise correspondences for pairs (i, j) where $i < j$ to avoid repeated calculations. For pairs where $j > i$, we set $c_{j,i} = c_{i,j}$ and $T_{j,i} = T_{i,j}^{-1}$.

Pairwise Transformation Matrix. We form a block matrix $\mathbf{A}$ using the weighted transformations as follows:

$$\mathbf{A} = \begin{bmatrix} c_1\mathbf{I}_4 & c_{1,2}\mathbf{T}_{1,2} & \cdots & c_{1,N}\mathbf{T}_{1,N} \\ c_{2,1}\mathbf{T}_{2,1} & c_2\mathbf{I}_4 & \cdots & c_{2,N}\mathbf{T}_{2,N} \\ \vdots & \ddots & & \vdots \\ c_{N,1}\mathbf{T}_{N,1} & c_{N,2}\mathbf{T}_{N,2} & \cdots & c_N\mathbf{I}_4 \end{bmatrix}, \tag{4.4}$$

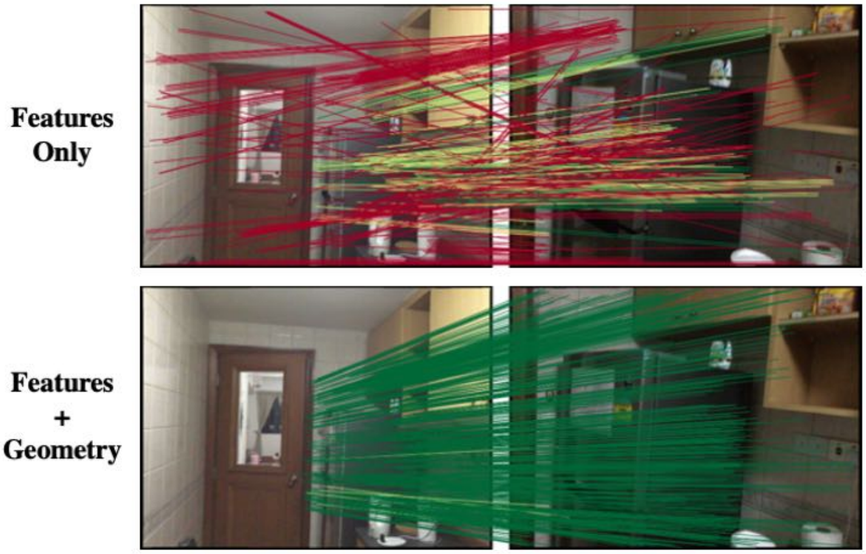


Figure 4.4: **Geometry-aware sampling greatly improves our correspondence set.** GART allows us to extract accurate correspondence even if the initial feature matches are very noisy.

where $c_{i,j}$ is the pairwise confidence, $\mathbf{T}_{i,j}$ is the estimated pairwise transformation, and $c_i = \sum_{k \in N} c_{i,k}$. We perform t matrix multiplications to calculate $\mathbf{A}^{2^t}$ and extract the synchronized transformations by taking the first block column and normalizing each transformation by its confidence (bottom right element). This results in N SE(3) transformations in the first view's frame of reference.

4.3.5 Refinement

While feature-based matching is powerful, it can produce false positive feature matches that could be easily filtered out through geometric consistency. To this end, we use the predicted scene alignment to refine our correspondences by filtering matches that are not geometrically consistent with the estimated transformation. We resample the correspondences but compute the ratio test based on both feature similarity and spatial proximity. We update our correspondence filtering criteria by changing the distance function in the ratio test to:

$$D_{refine}(p, q) = D_C(\mathbf{f}_p, \mathbf{f}_q) + \lambda \|\mathbf{x}_p - \mathbf{x}_q\|_2, \tag{4.5}$$

where $D_C(x, y)$ is cosine distance, $\mathbf{f}_p$ is the feature vector belonging to point p, λ is a weighing constant, and $\mathbf{x}_p$ is the aligned 3D coordinate of point p in a common global frame. We refer to this updated ratio test as a Geometry-Aware Ratio Test (GART).

4.3.6 Losses

We emphasize that our approach is self-supervised. Hence, we only rely on the consistency within the video for training. We use the registration loss proposed by El Banani and Johnson [75], which minimizes the weighted residual error of the estimated correspondences using the estimated alignment. For a given pair (i, j), we compute a registration loss as follows:

$$\mathcal{L}_{reg}(i,j) = \sum_{(p,q,w)\in\mathcal{C}_{i,j}} w \|\mathbf{T}_j^{-1}\mathbf{x}_q - \mathbf{T}_i^{-1}\mathbf{x}_p\|_2, \tag{4.6}$$

where p and q are the corresponding points, w is their weight, and $\mathbf{T}_i$ is the synchronized transformation (Sec. 4.3.4). We compute this loss for both the initial and refined correspondence sets and predicted transformations for all view pairs.

4.4 Experiments

We evaluate our approach on two indoor scene datasets: ScanNet [56] and ETH3D [244]. Our experiments address the following questions: (1) does multiview training improve over pairwise training?; (2) can multiview self-supervision replace full-supervision?; (3) can we reconstruct scenes from RGB-D sequences?; (4) can we learn from videos that cannot be reconstructed using standard approaches?

Training Dataset. ScanNet provides RGB-D videos of 1513 scenes and camera poses computed using BundleFusion [57]. We use the train/valid/test scene split from ScanNet v2. While automated reconstruction models like BundleFusion are able to reconstruct ScanNet scenes, ETH3D [244] videos are more challenging to such systems, with BundleFusion failing on most of those sequences. As a result, ETH3D offers us an interesting set of videos that cannot be currently used for supervised training. We emphasize that we only use RGB-D video and camera intrinsics for training, and any provided camera poses are only used for evaluation. We generate view pairs by sampling views that are 20 frames apart. For longer sequences, we combine adjacent pairs to get N-tuples.

Training Details. We train our model with the AdamW [137, 178] optimizer using a learning rate of 10^{-3} and a weight decay of 10^{-3}. We train for 100K iterations with a batch size of 16. Unless otherwise stated, we use 6 views for training. Our implementation is in PyTorch [211], with extensive use of PyTorch3D [224], FAISS [127], PyKeOps [38], and Open3D [353]. We make the code publicly available.[1]

[1] https://github.com/facebookresearch/SyncMatch

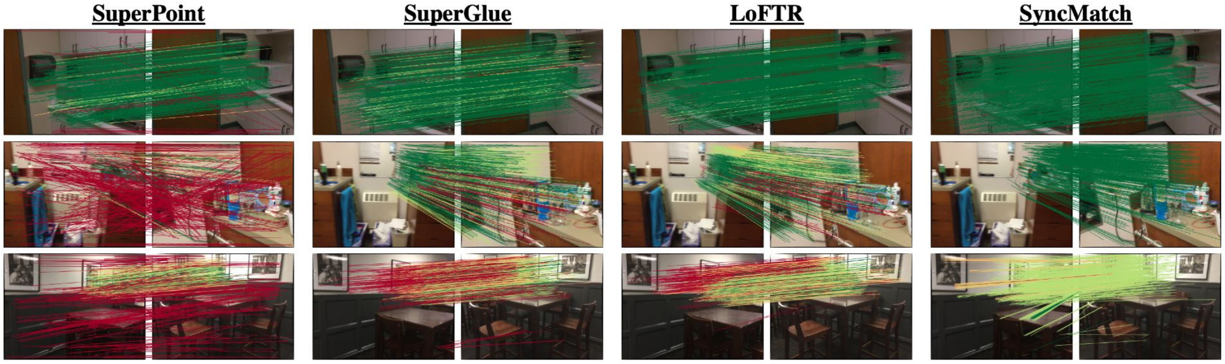


Figure 4.5: **Correspondence Estimation on ScanNet.** Our model extracts accurate correspondences for large viewpoint change. Through combining both strong feature descriptors and geometric refinement, we can successfully handle cases where prior approaches fail. Correspondences are color-coded by 3D error.

4.4.1 Correspondence Estimation

We evaluate our model on correspondence estimation. While our model is trained on adjacent pair sequences, the primary challenge is how it performs for wide-baseline correspondence estimation. We evaluate this on the test set proposed by SuperGlue [241] of 1500 view pairs. This dataset includes difficult pairs with large camera motions; *e.g.*, images from opposite sides of a room.

Evaluation Metrics. We evaluate the estimated correspondences based on their 2D and 3D errors. We lift the estimated correspondences into 3D using known depth and intrinsics and only consider keypoints with valid depth values. We use ground-truth transformations to align the keypoints and compute the 3D error and the 2D reprojection error. We extract 500 correspondences for all methods to allow for a meaningful comparison between precision values.[2]

Baselines. We compare our approach against classic, self-supervised, and supervised correspondence estimation methods. First, we compare against two commonly used feature descriptors: RootSIFT [5] and SuperPoint [66]. RootSIFT is a hand-crafted feature that is still used in modern pipelines, while SuperPoint is a self-supervised descriptor trained on synthetic data and affine transformations of web images. We report the performance of these features for image-only matching using the ratio test [179] as well as depth-refined matching using our proposed GART.

[2] LoFTR and SuperGlue use a mutual check heuristic for matching, which can produce fewer correspondences. In such cases, we use all the correspondences they produce.

Table 4.1: **Wide-Baseline Correspondence Estimation on ScanNet.** SyncMatch extracts accurate wide-baseline correspondences; performing on-par with supervised methods. Our proposed GART uses estimated alignment to sample more accurate correspondences regardless of the underlying feature descriptor.

	3D Corres.			2D Corres.		
Method	1cm	5cm	10cm	1px	2px	5px
Unsupervised Features with Heuristic Matching						
RootSIFT [5]	7.1	29.2	35.1	2.8	8.6	22.9
RootSIFT [5] + GART	14.1	72.4	84.3	3.8	12.8	42.8
SuperPoint [66]	7.5	41.4	51.3	2.5	8.6	29.5
SuperPoint [66] + GART	16.8	73.7	84.3	4.7	15.5	47.9
BYOC† [75]	12.8	53.3	63.0	4.5	14.6	41.9
BYOC† [75] + GART	22.8	73.1	81.4	6.0	19.6	54.0
SyncMatch (**Ours**)	13.1	55.1	65.4	4.6	15.3	43.9
SyncMatch (**Ours**) + GART	**26.8**	76.5	84.4	**7.5**	**23.5**	**59.7**
Supervised Features with Trained Matching						
SuperGlue [241]	8.7	62.4	78.7	2.5	9.0	36.9
SuperGlue [241] + GART	13.8	74.8	87.7	3.3	11.7	44.4
LoFTR [272]	16.0	72.2	84.6	5.6	18.5	55.5
LoFTR [272] + GART	21.4	**80.8**	**90.2**	6.5	21.2	59.0

We consider three end-to-end approaches: SuperGlue [241], LoFTR [272], and BYOC [75]. SuperGlue is an attention-based matching algorithm trained with SuperPoint features. LoFTR and BYOC are detector-free approaches that train features from scratch. SuperGlue and LoFTR use transformers for matching that are trained with correspondence supervision. BYOC is self-supervised and uses a variant of the ratio test for matching.

We take several measures to ensure a comprehensive fair comparison. First, we update BYOC's visual backbone from ResNet-5 to ResNet-18. This results in a stronger and fairer baseline which we refer to as BYOC†. Second, SuperGlue and LoFTR are both fully supervised image-based approaches that use transformers for feature matching. Hence, it was unclear how to adapt their matching algorithm to use depth information. Instead of adapting their matching, we use GART to re-rank their proposed matches and only retain the top set of matches. This resulted in large performance improvements as seen in Tab. 4.1

How does heuristic matching perform? We find that well-tuned matching allows hand-crafted features to achieve a strong performance against learned features, as observed by Efe et al. [72]. Nevertheless, self-supervised feature descriptors still retain a performance advantage. Furthermore, our proposed approach outperforms both hand-crafted and self-supervised descriptors regardless of whether depth is used at test time for refinement.

Table 4.2: **Pairwise Registration on ScanNet.** SyncMatch outperforms all approaches on narrow-baseline registration, while under performing some methods for wide-baseline registration. WP-RANSAC results in large performance gains for all methods across metrics.

	Narrow Baseline		Wide Baseline	
	AUC_{5°	$\text{AUC}_{10\text{cm}}$	AUC_{5°	$\text{AUC}_{10\text{cm}}$
Unsupervised Features with Heuristic Matching				
RootSIFT [5] + RANSAC	36.7	28.9	15.5	11.1
SuperPoint [66] + RANSAC	50.3	39.8	24.9	17.0
RootSIFT [5] + **Ours**	84.3	77.9	64.4	52.3
SuperPoint [66] + **Ours**	83.8	77.0	61.7	49.2
BYOC† [75]	84.6	77.6	60.4	48.4
SyncMatch **(Ours)**	85.3	78.8	63.4	50.5
Supervised Features with Trained Matching				
SuperGlue [241] + RANSAC	65.7	54.0	47.6	33.2
LoFTR [272] + RANSAC	75.0	64.6	57.2	41.8
SuperGlue [241] + **Ours**	82.3	75.0	66.0	51.2
LoFTR [272] + **Ours**	84.5	78.1	70.5	56.2

Can self-supervision replace correspondence supervision? While our approach outperforms classic and self-supervised approaches, it still underperforms the strongest supervised approaches. This is expected since we use pseudo correspondence labels from short sequences, whereas supervised approaches are trained with view pairs that were sampled the same way as the test pairs [241]. Nevertheless, we argue that our approach is still promising, as it can match SuperGlue supervised features and matching despite being self-supervised and using the ratio test for matching.

Does geometry-based refinement help? We find that our proposed geometry-based refinement improved the performance for all methods. Furthermore, the improvement is most pronounced for our method, which performs on par with supervised methods when using depth and outperforms them for some thresholds.

4.4.2 Pointcloud Registration

We next evaluate pairwise and multiview pointcloud registration performance. We evaluate pairwise registration using view pairs extracted from ScanNet. We also evaluate our model's ability to scale up to large sequences by reconstructing the challenging sequences in the ETH-3D dataset.

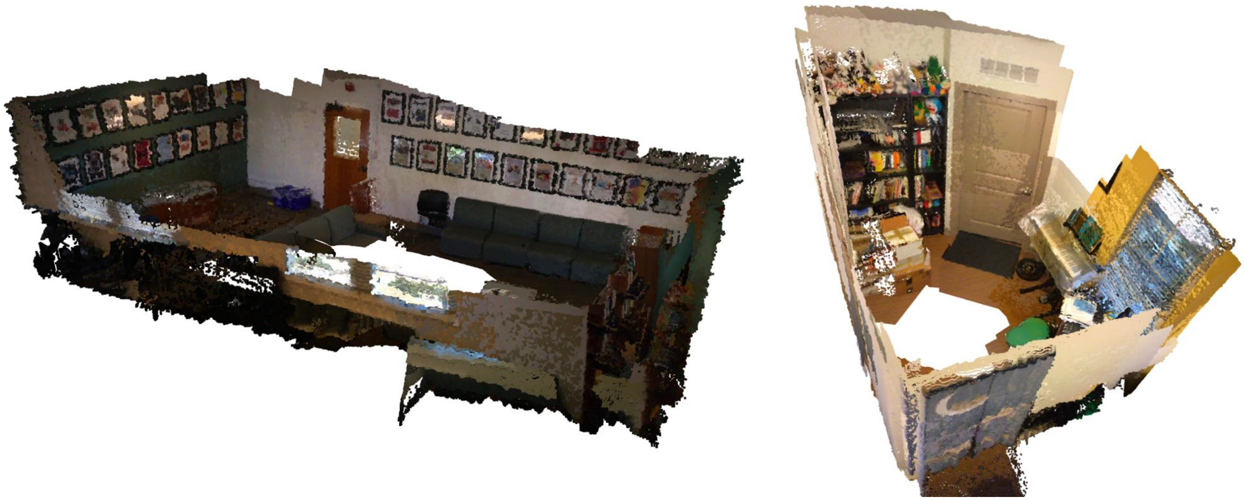


Figure 4.6: **RGB-D Scene Reconstruction.** SyncMatch can scale at inference time to longer videos to reconstruct longer sequences.

Pairwise Registration. We evaluate the approaches on pairwise registration for both narrow-baseline and wide-baseline view pairs, as shown in Tab. 4.2. We evaluate narrow-baseline view pairs similar to BYOC and wide-baseline view pairs similar to SuperGlue. We report the area under the curve for pose errors with a threshold of $5°$ and 10cm for rotation and translation errors, respectively. For RootSIFT and SuperPoint, we compute correspondences using the ratio test, while SuperGlue and LoFTR provide us with matches. We use either Open3D's [353] RANSAC or our proposed WP-RANSAC for alignment. For SyncMatch and BYOC, we use the method's estimated transformation. We report numbers without depth refinement to avoid confounding the evaluation.

Our approach outperforms all baselines for narrow-baseline registration but underperforms several in wide-baseline registration. This is surprising given our model's strong performance in wide-baseline correspondence estimation, but probably arises from the domain shift from the mostly narrow-baseline pairs used for training. Furthermore, we note that our proposed alignment algorithm greatly improves the performance of all baselines, especially RootSIFT. We observe this improvement from supervised models with trained correspondence estimators, suggesting that their predicted correspondences still contain significant structured errors that benefit from robust estimators that can utilize the match confidence weights.

Scaling up to longer sequences. Computing pairwise correspondence for N frames scales as $\mathrm{O}(N^2)$, which is problematic for longer videos. However, with minor reformula-

Table 4.3: **Impact of number of training views.** Training with more views improves correspondence estimation performance.

	3D Corres.			2D Corres.		
Number of Views	1cm	5cm	10cm	1px	2px	5px
2	24.7	73.9	81.6	6.2	19.8	54.0
3	26.2	76.8	85.2	7.0	22.6	58.8
4	26.8	75.4	83.5	7.5	23.3	59.2
5	26.9	75.9	83.6	7.6	23.5	59.7
6	26.8	76.5	84.4	7.5	23.5	59.7

tion, SyncMatch can be adapted to significantly reduce its run-time. Instead of considering all pairs, we only consider adjacent frames in the first step to give us an approximate camera trajectory from $N-1$ pairs. We use this trajectory to refine the correspondences and then consider all frames within a specific window w; *i.e.*, only consider frames (i, j) if $|i-j| < w$. We can then run the synchronization step with the confidence of all other pairs set to 0. This allows us to scale the model's run-time linearly with the number of frames, instead of quadratically, which allows us to handle hundreds of frames. We visualize two reconstructions from ScanNet in Fig. 4.6.

We apply our adapted model to the challenging ETH3D dataset. ETH-3D sequences are challenging to traditional RGB-D reconstruction models, with BundleFusion failing on nearly 75% of the sequences. SyncMatch can reconstruct 33/61 training scenes and 16/35 test scenes. This outperforms standard systems such as BundleFusion (14/61 and 7/35) and ORB-SLAM [199] (25/61 and 16/35). Since such systems are often used to generate annotations automatically for RGB-D video datasets, SyncMatch's strong performance against them shows the promise of self-supervised approaches to using videos that are currently missing from large-scale RGB-D scene datasets. We emphasize that our model was not designed for full-scene reconstruction; this evaluation is only intended to showcase our model's performance against existing methods for automatically generating pose annotation for RGB-D videos.

4.4.3 Analysis

We analyze our model through a series of experiments aimed at answering some key questions regarding its performance. The analysis experiments are aimed at understanding the impact of the multiview setting, training data, as well as the impact of our model's components during both training and inference.

Table 4.4: **Ablation Experiments.** While WP-RANSAC is crucial to model performance, the impact of other components depends on downstream tasks.

	Ablation		Pairwise Correspondence		Multiview Registration	
	Train	Test	AUC$_{10cm}$	AUC$_{10px}$	AUC$_{5^\circ}$	AUC$_{10cm}$
Full Model			65.9	48.1	83.4	77.8
Naive Synchronization	✗		65.7	48.9	82.8	76.2
No Depth Refine	✗		66.0	49.0	82.9	76.3
No Confidence Threshold	✗		35.2	20.4	17.9	14.5
No WP-RANSAC	✗		1.2	0.2	1.1	3.3
Naive Synchronization		✗	65.9	48.1	82.6	76.9
No Depth Refine		✗	29.7	19.4,	83.4	77.7
No Confidence Threshold		✗	65.9	48.1	76.9	70.9
No WP-RANSAC		✗	42.2	27.7	74.4	66.0

What's the impact of the number of training views? We train our model with a variable number of training views to understand the impact of multiview training. We observe that increasing the number of training views results in progressively better performance. However, the performance gains saturate after 5 views. This could be explained by how ScanNet videos were captured: often, the camera is moving laterally, and after 5 frames (3 seconds), there is often no overlap. Our results suggest that using more views for training will not help until enough frames are used to provide loop closure for the model to learn from.

How do the different components contribute to the model performance? We analyze the impact of various model components through a series of ablations that are applied either during training or testing. We report the performance for both pairwise correspondence estimation (3D and 2D correspondence error) as well as multiview registration with 6 views (rotation and translation error) in Tab. 4.4. Similar to prior work [77], we observe that ablations that make it harder to register the point cloud can boost correspondence performance when ablated during training; *e.g.*, using naive synchronization based only on adjacent views or training without depth refinement. However, replacing WP-RANSAC with WP prevents the model from learning due to inaccurate registration early during training. We also observe that almost all test-time ablations result in worse performance. One surprising exception is removing depth refinement, which greatly reduces wide-baseline correspondence estimation accuracy while not impacting multiview registration. This could be explained by the performance saturating for narrow-baseline registration such that depth refinement is not needed there.

Table 4.5: **Training on ETH-3D data.** Our model is capable of learning for the more challenging videos in ETH-3D.

	3D Correspodence			2D Correspodence		
Training Set	1cm	5cm	10cm	1px	2px	5px
ETH-3D	17.4	66.4	74.5	4.8	16.1	47.3
ScanNet Mini	18.5	67.0	75.6	4.8	16.2	47.6

Can we learn from more challenging sequences? While ScanNet offers a large set of videos, these videos are carefully captured to ensure successful downstream 3D reconstruction. We investigate whether our approach can be trained on more challenging videos, such as ETH-3D. ETH-3D has fewer and more erratic videos, forcing us to reduce the view stride during training. However, our model can still learn without supervision on ETH-3D videos, and the features can be used for wide-baseline correspondence estimation as shown in Tab. 4.5. For comparison, we train on a subset of ScanNet to match a number of instances and view spacing. Both models achieve similar performance. This suggests that our approach could scale to challenging videos beyond carefully captured ScanNet sequences.

4.5 Qualitative Analysis

We include additional qualitative results to provide a better sense of our model's performance. We also clarify some of the color schemes used throughout the chapter.

Correspondence color. We color-coded our correspondence using their 3D error. Specifically, correspondences with an error of less than 5 cm were plotted in dark green, errors between 5 cm and 10 cm were plotted in yellowish green, errors between 10 cm and 15 cm were plotted in orange, and errors larger than 15 cm were plotted in red.

Correspondence Estimation. We provide additional qualitative examples of correspondence estimation results in Fig. 4.7. We find that for easy cases that involve the camera panning or zooming, all approaches perform fairly well (rows 1-3). Meanwhile, cases with large camera rotation can be challenging to all models, with different models failing for different cases. We find that our model can overcome those challenges in some cases where some prior approaches have limited performance (rows 4-9). In cases with repeated textures, our model can inaccurately predict a consistent set of correspondences that are accurate, as shown in row 10 of Fig. 4.7. We find that LoFTR can succeed in such a case, likely due to its use of cross-attention, which is noted by the authors of both LoFTR and SuperGlue. Future iterations of self-supervised correspondence estimation should explore the incorporation of attention modules and integrate it with geometric-aware matching. Finally, we observe that some cases are challenging to all models, especially when there is a very large camera motion, such as looking at the same object from opposing sides (row 11) or when there is limited overlap and plain textures (row 12).

Correspondence Refinement. We provide qualitative examples of estimated correspondences before and after refinement in Fig. 4.8. In many cases, the initial feature-based correspondences are already fairly accurate. In those cases, we find that refinement results in the correspondences being more spread out and increasing in accuracy. More interesting cases involve a very noisy initial set that can be refined into a dense, accurate set of correspondences. This can be seen clearly in rows 5-7 in Fig. 4.8. Finally, in some difficult cases, our initial estimation is extremely noisy, and our model is unable to recover from that.

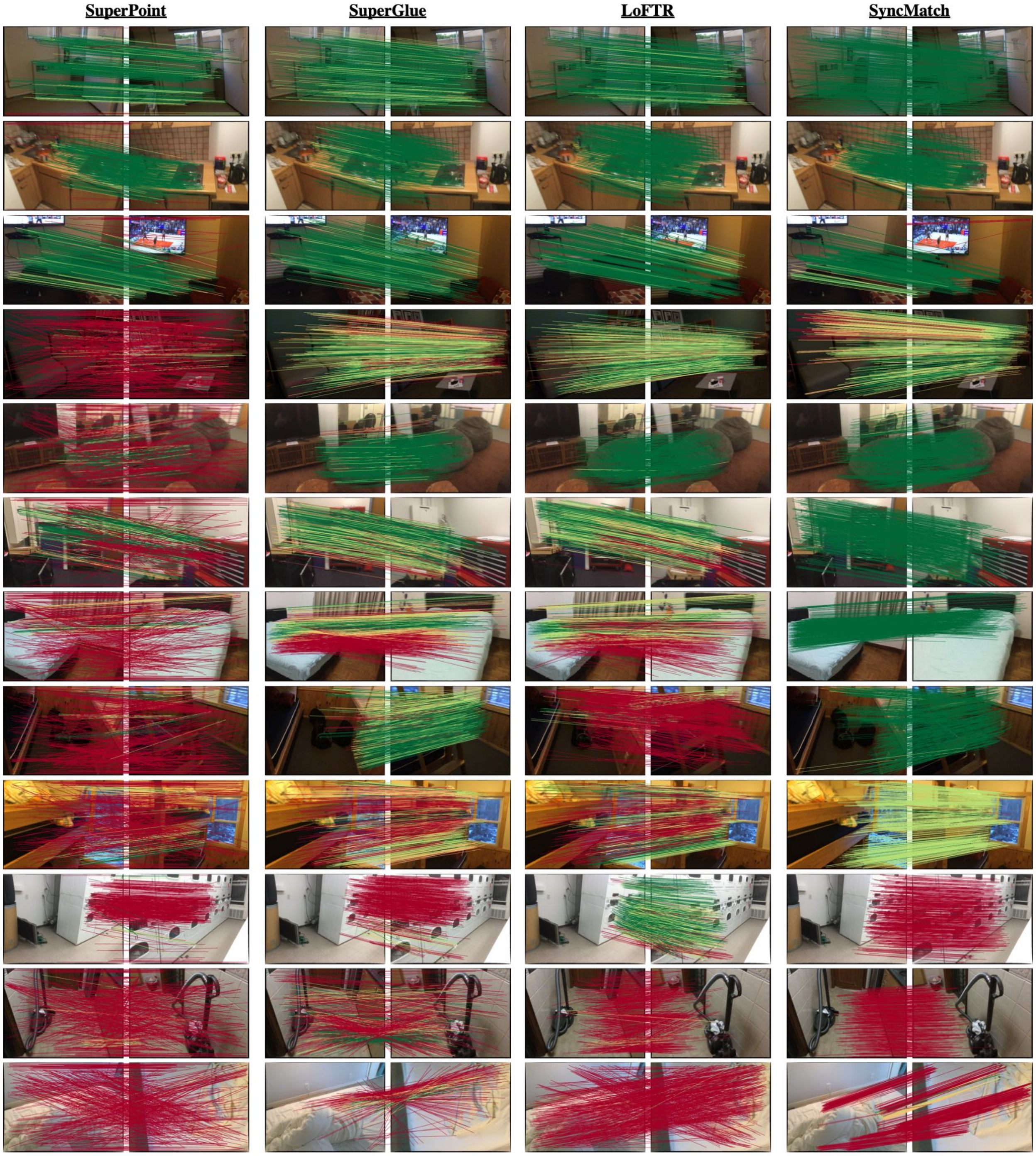


Figure 4.7: **Correspondence Estimation.** We observe several modes of qualitative results. The top three rows present examples where all models perform well. The following six rows present cases where our model succeeds and other models perform poorly. Those are cases where geometric refinmenet allows us to refine a noisy set of correspondence. Finally, we report some challenging cases for our model: repetitive textures (row 10), very large camera motion (row 11), limited overlap and plain textures (row 12). In some cases, we find that LoFTR perfroms better for large camera motions.

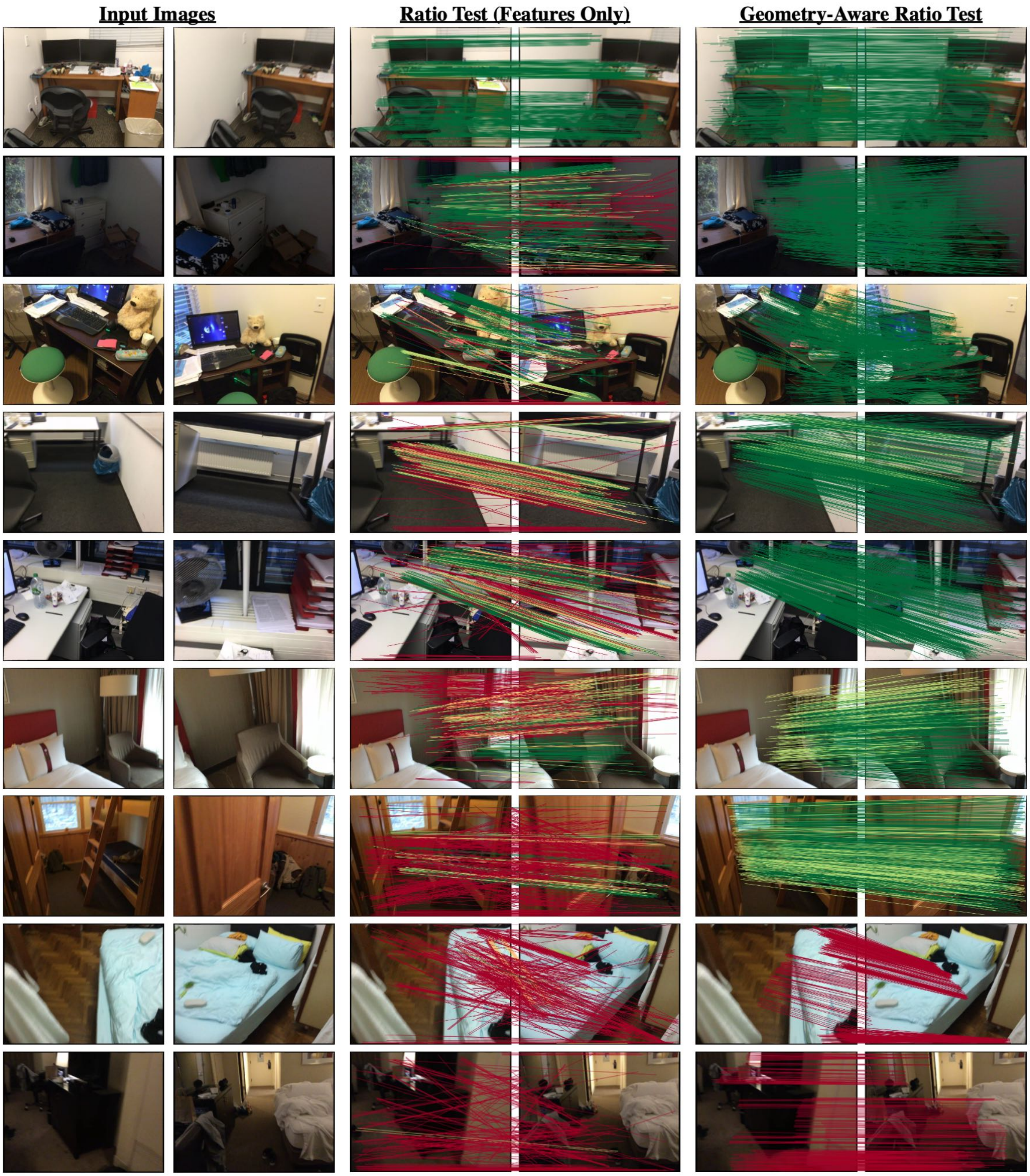


Figure 4.8: **Correspondence Refinement.** We show the impact of correspondence refinement using depth. In the top three rows, we find that feature-based correspondences are already accurate and incorporating geometry simply improves the accuracy. The following four rows present cases where refinement greatly improves the correspondence quality as the feature based correspondence had a small subset of accurate matches. Finally, the last two row present cases where the initial correspondence is so noisy that the model cannot generate a good transformation estimate rendering refinement useless.

4.6 Discussion

In this chapter, we presented a self-supervised approach to learning correspondence estimation that relies on multiview registration to learn from difficult view pairs. Our core insight is that multiview registration allows us to leverage the consistency within short video sequences to obtain difficult view pairs for learning. To this end, we propose a series of components that integrate self-supervised correspondence estimation with multiview registration in a single end-to-end differentiable pipeline. Our approach follows the conventional registration pipeline with several technical contributions that improve registration and correspondence estimation performance while maintaining differentiability.

This chapter concludes our arc on learning from 3D correspondence and point cloud registration in RGB-D video. Our goal was not to beat supervised methods or even to argue that one should not use supervised learning when labels are available. Rather, we aim to demonstrate that supervision is not needed as the raw sensory stream available through RGB-D video already provides a sufficiently strong learning signal. While standard 3D reconstruction systems like COLMAP and BundleFusion provide us with good reconstructions, learned approaches trained with their outputs are starting to rival their accuracy. Furthermore, those systems share the same inability to handle dynamic scenes that our current approaches suffer from. We argue that the way forward is to think about the problem more holistically and go back to the source of the data, which is video. By proposing self-supervised pipelines that learn directly from videos, we take a step towards enabling the development of approaches that go beyond conventional setups and scale to more uncurated data, allowing us to both achieve better 3D reconstruction and tackle difficult challenges like dynamic scene reconstruction.

Moving forward, we argue that the next advancements will come from learning from dynamic scenes. In our work, we assume that the motion in the scene can be explained by a global rigid body transformation. There has also been exciting work that learns from videos of dynamic scenes through through tracking or optical flow estimation [121, 302]. However, such models learn from motion in the 2D plane and do not handle the underlying 3D world that they represent. As a result, they often struggle with occlusion and dis-occlusion, similar as shown by Shrivastava and Owens [256]. We argue that the next advancement will come from models that can handle the full 3D dynamics of the world. There is already some exciting progress on this front with dynamic scene representations that capture both static and dynamic components [29, 181]. Through integrating such representations with self-supervised learning, one can expect models that learn better visual representations compared to ones that rely on rigid body transforms or optic flow.

Chapter 5

Learning from Conceptual Consistency

In the previous chapters, we used the traditional notion of 3D correspondence: *i.e.*, image patches that depict the same 3D point. However, the concept of correspondence also arose in the study of multisensory perception and representation [265] where it has a more general meaning: two sensory inputs that depict the same "thing." We experience cross-modal correspondence all the time; *e.g.*, seeing a cat, hearing its purr, and feeling its fur.

This general notion of correspondence is more relevant to representation learning. Perhaps the most popular instantiation is CLIP [219], which learns by associating image and text embeddings of a captioned image. In this formulation, there is a single multimodal representation where everything is jointly embedded; *e.g.*, the visual embedding of an image and the language embedding of its caption are the same. However, this approach suffers from a modality gap, as shown by Liang et al. [168]. An intuitive explanation of the modality gap is that visual and linguistic modalities, while sharing some commonalities, are different. We expect this gap to be exacerbated as we incorporate more modalities.

We argue that a different approach is to consider cross-modal correspondence between unimodal representations. This is motivated by the *representation as second-order isomorphism* theory of Shepard and Chipman [252]. Rather than require the representation of multiple modalities to be the same, we can learn unimodal representations such that if two instances have similar representations in one modality, it should also be similar in other modalities. This deviates from current learning approaches and offers a new learning paradigm from multimodal learning.

In this chapter, we explore this idea in the space of vision and language. Rather than embed images and captions in the same space, we use language models to find image pairs whose captions have similar language embeddings. We use these language-sampled image pairs for contrastive learning. We call this *language-guided visual learning* and show that it learns more generalizable representations than both self-supervised and language-supervised approaches. This presents a first step in this new paradigm, which we hope to expand to other modalities. This chapter is based on work previously published in [78].

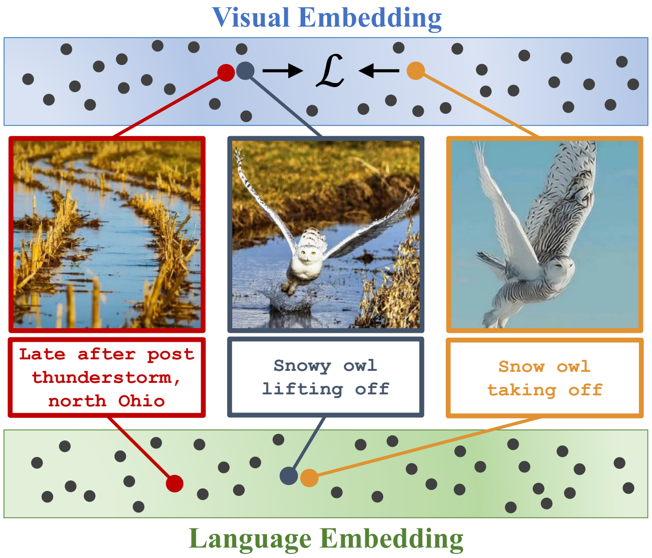

Visual Embedding
$\rightarrow \mathcal{L} \leftarrow$
Late after post thunderstorm, north Ohio
Snowy owl lifting off
Snow owl taking off
Language Embedding

Figure 5.1: Language allows us to find conceptually similar image pairs even if they are visually dissimilar. We use those pairs to learn generalizable visual features.

5.1 Introduction

Consider the images in Fig. 5.1, is the center image more similar to its left or right neighbor? Despite the difference in background and lighting, it is clear that the right pair captures the same concept: a flying snow owl. Nevertheless, a self-supervised image model will judge the left pair as more similar. Human perception and language abstract away appearance differences to capture conceptual similarity rather than just visual similarity. Ideally, we could learn visual features that capture conceptual similarity, allowing them to effectively generalize to other visual tasks. In this work, we show how language can be a proxy for conceptual similarity, allowing us to sample better pairs for contrastive learning and train more generalizable visual models.

Image-only contrastive learning uses visual similarity as a proxy for conceptual similarity. This is based on the observation that discriminative approaches can discover inter-class similarity–*e.g.*, cheetahs are similar to lions– without explicit annotation [315]. The core idea is to then train a discriminative model where each instance is treated as a separate class, and the model is trained to map augmented versions of the same image to similar features [40–43, 315]. While successful, instance discrimination ignores the similarity between different instances as it assumes all other images are unrelated. Later work focused on inter-image relationships by estimating clusters [9, 31, 32] or finding nearest neighbors [71]. However, those relationships are estimated using visual embeddings, resulting in visually, rather than conceptually, similar pairs.

Language similarity is a strong proxy for semantic relationships. Consider the example in Fig. 5.1; images that depict the same concept are often described in very similar ways. Radford et al. [219] propose language-image contrastive learning by mapping images and captions to the same space and achieve impressive generalization capabilities. However, it is unclear whether forcing models to map onto a joint space is optimal for visual learning. Although linguistic and visual similarity might align for very similar instances, it is not clear that all distances in one space should map exactly to the other. Instead of learning a joining embedding for vision and language, we argue that it is better to use linguistic similarity to guide visual learning.

To this end, we propose language-guided contrastive learning: a simple adaptation to contrastive learning that uses language models to find conceptually similar image pairs for visual learning. Our approach is motivated by the observation that language models that were never trained on visual data can sample caption pairs that belong to conceptually similar images, as seen in Fig. 5.2. Furthermore, the sampled pairs exhibit desirable variations in pose, lightning, and context, which are very different from hand-crafted augmentations which can be ill-suited to downstream tasks [318] or too focused on background textures [247]. We use the sampled pairs instead of image augmentations within standard self-supervised visual learning approaches such as SimCLR [40], SimSiam [42], and SLIP [198]. Our approach departs from image-only contrastive learning by relying on conceptually similar image pairs rather than visually similar augmentations or cluster assignments. We also depart from image-caption pre-training by allowing the model to be guided by language similarity rather than learning a joint embedding space.

We conduct a series of controlled experiments to analyze our approach and compare it to commonly used representation learning paradigms on generalization to downstream classification tasks. In controlled settings, our approach outperforms all baselines on linear probe and few-shot classification on a range of downstream classification datasets. Our analysis suggests that while learning multi-modal joint embeddings can result in good representations, it is better to use one modality to guide the training of the other. Furthermore, while language guidance can boost performance, we found that the exact choice of sampling strategy or language model appears to be less impactful.

5.2 Related Work

Visual Representation Learning aims to learn visual embedding spaces that capture semantics, with a typical focus on learning from scalable data sources. Broadly speaking, there are two general approaches: generative and discriminative. Generative approaches hypothesize that a model that can capture the image distribution will learn semantically relevant features [68, 87, 103, 204, 298, 344]. In contrast, discriminative approaches posit that differentiating between images will give rise to better features. This idea can be traced to early work on metric learning [48] and dimensionality reduction [98] and is clearly seen for supervised classification models [250]. More recently, Wu et al. [315] proposed treating each image as a separate class and using augmented images as class instances to relieve the need for human annotation. This was followed by papers that simplified this approach [40, 41, 43, 102] and proposed non-contrastive variants [42, 97]. While those approaches have been successful, the utility of augmentation-based self-supervised learning has been questioned [201, 318] with follow-up work proposing the use of objectness [196, 212] and saliency [247] to alleviate some of those concerns. While we share the motivation for visual representation learning, we also question the reliance on image augmentation and propose using language sampling to learn better invariances.

Language-supervised vision pre-training aims to learn visual representations from language data. Early work of Li et al. [161] trained n-gram models using YFCC [278] images and user-tag metadata. While some work learns joint embeddings for vision and language tasks like visual question answering [3, 95, 117, 355], visual reasoning [133, 271, 336], and retrieval [209, 333], we are interested in methods that use language to learn better visual features [64, 64, 219, 240, 270]. Early work learned features by generating captions [64, 240], but contrastive approaches gained more popularity due to their relative simplicity and generalization capabilities [123, 219]. Follow-up work extends the contrastive formulation to learn denser features [322, 326] or uses additional self-supervised losses to improve performance and data efficiency [55, 155, 166, 198]. While we share the motivation of using language for visual learning, we depart from the typical discriminative approach and use a pre-trained language model to guide the learning by providing conceptually similar visual pairs.

Leveraging structure in the data. This is commonly done in dense feature learning, where optical flow [99, 121, 248, 302] or 3D transformations [77, 111, 249, 272, 314] provide natural associations between image patches. For images, prior approaches used class names [134, 232], class hierarchies [162, 324], meta data [91, 128, 161] or clustering [9, 31, 32, 282, 351] to improve learning and inference. Within contrastive learning, clustering has been a popular choice for leveraging dataset structure. The intuition is that natural clusters emerge in feature spaces, which can provide an additional training signal or useful pseudo-labels. While such approaches work well on curated datasets (*e.g.*, ImageNet) where the label set provides an estimate of the number of clusters, it struggles with imbalanced and uncurated data [10]. Others sample the nearest neighbor as a feature-driven within-domain augmentation [71, 166]. While these approaches differ in how they extract inter-instance relationships, they all use within-domain feature similarity to sample positive pairs or clusters and, hence, do not leverage the rich cross-modal relationships. Closest to our work is Han et al. [99], who propose a co-training [23] scheme for jointly learning image and optic flow representations. We share their motivation of using similarity in one space to learn in another but apply it instead to vision and language. Furthermore, instead of relying on co-training on the same dataset, we extract distances from a text-only language model, allowing us to better leverage unaligned data.

5.3 Language-Guided Self-Supervised Learning

The goal of this work is to learn visual representations that can be generalized to other datasets. We extend visual contrastive learning beyond hand-crafted augmentations and visually sampled clusters to learn from conceptually similar images. Through learning to associate images that depict the same *visual concept*, models can learn visual invariances that more closely capture human semantics. To achieve this, we propose sampling images that have similar captions using a pre-trained sentence encoder [225]. This work does not propose a new model or loss but rather a novel way of sampling image views that is applicable to a variety of approaches and losses.

5.3.1 Learning from Conceptual Similarity

Instance discrimination has been the dominant task for visual representation learning. Its core intuition is that visual similarity is a good proxy for semantic similarity. At a high level, these approaches generate positive *view* pairs by randomly augmenting the input image and maximizing their embedding similarity, with or without negative views. While there has been a large number of contrastive learning approaches, view pair generation has largely remained the same. Other methods use visual feature similarity to sample learned prototypes [9, 31, 32] or previously seen instances [71] for contrastive learning. While these approaches extend beyond instances and consider relations in the dataset, they still rely on visual similarity to generate their contrastive pairs. This limits what visual invariances they can learn [318].

We propose training models to identify the same visual concept instead of the same instance. Our key observation is simple: Images that have similar captions often depict similar concepts regardless of the actual similarity in appearance. This can be clearly seen

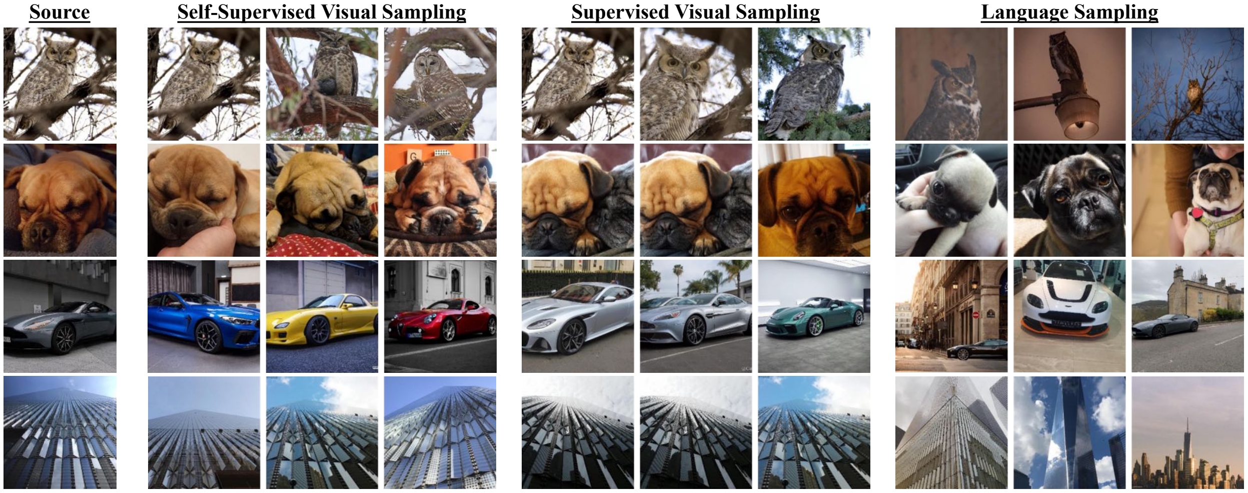


Figure 5.2: **Language sampling provide semantically-similar and visually-diverse image pairs.** We sample the top 3 nearest neighbors using a self-supervised visual model [41], an ImageNet supervised model [101, 309], and a self-supervised language model [225, 262]. While visual sampling retrieves pairs with high visual similarity, language sampling retrieves both semantically relevant and visually diverse images. We argue that the combination of semantic consistency and visual diversity allows our model to learn more generalizable features.

in Fig. 5.2. Nearest neighbors in visual feature space depict objects in similar scenes and poses, with self-supervised models showing some color invariances due to color augmentation. Conversely, similarly captioned images depict objects in different colors, poses, and contexts. This makes language-sampled images an excellent source for visual learning as they implicitly capture human-like visual invariances.

5.3.2 Sampling Image Pairs using Language

Given a captioned image dataset, we want to sample image pairs that have very similar captions. While caption similarity may be a good proxy for conceptual similarity, measuring caption similarity is a challenge on its own. Traditional metrics such as BLEU [208] and CIDER [295] rely on n-gram overlap, which can be too sensitive to phrasing and sentence structure. This makes them ill-suited for our needs. Other metrics such as SPICE [2] account for such variety by comparing parse trees. However, they still can not account for different word choices. Inspired by advances in language models as well as approaches like BERTScore [345] and CLIPScore [109], we use a pre-trained language model to compute caption similarity.

Sentence encoders are trained to extract sentence-level features [139, 175, 225]. We

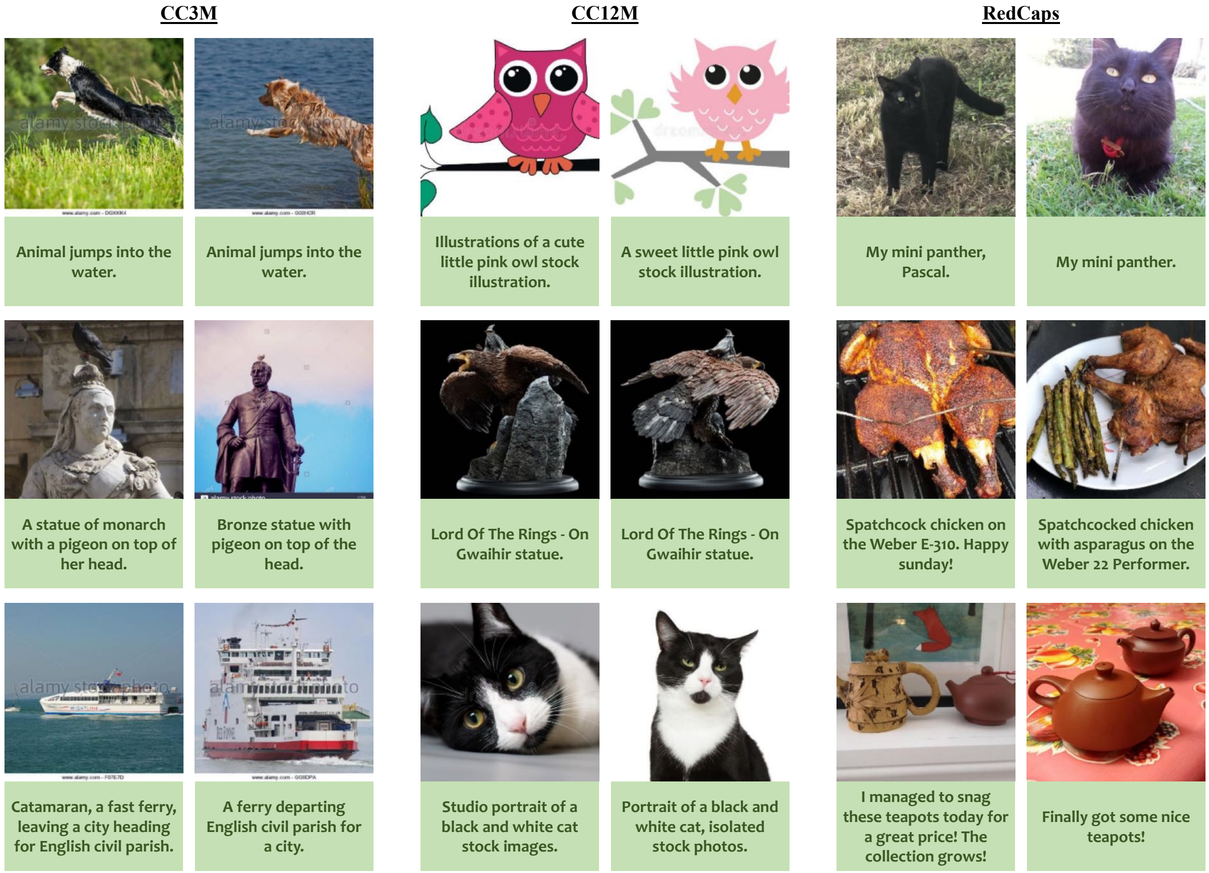

CC3M
CC12M
RedCaps
Animal jumps into the water.
Animal jumps into the water.
Illustrations of a cute little pink owl stock illustration.
A sweet little pink owl stock illustration.
My mini panther, Pascal.
My mini panther.
A statue of monarch with a pigeon on top of her head.
Bronze statue with pigeon on top of the head.
Lord Of The Rings - On Gwaihir statue.
Lord Of The Rings - On Gwaihir statue.
Spatchcock chicken on the Weber E-310. Happy sunday!
Spatchcocked chicken with asparagus on the Weber 22 Performer.
Catamaran, a fast ferry, leaving a city heading for English civil parish.
A ferry departing English civil parish for a city.
Studio portrait of a black and white cat stock images.
Portrait of a black and white cat, isolated stock photos.
I managed to snag these teapots today for a great price! The collection grows!
Finally got some nice teapots!

Figure 5.3: **Language-Guided image pairs for the different pre-training datasets.**

use SBERT, which fine-tunes a pre-trained language model to allow it to better capture semantic similarity using feature cosine distance [225]. SBERT is trained in two stages: first, a language backbone is trained using a standard self-supervised task such as masked [67, 193] or permuted [262] language modeling; second, the language modeled is fine-tuned via contrastive learning. Contrastively fine-tuning the model simplifies downstream usage as it allows features to be compared directly using cosine similarity. We use an SBERT [225] model with an MPNet [262] backbone. However, we find that our formulation is not sensitive to the choice of language encoder, as shown in Tab. 5.8.

We present language-sampled nearest neighbors for different datasets in Fig. 5.3. CC3M and CC12M were collected using alt-text data and were heavily post-processed, resulting in these generic descriptions. Meanwhile, RedCaps contains captions that were written by someone as they uploaded their image to Reddit. As a result, they tend to be more descriptive, referring to pet names or specific product brands or models, as well as occasionally noisy. Empirically, we find that RedCaps results in better performance.

One concern might be that the nearest neighbor search on a large dataset will be prohibitive. We find that modern similarity search libraries [126] significantly speed up the sampling and scale up to very large datasets. For example, nearest neighbor sampling runs in under 3 hours for RedCaps (12 million instances) on 4 GPUs, with 43 minutes spent on feature extraction and 117 minutes on nearest neighbor search. Furthermore, we find that we can reduce the complexity of the sampling by only searching within subsets of the data, as shown in Tab. 5.9.

5.3.3 Language-Guided Visual Learning

Our approach is applicable to several representation learning methods as it only affects the training views. We focus on contrastive learning since its fairly minimal setting allows us to analyze the impact of language guidance with minimal confounding factors. We train SimCLR with the language-sampled pairs and refer to it as LGSimCLR. We also evaluate the impact of language guidance on SimSiam [42] and SLIP [198], and find that they can also benefit from language guidance. We only use random cropping for image augmentations since language-sampled pairs are naturally augmented versions of each other. For LGSLIP, we match their setup by applying the CLIP loss only between the source's image and caption, ignoring an additional loss between the nearest neighbor image and its caption.

5.4 Experiments

Our experiments evaluate the efficacy of learning visual features from conceptually similar images. We hypothesize that a model trained with language guidance will learn useful visual invariances and better generalize to downstream tasks. We are interested in answering these questions: Does language guidance improve generalization over other pre-training approaches? Does language guidance generalize to other datasets and pre-training approaches? How can language be used for visual pre-training?

5.4.1 Experimental Setup

We formulate our experimental setup to compare the efficacy of different learning signals. We train models with language-guided sampling and compare them with image-only self-supervised models and image-text contrastive models. We are interested in conducting controlled experiments for a fair comparison.

Recent work in self-supervised learning has demonstrated the impressive impact of scaling on existing approaches [40, 219, 338]. While such work has shown impressive performance, it has complicated the evaluation as different models are trained on different pretext tasks on different datasets using varying amounts of compute and training recipes. Furthermore, replication is difficult, if not impossible, due to the unavailability of training data or prohibitive compute requirements. Fortunately, several papers report results that indicate that performance patterns often hold at smaller scales [32, 40, 55, 198, 219]. Hence, we conduct our experiments at a scale that allows us to perform a comprehensive evaluation and allows its replication by others.

We conduct our experiments with a standard backbone [101] on publicly available datasets [37, 65, 251]. To account for variation in training recipes, we retrain all methods from scratch using the same training recipe. We scale down experiments to a level that permits fair comparisons and replication. We also provide system-level comparisons in Tab. 5.4 and scaling results in Sec. 5.5.3. We also released all code and pre-trained models.[1]

Training details: We use a ResNet-50 backbone and train all models using the AdamW optimizer [178] with a learning rate of 10^{-3} and a weight decay of 10^{-2}. We use a cosine learning scheduler [177] with 5000 warm-up steps. Models are trained using a batch size of 512 for 250k steps; this corresponds to 10.5 epochs on RedCaps. We use a constant number of steps to make meaningful comparisons between models trained on different datasets.

Evaluation setup: We evaluate all approaches using linear probe and few-shot classification on 15 classification datasets shown in Appendix B.2 inspired by [144, 219]. We use the linear probe evaluation proposed by [144] and learn a single linear layer using logistic regression. We sweep over a large range of cost values and choose the one with the highest validation performance. We then retrain a classifier on both the train and validation split and report test performance. We also evaluate all approaches on fewshot classification to understand their generalization ability. Rather than train a probe for few-shot classification, we simply use a weighted kNN classifier on the frozen support features inspired by the findings [303]. Please see Appendix B.1 for more details.

Baselines: While there have been a large number of proposed approaches to visual representation learning, one can identify several key directions that differ in the training

[1] https://github.com/mbanani/lgssl

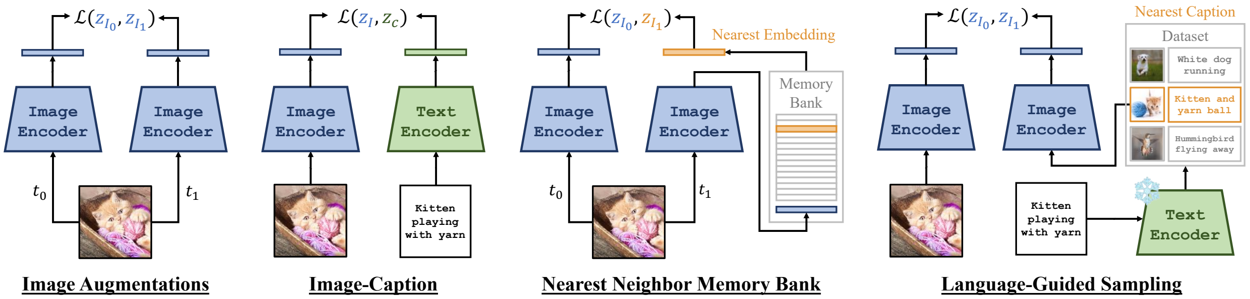


Figure 5.4: **Contrasting Contrastive Formulations.** While Image-only and Image-Text contrastive learning directly extract views from the instance, Nearest Neighbor methods rely on a memory bank of previously extracted features for training. In contrast, our approach samples nearest neighbors in caption space using a pretrained language model and use the associated image for visual representation learning.

signal and pre-text task. We focus our comparison on representative approaches of each key direction as this allows us to explore the impact of learning signals. We overview our baselines here and provide more details in Appendix B.3.

The first set of approaches rely on contrastive learning. Contrastive approaches operate over two sets of paired source and target feature embeddings: $\boldsymbol{z}^s$ and $\boldsymbol{z}^t$. The goal is to maximize the similarity between the paired embeddings and minimize it with respect to all other embeddings. Given a batch size N and embedding dimension F, $\boldsymbol{z}^s, \boldsymbol{z}^t \in \mathbb{R}^{N \times F}$. The contrastive loss [261] is:

$$\mathcal{L}(z^s, z^t) = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i^s, \boldsymbol{z}_i^t)/\tau)}{\sum_{k=1}^N \exp(\text{sim}(\boldsymbol{z}_i^s, \boldsymbol{z}_k^t)/\tau)}\ , \tag{5.1}$$

where τ is a scaling parameter and $\text{sim}(\cdot,\cdot)$ is cosine similarity. Contrastive approaches primarily differ in the source of the embeddings used within this loss.

Image-based Contrastive Learning contrasts features extracted from two randomly augmented *views* of the same image to perform instance discrimination [315]. We use SimCLR [40] as a representative approach due to its simplicity and strong performance.

Image-Text Contrastive Learning learns by contrasting features extracted from images and their captions. Unlike image-only approaches, this approach can inject some semantics into the learning process. We use the CLIP [219] formulation due to its simplicity. SLIP extends this formulation by adding an image-only contrastive loss.

Table 5.1: **Linear Probe Evaluations.** We train ResNet-50 models on RedCaps and report performance of logistic regression using frozen features on 15 downstream tasks. Models are split based on whether they use image captions. LGSimCLR outperforms all previous approaches with strong performance gains for fine-grained classification datasets.

Model	**Food-101**	**CIFAR-10**	**CIFAR-100**	**CUB**	**SUN397**	**Cars**	**Aircraft**	**DTD**	**Pets**	**Caltech-101**	**Flowers**	**STL-10**	**EuroSAT**	**RESISC45**	**PCAM**	**Mean**
SwAV	63.6	81.3	57.5	21.6	47.5	22.9	35.4	68.1	61.1	70.5	78.0	87.7	94.3	79.9	84.3	63.6
SimSiam	64.1	79.9	56.1	28.2	48.3	29.5	41.2	66.2	69.1	73.6	83.6	85.7	94.4	82.1	83.3	65.7
Visual NNCLR	65.4	82.8	60.2	26.6	50.0	26.6	40.9	68.0	65.2	75.4	83.5	88.5	95.3	82.2	83.8	66.3
SimCLR	69.0	82.9	61.6	30.6	52.6	33.7	43.7	69.8	70.5	74.1	86.9	88.0	95.4	84.6	84.4	68.5
Language NNCLR	81.2	83.1	61.9	48.6	56.5	45.1	37.2	68.8	78.1	82.0	90.2	93.4	92.5	81.1	80.7	72.0
CLIP	80.9	84.7	62.7	50.4	57.4	45.8	36.7	67.6	79.8	84.0	91.0	93.5	93.9	82.2	82.6	72.9
SLIP	77.7	87.2	67.0	42.4	58.1	48.7	45.2	72.3	79.5	82.7	92.1	92.7	95.6	85.5	83.4	74.0
LGSimCLR (Ours)	83.2	87.8	69.0	59.3	60.3	62.3	53.4	71.2	81.8	89.4	95.9	94.0	95.6	88.0	81.1	78.2

Nearest Neighbor Contrastive Learning contrast image embeddings with retrieved embeddings from a memory bank. The target features are used to retrieve the nearest neighbor embedding from a memory bank of previous batches; see Fig. 5.4. We use the NNCLR [71] as Visual NNCLR. Li et al. [166] applied this idea to image-language pretraining; we adopt their approach by using the CLIP and NNS losses as Language NNCLR.

Image-based Non-Contrastive Learning deviates from the typical contrastive setup by learning without negative samples [42, 97]. We use SimSiam as a representative approach due to its simplicity and strong performance.

Cluster-based methods extracts prototypes through clustering and learns by predicting cluster assignments [9, 31, 32]. Caron et al. [32] show that cluster-based methods perform similarly when compared fairly. We use SwAV without the multi-crop strategy as it is applicable to other methods. We compare with the full SwAV model in Tab. 5.4.

5.4.2 Results

We train all models with a ResNet-50 backbone on RedCaps and report results in Tabs. 5.1 and 5.2. Our model outperforms all baselines with a significant margin for both evaluations. We discuss the results below through a series of questions:

Does language-guided sampling provide better training pairs than image augmentations? By using language sampling, instead of image augmentations, LGSimCLR learns stronger invariances and outperforms SimCLR. We find that the largest gains arise in fine-

Table 5.2: **Few-Shot Evaluations.** We train ResNet-50 models on RedCaps and report 5-way, 5-shot fewshot classification performance. We observe that language results in huge performance gains as shown by the performance of CLIP and LGSimCLR. Furthermore, the use of any augmentations hurts performance as seen by SLIP's drop in performance.

Model	**Food-101**	**CIFAR-10**	**CIFAR-100**	**CUB**	**SUN397**	**Cars**	**Aircraft**	**DTD**	**Pets**	**Caltech-101**	**Flowers**	**STL-10**	**EuroSAT**	**RESISC45**	**Mean**
SwAV	64.5	54.0	61.8	45.8	84.9	36.5	34.1	74.8	66.5	78.1	75.5	72.6	80.4	72.9	64.5
SimSiam	63.9	49.9	57.2	49.5	84.5	39.3	37.9	75.7	67.8	79.7	81.5	69.6	80.6	79.4	65.5
Visual NNCLR	65.6	54.1	61.7	45.8	85.3	37.9	34.9	75.2	67.3	81.1	75.4	74.3	83.6	76.7	65.6
SimCLR	66.9	45.7	51.0	51.5	87.1	44.0	38.4	77.6	70.1	80.0	86.9	69.6	83.5	81.3	66.7
Language NNCLR	89.3	65.3	73.4	78.6	90.8	68.4	40.4	75.2	78.8	90.9	94.3	89.6	75.2	71.9	77.3
CLIP	88.9	64.6	73.1	78.3	90.9	69.7	40.7	75.7	77.5	91.6	94.7	89.8	75.3	74.8	77.5
SLIP	81.5	63.5	70.8	63.1	91.3	62.9	42.1	79.6	76.4	88.4	92.2	83.4	82.7	80.8	75.6
LGSimCLR (Ours)	90.3	66.3	75.5	83.1	92.7	77.6	50.6	81.1	84.1	95.4	97.6	86.5	85.0	89.0	82.5

grained datasets: Cars, CUB, and Food101. The performance gains can be explained when considering the critique of Xiao et al. [318]: the training augmentations dictate the invariances learned by the model. Consider the third row of Fig. 5.2, while language sampling yields three Aston Martin cars in different perspectives, locations, and appearances, visual nearest neighbors are sports cars in similar poses and different colors; closely resembling the flip and color augmentations used for training. Similarly, while visual features nearest neighbors result in owls of different species in similar poses, language sampling retrieves three great horned owls in different contexts. These trends are further amplified when features are used directly for fewshot classification. Language sampling captures relationships that go beyond visual similarity and more closely resemble semantic similarity.

Can we just sample nearest neighbors from previous batches? LGSimCLR also greatly outperforms NNCLR. NNCLR uses the nearest feature embedding from a memory bank. However, the quality of their retrieved samples is limited by the size of the queue. Furthermore, those features were only trained on image augmentations, limiting the invariances they capture. To further capture this point, we visualize the nearest neighbor embedding retrieved by NNCLR for different memory bank sizes in Fig. 5.5. We find that the retrieval quality varies greatly and is poor for smaller queue sizes but that even a larger queue size isn't guaranteed to retrieve a good positive pair. Interestingly, we find that NNCLR also under-performs SimCLR on RedCaps, despite performing better on ImageNet. We posit that ImageNet's curated distribution explains this. With only 1000 classes, a queue of 16k will most probably contain some examples from each class, resulting in both visually and conceptually similar retrievals.

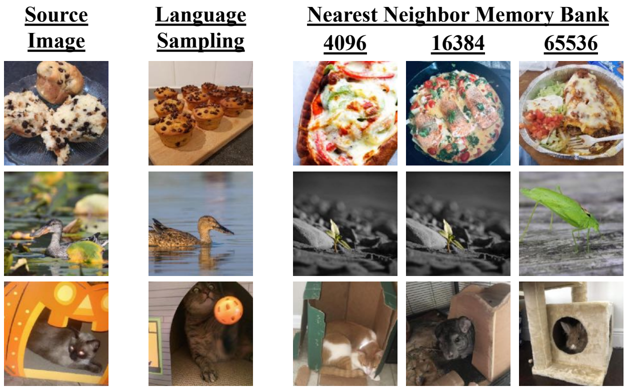


Figure 5.5: **Nearest Neighbor methods are limited by the memory bank size.** Even with a large memory bank, the nearest embedding can still be unrelated to the source image while language sampling provides us with conceptually similar pairs.

Can cluster-based approaches learn better features? Similar to nearest-neighbor sampling, clustering is performed based on visual similarity. Furthermore, it is based on an assumed number of clusters in the dataset for training. Although this can be determined for ImageNet due to its known class structure, the number of clusters in an arbitrary uncurated dataset is unknown. This results in a large performance drop, as seen in Tab. 5.1 and Tab. 5.2. On the other hand, sampling related pairs assumes no global structure within the data, and hence is able to better capture inter-instance similarity. This results in nearest-neighbor sampling outperforming clustering and both being outperformed by contrastive learning and language-guided contrastive learning.

Should we use language for guidance or supervision? LGSimCLR outperforms both CLIP and SLIP. We consider two possible explanations: (a) SBERT extracts better language embeddings than CLIP can learn from the data, or (b) language-guided contrastive learning is a better training objective than image-text contrastive learning. To evaluate this, we compare four models in Tab. 5.3. The first two models use CLIP's training objective, but one trains a language encoder from scratch, while the other uses a frozen SBERT model and only learns the visual encoder and projection layers. The second two models use LGSimCLR's training objective but sample pairs using a pre-trained language-only SBERT or the language encoder from a CLIP model trained on RedCaps. While CLIP does not benefit from an SBERT backbone, LGSimCLR benefits from sampling using a language encoder trained on the same dataset. Our results suggest that joint embedding results in worse features than language-guided self-supervision. This is likely the result of the modality gap identified by Liang et al. [168].

Table 5.3: **Language guidance outperforms language supervision.** We compare two language-based objectives: using captions to sample image pairs or using captions in an image-text contrastive loss. We find language-guidance yields better visual features regardless of the choice of text encoder.

Objective	**Text Encoder**	**Linear Probe**	**Few-Shot**
Image-Text	CLIP (RedCaps)	72.9	77.5
	Frozen SBERT	71.8	77.1
Image-Image	CLIP (RedCaps)	78.3	82.4
	Frozen SBERT	78.2	82.5

System-level comparisons: We compare LGSimCLR with publicly-available checkpoints of prior approaches; see Appendix B.3 for details. We emphasize that while previous experiments were done in a controlled setup–same batch size, training data, optimizer– the system level comparisons are trained on different datasets using different training recipes and utilizing several training enhancements to further boost performance; *e.g.*, large batch sizes, longer training, multi-crop augmentation. Furthermore, it has been shown that models trained on ImageNet implicitly benefit from its curated nature [10, 198]. Nevertheless, our approach still outperforms prior self-supervised approaches. We fall short of CLIP's ResNet-50 due to its training scale; $64\times$ larger batch, $32\times$ larger dataset, and 75×. We also observe that ImageNet-supervised ResNet-50 achieves better fewshot performance. Examining the performance breakdown, see Tab. B.3, we find the improvement mainly comes from CIFAR10, CIFAR100, and Pets. We posit that this can be explained by ImageNet's class structure; mostly pets with a large overlap with CIFAR's classes.

Table 5.4: **System Level Comparisons.** We outperform prior self-supervised approaches despite them benefiting from ImageNet's curation for training and using larger batch sizes. CLIP outperforms us due to the scale of their training.

	Batch	# Image Updates	Dataset	Linear	Fewshot
Supervised [309]	1024	1.3×10^8	ImageNet	78.0	85.7
SimSiam [42]	512	1.3×10^8	ImageNet	72.9	78.7
SimCLR [41]	4096	1.0×10^9	ImageNet	75.4	77.4
MoCo [44]	4096	1.3×10^8	ImageNet	77.7	80.1
SwAV [32]	4096	1.3×10^8	ImageNet	78.2	78.5
CLIP [219]	32768	1.0×10^{10}	CLIP	81.8	87.8
LGSimCLR	512	1.3×10^8	RedCaps	78.2	82.5

5.5 Analysis

We now analyze language-guided contrastive learning by evaluating the impact of pre-training data, the choice of embedding space, and the pre-text task. Through understanding the impact of those choices, we can better understand what the model is learning.

5.5.1 Approach generality

We extend language guidance to two other self-supervised representation learning approaches: SimSiam and SLIP. Using language-sampled pairs consistently improves the performance of models compared to image-augmented pairs as shown in Tab. 5.5. Interestingly, we find that SLIP's performance is still improved by using language-sampled pairs despite the fact that it already uses a language-based loss. This further supports our claim that language guidance might be superior to language supervision.

Table 5.5: **Approach Generality.** Using our language-sampled image pairs can provide strong performance boosts over image augmentations for several self-supervised learning formulations.

	Image Augmentations		**Language Sampling**	
	Linear	**Fewshot**	**Linear**	**Fewshot**
SimSiam	65.7	65.5	71.2	75.7
SimCLR	68.5	66.7	78.2	82.5
SLIP	74.0	75.6	78.8	82.8

5.5.2 Training Data

We train our model on four datasets: CC3M [251], CC12M [37], RedCaps-2020, and RedCaps [65]. In Tab. 5.6, we observe that larger datasets result in stronger performance, indicating that our approach could scale well with even larger datasets. Furthermore, we observe that RedCaps results in better performance than Conceptual Captions. This may be attributed to the higher quality of captions in RedCaps – while the alt-text captions CC3M and CC12M can be short and contain image metadata, RedCaps captions are diverse, longer, and more descriptive. This allows our model to sample more interesting visual pairs that capture more visual diversity. Results in Sec. 5.6 further support this.

Table 5.6: **Language-guided learning scales well with datasets size and quality.** We observe that training on larger datasets results in consistent performance improvements. Furthermore, training on datasets such as RedCaps which have longer and more natural captions results in better performance than training on CC3M or CC12M whose captions are more generic and heavily processed.

	Size	Linear	Fewshot
CC3M	2.7M	71.5	76.3
CC12M	10.9M	76.8	81.9
RedCaps 2020	3.2M	73.8	78.8
RedCaps	12.0M	78.2	82.5

5.5.3 Batch Size Scaling

We explore the scaling performance of our data with respect to batch size. Prior work has shown that contrastive methods can benefit larger batch sizes [40, 41, 219]. While we conduct our comparisons with a batch size of 512 to allow us to perform comprehensive experiments and evaluations, we show that our models can also scale to larger batch sizes. Our results, shown in Tab. 5.7, show that our performance scales with batch size and that we maintain our performance gains over SimCLR for larger batch sizes. Furthermore, we show in Tab. 5.6 that our model benefits from larger datasets. We note that the improvement is not limited to finding the nearest neighbor in more images, but rather than having more images to train on improves performance as can be seen in the improvement of the RedCaps model sampled on Years compared to the RedCaps-2020 models. While both models sample the nearest neighbors from each year, the RedCaps trained model has $4\times$ more data and shows clear performance gains over the RedCaps-2020 trained model.

Table 5.7: **Batch Size Scaling.** Our model scales well with larger batch sizes similar to other contrastive approaches.

	SimCLR		LGSimCLR	
Batch Size	**Linear**	**Fewshot**	**Linear**	**Fewshot**
256	67.5	67.2	77.7	82.3
512	68.5	66.7	78.2	82.5
1024	69.3	68.4	78.6	82.6
2048	69.8	68.6	79.1	83.1

5.5.4 Sampling space

The idea of using offline nearest neighbor sampling does not require a specific language model or even language as a modality. We explore other choices for embedding space: four sentence encoders and two image models. In our experiments, we use SBERT's MPNet model [225, 262]; the highest performing SBERT model for sentence similarity. We compare it to two other sentence transformers: a smaller SBERT model, MiniLM [301], and the language encoder from CLIP [219]. We also compared against a bag-of-words (BoW) sentence encoder that uses FastText [24] embeddings. Results are presented in Tab. 5.8.

While we expected that using CLIP for sampling would improve performance due to its multimodal training, we were surprised that MiniLM also improved performance despite its lower performance on language tasks. We find that pairs obtained using a BoW model result in weaker performance which might hint at the importance of contextual sentence embeddings. Nevertheless, the BoW-sampled pairs still result in higher performance than all previous approaches on RedCaps. This indicates that language guidance still provides a useful performance gain even with a naive choice of language embedding and that choosing a more sophisticated sentence embedding can further improve performance.

We also consider two visual models: ImageNet-supervised ResNet-50 [309] and ImageNet-trained SimCLR [41]. Our results indicate that using a visual model for sampling is only beneficial if the visual model captures some semantic relations; *e.g.*, through supervised training. Using a self-supervised language model results in a strong drop in performance relative to the other sampling spaces. Nevertheless, it still allows the model to achieve better performance than using a self-supervised visual approach on the same data. This indicates that while language is a better modality to use, "sample-guided" contrastive learning can still achieve a stronger performance than only using self-supervised learning.

Table 5.8: **Impact of Sampling Space.** While language sampling consistently results in good pairs for training, visual sampling only helps if it has access to semantics through labels through supervised training or language through vision and language pre-training.

Sampling Space	Linear	Fewshot
SBERT (MPNet)	78.2	82.5
SBERT (MiniLM)	78.6	83.3
CLIP Language (ViT-B/32)	78.3	83.1
FastText BoW	76.1	80.9
ImageNet-supervised	78.3	81.8
SimCLR (ImageNet)	73.1	74.6

5.5.5 Sampling Scope

One issue with our approach is the computational cost of nearest neighbor search. While fast similarity search libraries have made this more feasible, searching over billions or trillions of images could be very expensive. As a result, we explore how our model's performance changes as we adapt the sampling to be restricted to subparts of the dataset. We use RedCaps since it provides us with several ways of splitting the data. Subreddits offer a natural separation of datasets that is domain-specific. This allows us to explore how our model would perform is restricted to search within more relevant instances. RedCaps was also collected over several years. Splitting by years allows us to explore the impact of *random* splits of the data as years would only have minimal domain shift. We also explore Subreddit-Year, which only samples nearest neighbors from the same subreddit that were posted in the same year. This sampling combines both size and domain-specificity.

Our results, presented in Tab. 5.9, show that domain-specific sampling actually improves performance. Meanwhile, more random sampling minimally degrades performance. Finally, restricting the scope of the nearest neighbor sampling is not the same as sub-sampling the data. This is shown by the higher performance of RedCaps with Year sampling compared to RedCaps-2020. Those results indicate two opportunities: First, our approach can scale to very large datasets by only performing nearest neighbor searches within subsets of the data. This is especially beneficial in some domains such as federated learning. Second, identifying further domain structure within the dataset can result in improved performance by allowing the model to sample nearest neighbor images within the same domain.

Table 5.9: **Impact of Restricted Sampling.** LGSimCLR can still learn good features if it is restricted to only sampling from a subset of the datasets. Domain-specific sampling (*e.g.*, subreddits) improves performance, while domain agnostic partitions (*e.g.*, Year or Subreddit-Year) minimally degrades performance.

Dataset	**Sampling Scope**	**Number of Partitions**	**Linear**	**FewShot**
RedCaps 2020	All	1	73.8	78.8
RedCaps	All	1	78.2	82.5
	Year	4	77.2	80.2
	Subreddit	350	79.0	80.5
	Subreddit-Year	1391	77.6	78.2

5.6 Qualitative Analysis

We now analyze the types of image pairs that we get via sampling. We identify four patterns when comparing sampling using language, supervised vision models, and self-supervised vision models. We present some examples in Fig. 5.6.

Similar objects in different context: The first set of results depict examples where language-guided sampling results in very diverse images of the same concept. For example, while visually sampled images (regardless of whether they are self-supervised or supervised) depict shoes on their owl, language-samples depicts 3 images of the same shoe model in different contexts. The third row also depicts humming birds in different poses, while self-supervised models provide birds on a branch, and supervised models provide humming birds taken in similar poses as the source image.

Visual similarity misses the object: The second set show examples where visual similarity misses the salient object in the image. The fourth row is a halibut dish with vegetables. Visual sampling results in other dishes pictured from the top, while language sampling gives us three other halibut dishes with vegetables that look different from the source image. Rows 5 and 6 show examples where visual sampling just focused on the overall appearance missing the herb scissors (row 5) and coyote (row 6). Self supervised models provide nearest neighbors that have animals in snow, but its different animals like a lynx or a dog.

Captions capture subtle relationships: The third set shows examples where the language captures subtle relationships. Can you guess what the captions were? In row 7, the source image was captioned "*itap of a tunnel created by the autumn leaves*." Visual similarity focuses on the trees, while language similarity resulted in images that more clearly depict autumn. In row 8, the source caption mentions a cheetah which can be seen at the right corner of the source image, but the overall sunset appearance results in different sets of visual nearest neighbors. Finally, the caption for row 9 mentions a mating ritual between birds. This element is captured by language guidance while visual similarity retrieves images of animals in grass. These results suggest that conditioning the model similarity on the caption could result in a better posed learning problem.

Vague Captions: Those examples show cases where the caption is very vague or unrelated to the image content, resulting in odd nearest neighbors in the language space. Can you guess the captions from the nearest neighbor images? Answers in footnote.[2] The caption of row 10 refers to the appearance of the eyes of the penguin, but since the "googly eyes" can also refer to a small toy, it retrieves images of that toy being used on a coffee machine and a wall. In row 11, the caption asks what the object is, but this is independent of the object, resulting in language retrievals with miscellaneous objects, while visual retrievals return other insects. Finally, row 12 shows a case where the retrieval uses the dog's name in some context, resulting in the retrieval of other pets playing in gardens. These cases represent limitations of language sampling that might result in poor learning. However, since the core issue arises from misalignment or vagueness in the caption, it is a limitation shared by any model that uses captions and images.

5.7 Limitations

We observe a few limitations in our approach. Image captions can be noisy, vague, and often omit obvious relations in the image [12]. While this broadly affects image-language models, it can result in us retrieving unrelated image pairs. For example, captions like "*I found this in the garden*" or "*Photo from our family trip*" could describe a large range of images, some of which are unrelated. We expanded on this in Sec. 5.6. Image descriptions also depend on the context and the perceiver; *e.g.*, a tourist and an art curator will describe an artwork in very different ways. We observe that descriptions in topic-focused subreddits (*e.g.*, **`r/birdpics`** and **`r/woodworking`**) are more specific than in generic subreddits (*e.g.*,**`r/itookapicture`** and **`r/pics`**). Our experiments in Sec. 5.5.5 support this observation. Since a caption only captures one aspect of the image, sampled pairs can be similar for a variety of reasons. Allowing the model to condition the feature extraction or similarity calculation on captions could alleviate this issue.

[2] Source Image Captions: row 10: "*Built-in googly eyes.*" row 11: "*I found this today. Anyone knows what it is?*" row 12: "*Cinda having fun in the garden!*"

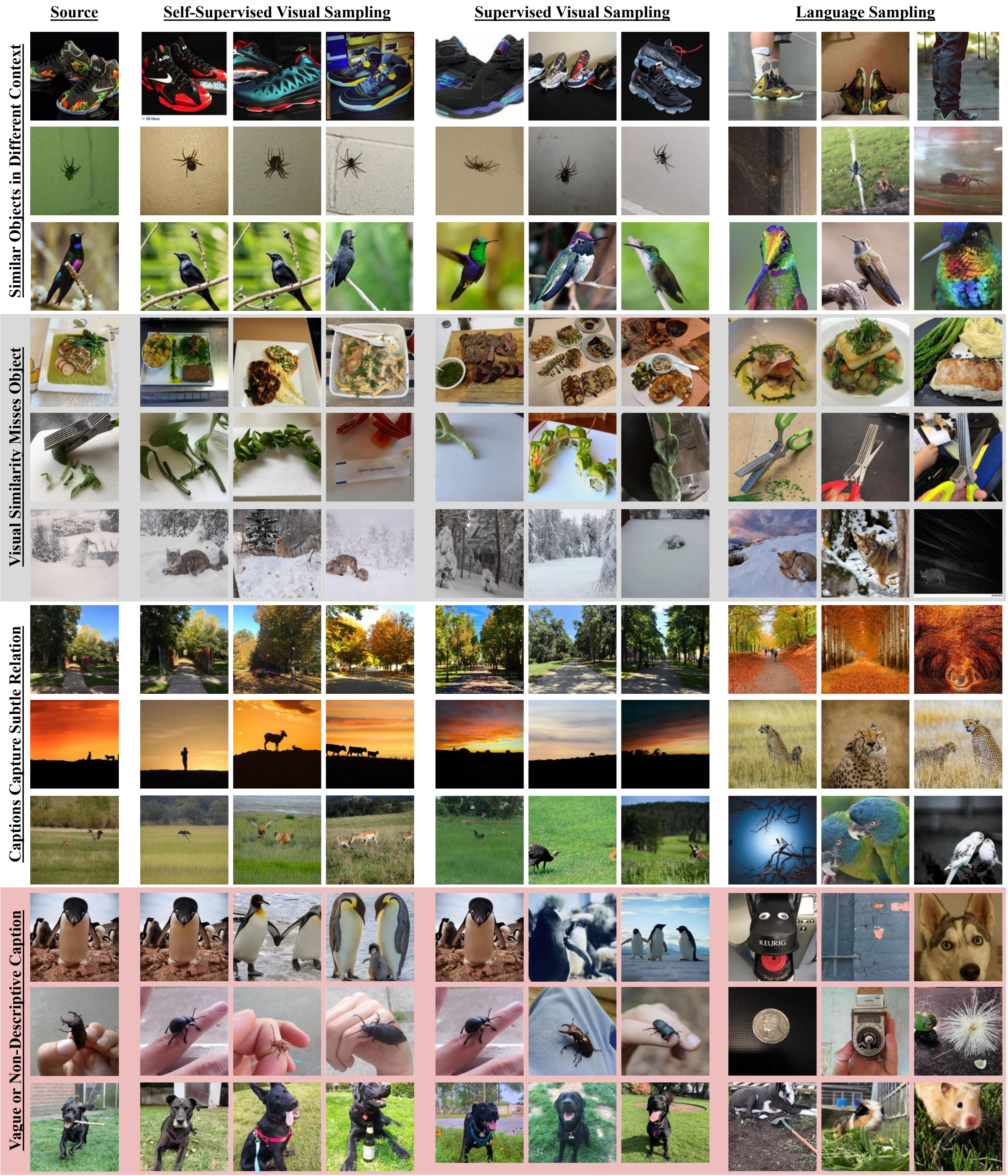

Source
Self-Supervised Visual Sampling
Supervised Visual Sampling
Language Sampling
Similar Objects in Different Context
Visual Similarity Misses Object
Captions Capture Subtle Relation
Vague or Non-Descriptive Caption

Figure 5.6: **Nearest neighbor image pairs sampled using different modalities.**

5.8 Discussion

In this chapter, we propose using language to find conceptually similar images for contrastive learning. This is based on a simple observation: people describe an object in similar ways even when it appears in different contexts. We use pre-trained language models to sample similar captions and use the captioned images for contrastive learning. We hypothesized that using language-guidance, instead of image augmentations, would result in learning invariances

We evaluate our approach on multiple train and test datasets and find that it outperforms previous self-supervised and image-text contrastive models. Our analysis demonstrates the utility of using nearest-neighbor instances for training and the superiority of language sampling over other approaches for unlabeled datasets. Our findings align with prior work that critiques the use of image augmentations [247, 318] and shows the utility of cross-modal guidance [99] and intra-instance relationships [71, 134]. Within the scope of vision and language research, our results demonstrate the potential of incorporating language as guidance rather than as a direct supervisory signal.

In a broader sense, this work provides a new paradigm for learning from multimodal data. We hope that by providing a first step in this direction, this work can inspire others to explore other modalities, such as audio or haptics. Those two modalities lie at the core of the study of cross-modal correspondence in humans [265], and it would be interesting to explore those theories in the context of machine learning. Finally, our approach only focused on global image representations. However, we hope that future work will extend this towards dense representation learning as the ability of the model to identify the corresponding components within the image should provide a strong learning signal for representation learning.

Chapter 6

Evaluating the 3D Consistency of Visual Representations

> The study of vision must, therefore, include not only the study of how to extract from images the various aspects of the world that are useful to us, but also an inquiry into the nature of the internal representations by which we capture this information.
>
> —David Marr, *Vision*

In the previous chapters, we demonstrated how correspondence can be used to learn visual representations without explicit supervision. Our primary claim is that the consistency signals provided by cross-modal correspondence allow the model to learn good representations. In this chapter, we consider the flip-side of this: *do "good" visual representations exhibit semantic and geometric consistency?*

Over the past decade, there has been a plethora of methods for learning good visual representations based on various learning signals and data sources. While earlier work focused on leveraging different forms of supervision, such as image labels, object masks, or 3D annotations, it soon became clear that such methods would be challenging to scale due to the high cost of annotation. This shifted attention to methods that could best leverage two forms of easily scalable types of data: curated [63, 206] or captioned images [65, 219, 245]. The combination of scalable data and large-scale training has yielded visual models with impressive generalization capabilities. Such models can classify [163, 219], caption [164], and generate [36, 229, 237] arbitrary images. Given that those models excel in various image understanding tasks, it became natural to ask what their representations capture.

There has been extensive work evaluating the semantic capabilities of visual models [90, 94, 299]. The standard evaluation of visual representations is ImageNet linear probe classification. The intuition is that good features should allow one to discriminate between different object classes. Furthermore, it has been observed that ImageNet

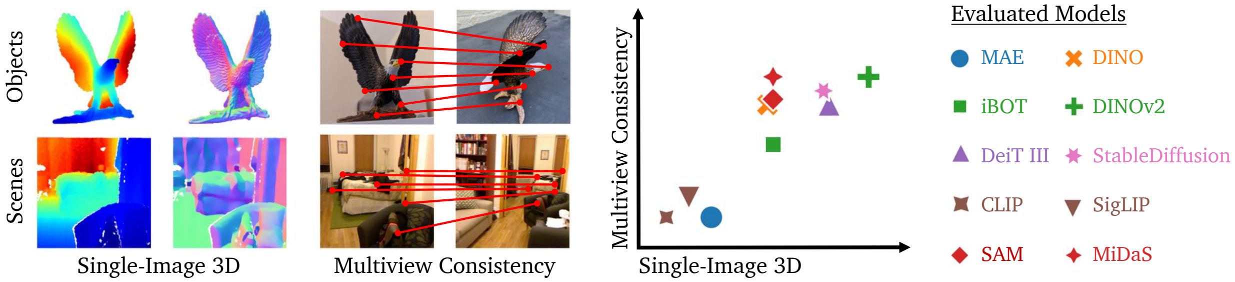


Figure 6.1: **Are visual foundation models 3D aware?** We analyze the 3D awareness of visual foundation models by evaluating how well their features encode depth and surface normals, as well as how consistent their representations are across views.

performance often correlates well with other datasets [144]. More recent work has expanded such evaluations to include dense localization tasks such as detection and segmentation [1, 190, 206, 323] with more mixed results. Overall, it appears that self-supervised and vision-language models learn representations useful for a wide array of visual tasks, but they still struggle with novel objects and compositions [279].

If models have a good understanding of images and their semantics, might they also be representing the 3D world that images depict? Recent work has explored using pre-trained vision models for 3D visual tasks with mixed results. Several works have shown that image-generation models implicitly represent depth in the process of generation [20, 46]. While others have shown that the intermediate representations can be used to estimate semantic correspondence [1, 105, 275, 341] and object pose [92]. However, when those models are used for 3D object reconstruction [216], they suffer from artifacts such as multiple faces that suggest a lack of 3D consistency [172]. Given the mixed performance, it remains unclear how well those models represent or understand the 3D world.

In this chapter, we analyze how well image-trained vision models represent the 3D world. We focus on two different 3D capabilities: single-view surface reconstruction and multi-view consistency. We argue in Sec. 6.1 that those two capabilities capture a representation's 3D awareness and lie at the core of many 3D tasks. We are particularly interested in understanding how different types of image supervision impact a model's 3D awareness. We focus our analysis on several performant vision models, which have been recently referred to as visual foundation models. We consider models that represent different strands in visual learning: classification supervision, generative modeling, vision-language, self-supervised models, and dense supervision. In our experiments, we keep the models frozen and use feature probes to evaluate the model's 3D awareness. We then consider overarching trends and correlations between different models and tasks.

6.1 3D Aware Visual Representations

We first discuss what it means for a representation to be 3D aware. When we view a scene, we seem to effortlessly understand its 3D structure. This enables us to make inferences about how to navigate scenes, manipulate objects, as well as reason about spatial relationships and scene dynamics. These impressive capabilities raise the question of the nature of the representations that mediate these capabilities.

Early work on computer vision proposed explicitly 3D representations such as generalized cylinders [187] or part-based models [83]. It seemed reasonable that if your internal model *was* the 3D model, then it represented the 3D environment by definition. However, our most performant models look nothing like that and instead consist of learning neural networks that represent images through dense feature grids. In order to understand the representations, we need a way to analyze those feature vectors and explain them. Existing interpretability methods, such as GradCAM [246], are not helpful here as they help us associate activations with inputs, while we are interested in the overarching structure of the representation.

To answer this challenge, we look at how this question has been framed and studied in developmental cognitive psychology. Specifically, we take inspiration from prior work on shape perception [142, 264] and mental representations [252, 253]. Spelke et al. [264] presented findings that humans appear to encode specific geometric properties such as distance, angles, and orientation. This provides us with the first property: if a feature is 3D aware, it should allow us to decode these geometric quantities with relative ease. Since such 3D qualities only represent the visible surface, they have historically been referred to as 2.5D [187]. While 2.5D understanding is important, it only captures view-specific surface properties, not the overall 3D shape. We consider how the second-order isomorphism ideas of Shepard and Chipman [252] could be extended to dense features. We posit that they imply some form of 3D consistency: points that are close to each other should have similar representations. This naturally maps to the notion of visual correspondence discussed in the earlier chapter, which is well-studied within computer vision.

We propose evaluating the 3D awareness of models by probing them on two capabilities: single-view 3D and multiview consistency. While those properties are not comprehensive, they capture two essential aspects of 3D awareness. Furthermore, they can be directly mapped to three well-studied problems in computer vision of estimating depth, surface normals, and correspondence.

Another important question is how to probe the models. One common approach is transfer learning, where the pre-trained model is fine-tuned using task-specific supervision. While this can be a good practical choice, it is ill-suited for our analysis for two reasons. First, fine-tuning changes the model weights, which negatively impacts any other zero-shot capabilities it may have. Hence, it sacrifices the generality of the pre-training to specialize the model. This can be problematic if we hope to use the model for several tasks. Second, transfer learning confounds the learned representations with the trainability of the weights. As a result, it is unclear if good transfer performance is due to the model being 3D aware or the model weights being a good initialization [94]. Instead, we propose using trainable probes or zero-shot inference methods that do not change model weights nor significantly alter model capacity. This allows us to evaluate the pre-trained representations of models with the assumption that the same model may be used for a wide range of tasks. This can be very useful for tasks where the same set of features allows one to segment objects, extract their 3D surface, as well as classify them.

6.2 Experiments

The goal of our experiments is to evaluate the 3D awareness of large-scale pre-trained visual models. Specifically, we hope to answer the following questions:

1. Do models encode the geometry of the visible surface?
2. Do the models represent surfaces consistently across views?
3. Are those two capabilities correlated with each other?
4. How does the training objective impact 3D awareness?

We provide high-level descriptions of our experimental setup in this chapter and refer the reader to Appendix C for more details.

Evaluated Models. We consider models that have shown strong image understanding and generalization performance, regardless of their training objective. To focus our analysis, we consider several different forms of training supervision and choose a representative model for each.

- **ImageNet supervision** used to be the standard pre-training recipe due to its strong generalization performance [144, 250] and recent papers have demonstrated their utility when scaled to ImageNet-22k [90]. We use DeIT III [287] vision transformers.
- **Vision-Language models** have recently emerged as a strong backbone due to their impressive zero-shot performance [219]. We use CLIP's vision transformers. We also use SigLIP [339] as a different (non-contrastive) vision-language objective.
- **Text-conditioned Generation** is a novel objective that trains models to generate images

conditioned on a specific text input. Besides its success at generating photorealistic images, such models also learn features that are useful for classification [160] and segmentation [323]. We use StableDiffusion 2.1 [229] as it is the prototypical example of this objective.

- **Self-supervised models** are trained with global or dense objectives on image datasets. While such models do not require labels, they have shown more success on curated datasets such as ImageNet [63] and LVD-142M [206]. We consider two different objectives: masked autoencoding and self-distillation. We consider those objectives as they have been commonly used as feature extractors or backbones to other performant models. We use MAE [103] for masked autoencoding. For self-distillation, we use DINO [33], iBOT [352], and DINOv2 [206].
- **Segmentation models** are trained to localize objects within images. SAM [138] has been recently proposed as a segmentation foundation model that can segment anything and has exhibited impressive generalization and robustness. We use SAM's transformer-based backbone as it provides the feature that supports this impressive performance.
- **Scale-Invariant Depth models** are trained to estimate the pixel-wise depth of the scene up to scale. This has allowed such models to train on a wide range of different scenes and achieve impressive generalization. We use MiDaS [222, 223] as a representative example.

A comment on the fairness of comparisons. One major challenge is how to fairly compare models that were trained with different data and compute resources. It is not possible to recreate most of those models as they are trained with datasets that are not publicly available [206, 219], and even recreating such datasets requires extensive resources [245, 321]. Furthermore, models have different data requirements from labels [287], captions [219], masks [138], or even simple curation [206]. Finally, even if such data is available, the prohibitive compute resources required would limit any kind of broad comparison. Instead, we choose to consider publicly available checkpoints of the same or comparable model sizes. However, the different data sources limit the conclusions that we can draw, as we discuss in Sec. 6.2.3.

6.2.1 Single-View Surface Reconstruction

We first evaluate how well can models represent the visible surface in the image. We consider two tasks for single-view 3D understanding: depth estimation and surface normal estimation. Those tasks are both well-established in computer vision and are commonly

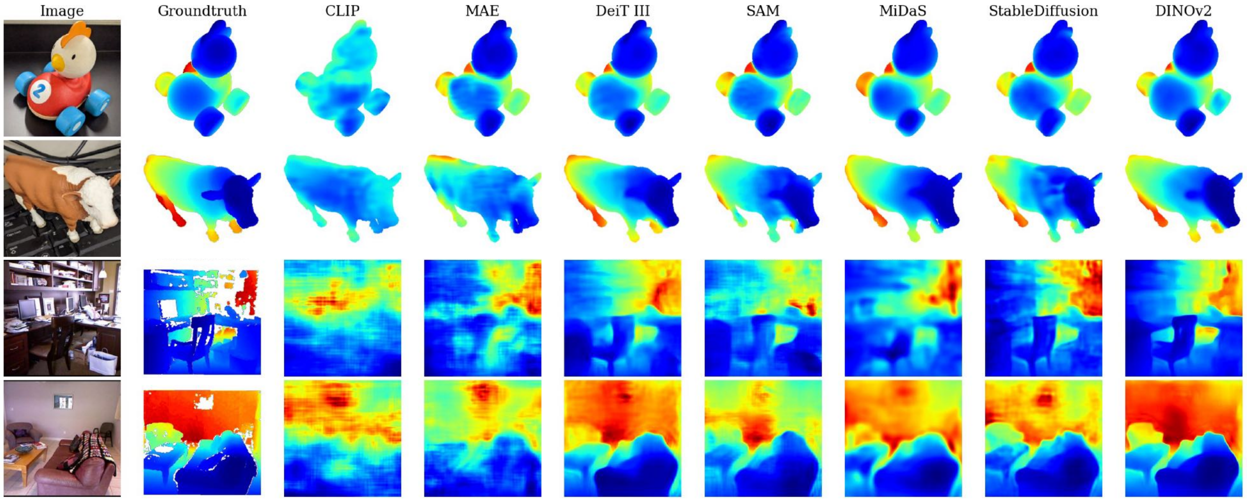


Figure 6.2: **Depth Estimation Results.** We observe that pretrained representations exhibit large variation in their representation of depth, but that their performance is consistent on objects and scenes. CLIP and MAE features do not encode depth and appear to instead capture rough priors such as ”floor pixels are close”. Other models appear to capture the rough structure of the scene and vary in how accurately they capture details. DINOv2 generates the most accurate estimates and accurately captures fine details; *e.g.*, cow’s ear, desk chair, coffee table.

studied in human perception and development [264]. While both tasks are closely related, they appear to rely on different visual cues in both human [143] and computer vision [84].

Evaluation. We apply the standard formulation of depth and surface normals but use scale-invariant depth estimation for objects. Depth losses and metrics often assume that the variance in depth is high compared to the mean depth. This assumption fails for objects where a 10cm object may be at 2 meters away. This results in both the loss and metrics being dominated by predicting the correct scale, which is a known ambiguity in monocular depth estimation. Hence, we instead rely on scale-invariant depth estimation for objects. We report the root-mean-squared prediction error for both tasks in meters and degrees for depth and surface normals, respectively. For scale-invariance depth, the ground-truth depth is scaled to a (0-1) norm, making the error unitless.

Probe. We use a multi-scale probe for both tasks. This deviates from the common choice of linear probing commonly used in bench-marking self-supervised models [144]. Linear probing is useful for semantic tasks since the linear separability of classes is a desired and expected property. However, it is not well suited for dense 3D prediction tasks for two reasons. First, it is unclear if the representation should be linearly separable, especially for

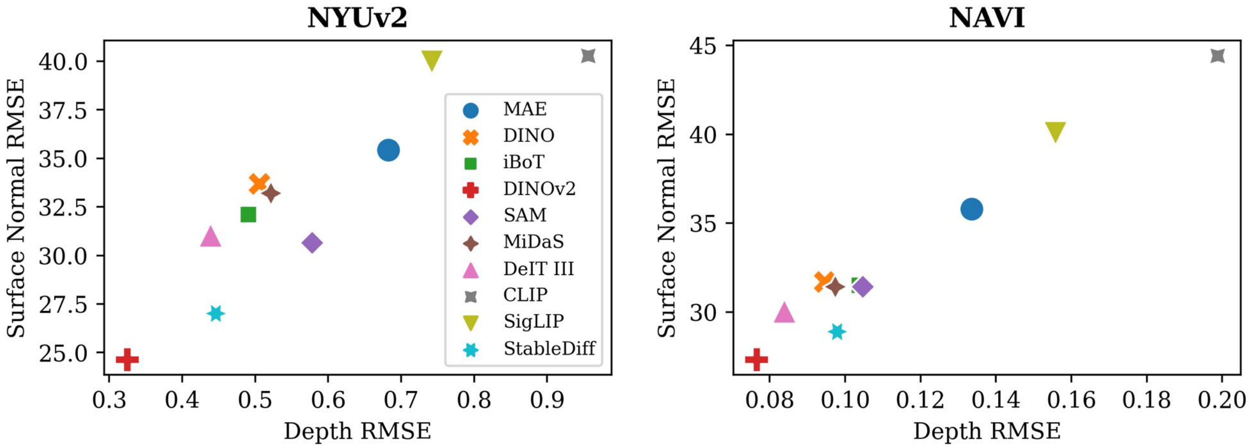


Figure 6.3: **Performance is well-correlated across tasks.** We find that the depth and surface normal performance is strongly correlated for both objects and scene.

regression tasks. Second, dense tasks often utilize multi-scale prediction heads as the information needed is distributed between different parts of the network. Hence, we extract features from multiple stages of the model and use a multi-layer probe. We use the same probe for both tasks with the exception of the final output parameterization: AdaBins [18] for depth estimation and uncertainty-aware angular prediction [11] for surface normals.

Optimization. We train probes for 10 epochs using the AdamW [137, 177] optimizer with a linear warm and cosine decay learning rate scheduler. While longer training further improves performance, trends stabilize after 5 training epochs due to the relatively small capacity of the probe.

Datasets. We evaluate models on two datasets to assess their performance on objects and scenes datasets. For scenes, we use the NYUv2 [257], which is a common benchmark for indoor scene 3D understanding. We also evaluate the performance on objects using the NAVI dataset [122], which depicts a set of object instances in a wide range of scenes and orientations. Both datasets provide aligned depth. For surface normals, we use the annotations generated by Ladickỳ et al. [150] and generate the surface normal annotations for NAVI.

Depth Estimation Results. We observe both a large variance in depth estimation performance across models and tasks as shown in Fig. 6.3. While DINOv2 features can predict depth for both objects and scenes very accurately, CLIP features do not appear to encode any depth information. This is evident when considering the qualitative results in

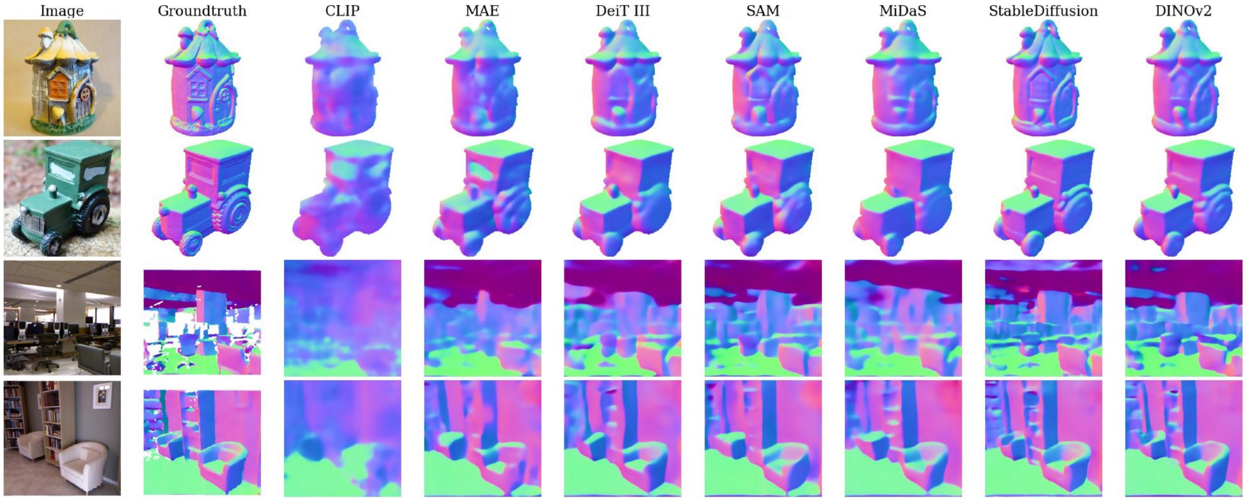


Figure 6.4: **Surface Normal Qualitative Examples.** With the exception of CLIP, models can capture the rough orientation of object and scene surfaces; *e.g.*, floors, walls, ceilings. The main distinctions seem in how well they capture finer details. Similar to depth results, we find that DINOv2 and StableDiffusion perform best and can capture fine details such as the edges of the toy car and the white seat. Surprisingly, we find that SAM's prediction are not as detailed despite its ability to predict accurate segmentation boundaries.

Fig. 6.2. While models can rely on rough semantic priors for scenes and generate coarse predictions—*e.g.*, floors (bottom pixels) tend to be close—this is more difficult for objects, resulting in more random and flat predictions. We find that model performance for objects and scenes is highly correlated with a coefficient of 0.97. This is surprising as one would expect that different pretext tasks would bias representations towards object- or scene-centric representations.

In terms of absolute performance, we find that DINOv2 performs best across the considered models, achieving an RMSE of 0.324 on NYUv2. This is an impressive performance, considering that this is a small probe on the frozen feature of a base-level backbone. To contextualize this, ZoeDepth [19] achieves an RMSE of 0.270 by training a larger backbone (ViT large) on a combination of 12 depth datasets before fine-tuning on NYUv2. This comparison underscores the generality of the representations learned by the models and the non-trivial performance that they achieve.

Another interesting result is that MiDaS [222, 223] does not perform much best on depth probing despite the strong overlap between the downstream data and its pre-training objective. Specifically, MiDaS is trained using a scale-invariant depth estimation objective on a combination of indoor and outdoor depth datasets. Despite this, probing its features results in worse predictions than models trained with self-supervision, class-supervision, or generative objectives. This is very surprising and suggests that strong, more generic pre-training might be just as useful as task-specific pre-training.

Surface Normal Estimation Results. Surface normal probing reveals similar trends to depth estimation. The qualitative results, shown in Fig. 6.4, indicate that most models capture coarse (planar) orientation but struggle with details and areas of large normal variation. It is unclear if this limitation is due to model representation or the coarse spatial resolution of the feature embedding.

In terms of absolute performance, we find that the best model performs favorably compared to the state-of-the-art. DINOv2 achieves an RMSE of $24.6°$ which is close to the $22.8°$ RMSE achieved by iDisc [214], the best-performing model on NYUv332. iDisc achieves this impressive performance by introducing several innovative components that are designed for dense geometric tasks and training the full model on this task. It is surprising that a self-supervised model learns representations that are even comparable.

Summary: Our single-view experiments suggest that most pre-training objectives end up encoding information about the visible surface. Furthermore, it seems that this ability is not specific to objects or scenes, as indicated by the high correlation shown in Fig. 6.3. Self-supervised models that rely on both dense and image-level objectives, such as DINOv2 and iBOT, perform very well. This is interesting as DINOv2 outperforms StableDiffusion, whose image generation pre-training objective seems to require a better understanding of scene geometry. The other surprising finding is how poorly language-supervised such as CLIP and SigLIP perform, especially when considering their strong semantic understanding. Overall, our experiments suggest that most visual foundation models do represent the visible surface, with the notable exception of language-supervised models.

6.2.2 Multi-view Consistency

We now evaluate the multiview consistency of representations. While single-image 3D tests, if the model represents the visible surface, multiple 3D tasks require those representations to be consistent across views. The canonical task for this is geometric correspondence estimation, where the goal is to identify image patches that depict the same 3D point.

Geometric vs. Semantic Correspondences. Prior work has shown that the dense features learned by StableDiffusion and DINO are useful for semantic correspondence [1, 182, 341]. Semantic correspondence [16] generalizes the correspondence problem to matching similar semantic parts across views; *e.g.*, match a dog's left ear across images

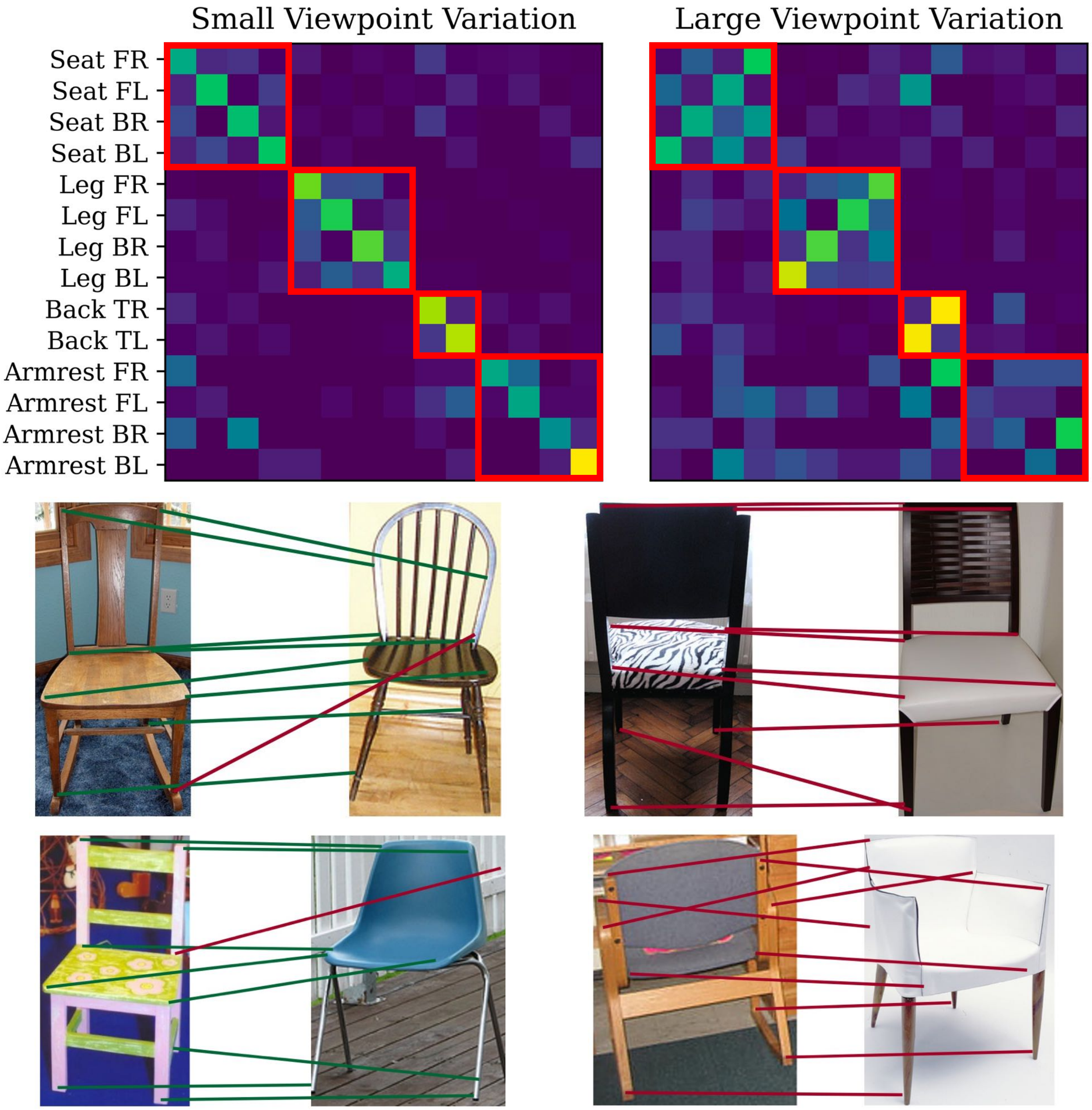


Figure 6.5: **StableDiffusion correspondence on SPair [195] chairs.** StableDiffusion appears consistently represent semantic parts and their 2D location. This results in accurate correspondence between objects viewed from similar angles, but very systematic errors for objects viewed from different poses as shown in the confusion matrices. Semantically similar classes are highlighted and correspondences are color coded for recall.

of two different dogs. While this provides a useful signal, we note that semantic and geometric correspondence are two different tasks.

Models can achieve good semantic correspondence performance even if they lack 3D awareness. While StableDiffusion can estimate very accurate semantic correspondence, we find that it makes very systematic errors that indicate a lack of 3D consistency. As shown in Fig. 6.5, the model will often predict false correspondence between semantically similar parts if they are located in the corresponding part of the image. For example, seat corners in one image are mapped to seat corners in the other image. However, the mapping depends on the 2D location on the object projection (seat corner on the left) as opposed to its 3D location (back left seat corner). This suggests that the model representation is a combination of the semantics of the part and its location relative to the object mask, not the object itself. This distinction is similar to ego-centric vs. allocentric, but rather depending on the viewer, it depends on the 2D projection vs. the 3D object.

To further support this observation, we analyze StableDiffusion's performance on SPair-71k [195], which is a common benchmark for semantic correspondence. However, instead of the common approach of evaluating key point recall, we match the predicted correspondence to the nearest key point and analyze the resulting confusion matrices. We focus on the chair class here, but we find these results across classes and present additional results in the supplemental. For image pairs with a small viewpoint variation, we find that the model achieves a good performance. However, for large viewpoint variations, the model consistently confuses semantically similar parts with each other. This is indicated by the flipped diagonals for semantically similar parts. We suspect that this also explains the Janus problem observed in Diffusion-based 3D reconstruction. Since most faces are front-facing, a model will often assume that an ear at the top left is the left ear and will generate a corresponding face. Our analysis suggests that semantic correspondence is not suitable for evaluating the 3D understanding of the model.

Task. We evaluate the models on 3D correspondence estimation task: given two views of the same object or scene, estimate the correspondence between them. Similar to our single-view 3D experiments, we consider both scenes and objects. For scenes, we evaluate our model on the Paired ScanNet [56] split proposed by Sarlin et al. [241]. For objects, we sampled view pairs from the NAVI wild-set, which depicts the same object instances in different environments. We sample views to have a maximum rotation of 120 degrees to ensure there is a mutually visible surface.

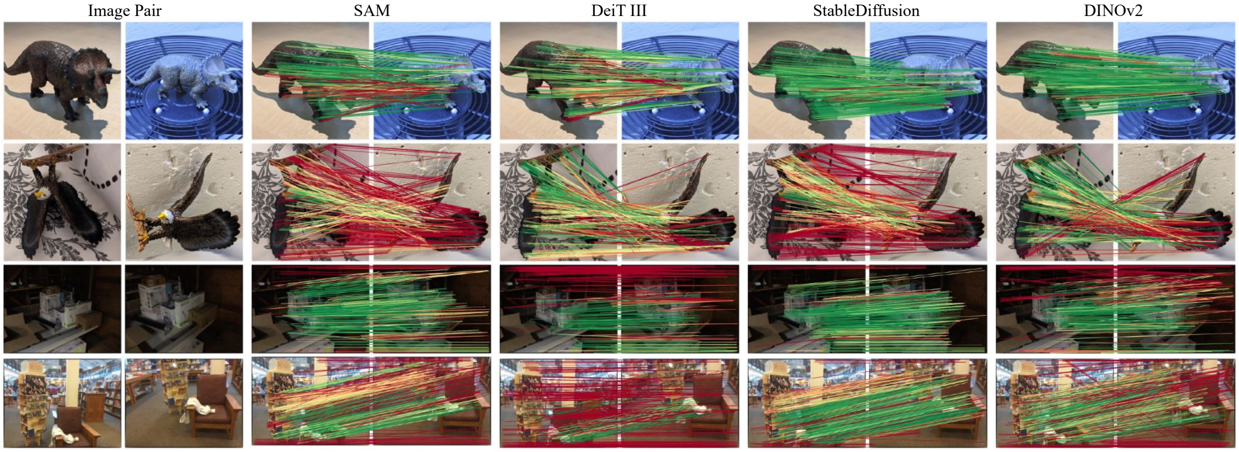


Figure 6.6: **Correspondence Estimation Qualitative Results.** We observe that models can estimate accurate correspondence for small viewpoint changes, but struggle with large viewpoint changes. This is true even if the change is an in-plane rotation as shown with the eagle. This pattern is consistent for both objects and scenes, although performance is not well-correlated: SAM and StableDiffusion perform better for scenes, while DeiT and DINOv2 are more consistent for objects. Correspondence color-coded for accuracy.

Inference Procedure. We consider a zero-shot inference procedure since we are interested in the consistency of the features as they are. This matches several recent approaches that evaluate features without additional training [1, 206, 275, 341] as well as the traditional correspondence estimation pipeline that depended on pre-trained feature descriptors. Following El Banani and Johnson [74], we use sample nearest neighbors in feature space and use the ratio test to filter the top 1000 matches. For each model, we use the layer that results in the best correspondence performance, as this can highly impact performance.

Evaluation: We evaluate the multiview consistency of representation using correspondence accuracy; *i.e.*, how many of the predicted correspondences are within some threshold in 3D space. Correspondence error is often computed in pixels to account for the large variation in depth; *e.g.*, a prediction off by 1 pixel can be a few millimeters on a close-by surface or several meters for outdoor scenes. While this is useful for scenes, it is not well-suited for object-level correspondence as many object parts get self-occluded. To account for those differences, we compute recall based on a 3D threshold for objects and a pixel threshold for scenes. We also compute average performance for different viewpoint changes to evaluate how it affects the model's performance.

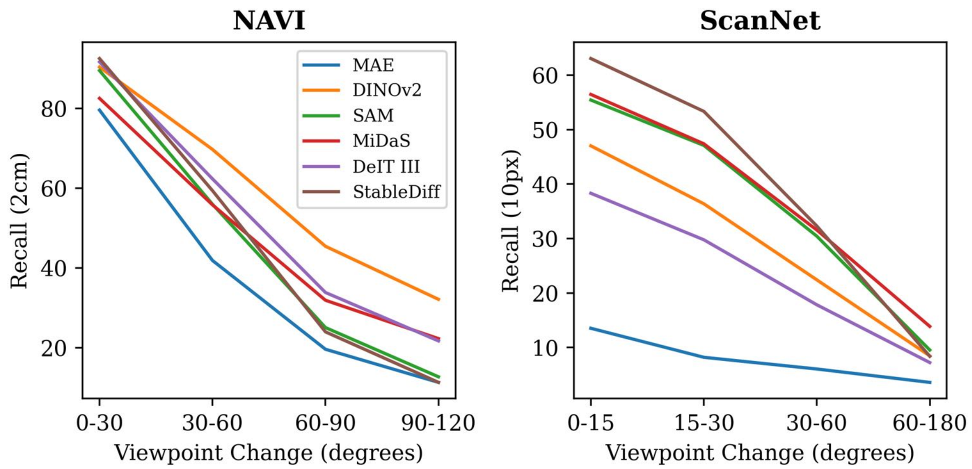


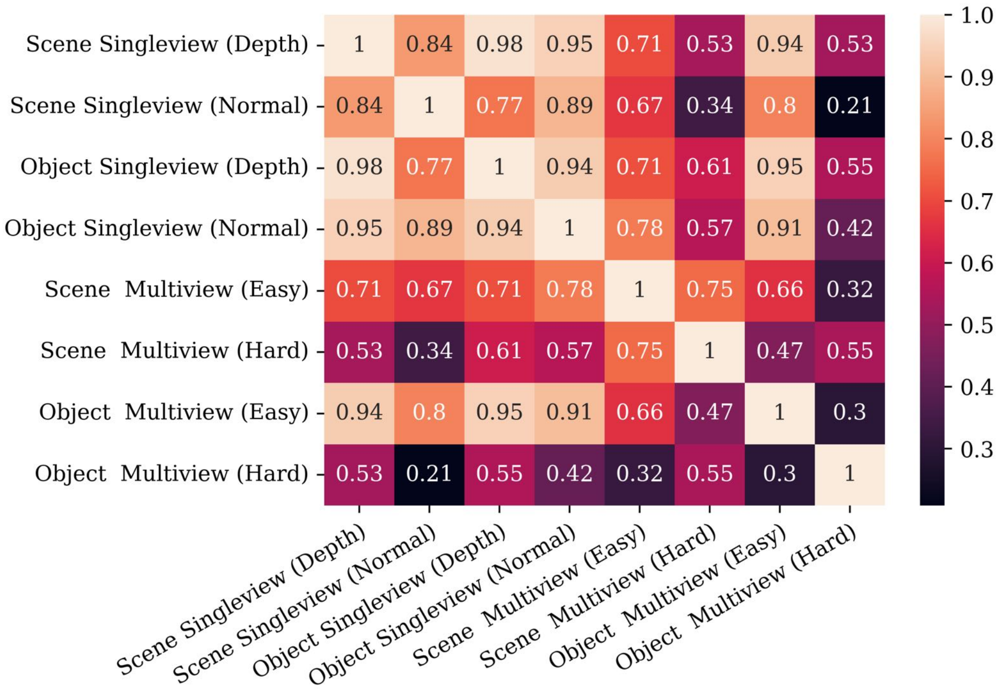
Figure 6.7: While all models do poorly for larger viewpoint changes, StableDiffusion performance drops the fastest, suggesting a strong single view performance and a weak multiview performance.

Results. Models tend to estimate accurate correspondence for small viewpoint changes as seen in Fig. 6.7 but that performance quickly deteriorates for larger viewpoint changes. While this is expected as larger viewpoint changes make the task more difficult, the rate of deterioration is interesting. StableDiffusion performance drops far quicker than other models. This suggests that its representations are not 3D consistent but rather capture a combination of semantics and viewpoint. This is supported by both the results on SPair in Fig. 6.5 and the qualitative results in Fig. 6.6. Meanwhile, DINOv2 performance drops slower than the other models, even on ScanNet, where it performs poorly.

Another interesting trend we observe is a much weaker correlation between the performance on objects and scenes. While performance on single-view 3D tasks was highly correlated, we find that some models perform well on objects but not scenes and vice-versa. This suggests that while a model's representations may encode information about the visible surface, its representations are view- or context-dependent. As a result, the increased variation results in a fast drop in performance for large viewpoints and suggests a limited multiview consistency in the features.

6.2.3 Analysis

We now analyze trends across all tasks to understand relationships across different tasks. We are also interested in understanding the relationship between training objectives and 3D awareness. We note that while we highlighted specific models in our analysis, we evaluated a much larger set of model variants and computed all correlations on the full set. Please see Appendix C for the complete set of results.



	Scene Singleview (Depth)	Scene Singleview (Normal)	Object Singleview (Depth)	Object Singleview (Normal)	Scene Multiview (Easy)	Scene Multiview (Hard)	Object Multiview (Easy)	Object Multiview (Hard)
Scene Singleview (Depth)	1	0.84	0.98	0.95	0.71	0.53	0.94	0.53
Scene Singleview (Normal)	0.84	1	0.77	0.89	0.67	0.34	0.8	0.21
Object Singleview (Depth)	0.98	0.77	1	0.94	0.71	0.61	0.95	0.55
Object Singleview (Normal)	0.95	0.89	0.94	1	0.78	0.57	0.91	0.42
Scene Multiview (Easy)	0.71	0.67	0.71	0.78	1	0.75	0.66	0.32
Scene Multiview (Hard)	0.53	0.34	0.61	0.57	0.75	1	0.47	0.55
Object Multiview (Easy)	0.94	0.8	0.95	0.91	0.66	0.47	1	0.3
Object Multiview (Hard)	0.53	0.21	0.55	0.42	0.32	0.55	0.3	1

Figure 6.8: While performance on single view tasks and correspondence estimation with similar viewpoints is strongly correlated, correspondence estimation across large viewpoint is not strongly correlated with any tasks.

One important question is how correlated are different tasks. For example, if the model's representations accurately depict depth, how likely is it that they are also useful for surface normals or correspondence? To address this question, we compute the correlations between the models' aggregated performance across multiple tasks. For single-view 3D, we separate the depth and surface normal performance for scenes and objects. For multiview consistency, we separate the correspondence recall for small and large viewpoint changes. Specifically, we consider the smallest and largest viewpoint bins for NAVI and ScanNet. We compute the correlations between all task pairs as shown it in Fig. 6.8. Our results indicate that while single-view tasks are strongly correlated, this correlation is much weaker for multiview consistency. Surprisingly, there is only a weak correlation between correspondence performance for small and large viewpoint changes.

While single-view performance and correspondence estimation with small viewpoint changes are well-correlated, none of the tasks are well correlated with correspondence across large viewpoint changes. One explanation that is consistent with our findings on SPair in Fig. 6.5 is that while the representations might not be 3D consistent, they might be view-consistent. In that case, the model generates consistent representations of similar views of the object, but has a limited understanding of how the different views are connected. While this formulation is consistent with our findings, more research is needed to further elucidate the nature of the representations learned by models.

Another important question is what properties of the training objective improve performance. Our results suggest that self-supervised learning seems to enable very strong feature learning. This is not simply due to scale or model capacity as DINOv1 and iBOT achieve a strong performance despite being trained on ImageNet-1k. Furthermore, vision and language models consistently perform poorly across tasks. While one explanation is that learning general semantics discourages the model from encoding 3D properties, we note that DeiT, which is trained on a classification task, performs very well. Furthermore, this is not a unique property on CLIP's training data as both SigLIP and the LAION-trained CLIP underperform as well. While our experiments point to self-supervised and text-conditioned generation as the strongest performing tasks, it remains unclear if this is due to the task itself or some auxiliary detail of the training.

6.3 Limitations

Our goal is to understand the degree to which current large-scale models "understand" the 3D world that images depict, as well as what factors encourage or discourage such understanding from emerging. This is very challenging as our collective understanding of what models learn or how they represent what they learn is still very limited. Furthermore, there are major debates about how 3D geometry should be represented as well as what it even means for a model to have 3D understanding. Finally, there are concrete challenges in evaluation that make it difficult to conduct controlled experiments. The study presented in this work is a first step toward answering these questions. While our experiments and analysis provide some initial answers to this question, our study suffers from several limitations that limit the strength of the conclusions we can make and point to several avenues for future exploration. We discuss the limitations below and outline some open questions that we hope future work will address.

Comparisons are limited to publicly available checkpoints. We focused on our analysis on publicly available checkpoints due to their availability as well as common use in the literature. However, as a result, our experiments often compare models trained with different recipes on different datasets. This is a significant confounding variable as it is unclear if the trends we are observing are due to the main differentiators we observe or some minor implementation detail.

Ideally, one would train the same backbone architecture on the same dataset with the same training recipe but with different objectives as we did in Chapter 5. However, the computational resources needed to train and tune all models are prohibitively large. This problem is also likely to be exacerbated with the current trend of moving towards ever larger scales. Beyond the resources required, different approaches often have different data requirements such as curation, labels, captions, or dense annotations. There are currently no large-scale datasets that meet all those requirements.

We attempted to make comparisons more fair by choosing model checkpoints of comparable model capacity and pretraining data scale. While we expected that the dataset or pre-training scale might end up dominating all other effects, our experiments suggest that other factors can be more important. For example, while CLIP is trained on a much larger dataset than DINO, DINO consistently outperforms CLIP. Furthermore, model performance does not seem to be very sensitive to training data with CLIP achieving similar performance whether it was trained on WIT or LAION. We hope that our analysis identified interesting patterns and that future work can conduct much more controlled experiments that focus on a specific model comparison or dataset comparison.

Our analysis focuses on two specific aspects of 3D awareness. Our ability to perceive and infer 3D properties is remarkable. While many tasks can showcase this ability, we restrict our analysis to single-image surface reconstruction and multi-view consistency. While those two aspects are fundamental to 3D understanding, they are not comprehensive. The ability to reconstruct the full 3D shape, predict deformation, and estimate physical properties such as support and containment all fall under the general umbrella of 3D understanding. However, it is unclear which of those properties should be readily perceived from the image as opposed to inferred with more complex processing. While such delineations can be more philosophical in nature, they can guide the experimental design as it is important to understand what we're looking for before designing experiments to find it. We expect that more comprehensive benchmarks of 3D understanding should measure such capabilities as well, and we hope that this work provides an initial step towards the study of this problem.

Our experimental methodology focuses on probing methods. Our analysis has focused on linear probe and zero-shot analysis approaches. We have done this to analyze the features as they are without changing them to better adapt to the 3D tasks. While we argue that frozen features provide a more accurate understanding of the 3D awareness of the features, it would definitely be useful to understand how much of those patterns transfer to fine-tuning setups. Furthermore, if we consider recent advances in natural language processing, we see a rise in in-context learning with similar adaptions in computer vision. While linear probes could still be applied, it is likely that prompting-based methods will be more suited to analyze the 3D awareness of such models.

6.4 Related Work

Analysis of Large-scale Pre-trained models. Since the recent revival of deep learning, there has been a lot of effort into understanding how and what these models learn [94]. Early work focused on analyzing what those models could be used for [39, 95, 144] as well as providing some interpretability into what they were learning [246]. We have been inspired by recent work on analyzing whether generative models represent depth in the process of generating images [20, 46]. While this work shares our goal of understanding whether those representations are 3D aware, their analysis is specific to generative models and does not tackle whether the model can represent depth for an image it did not generate. This also makes it difficult to analyze non-generative models. More closely related to our analysis is the concurrent work of Zhan et al. [340], which analyzes the 3D understanding of StableDiffusion through a series of binary classification tasks. While we share the same goal, we argue that forcing the model to densely estimate the 3D properties directly provides more analytical power and is less susceptible to semantic priors. Our analysis also expands to include multiview consistency, not just depth and normals, which were the focus of [340].

Using foundation models for 3D tasks. There has been a growing interest in leveraging large-scale pre-trained models for 3D tasks. One line of work extracts features from models for correspondence estimation [1, 105, 182, 206, 275, 341] and pose estimation [92, 343]. Others have shown how those models could be fine-tuned for accurate depth estimation with Zhao et al. [349] achieving state-of-the-art performance by simply fine-tuning StableDiffusion. Another line of work combines image generation with 3D representations for text- or image-conditioned 3D reconstruction [216, 300, 320]. While those methods generate impressive 3D shapes, it has been observed that their generations are not 3D consistent and can generate animals with multiple heads (the Janus Problem). Fortunately, recent work has shown that fine-tuning with 3D data can improve the generation quality [136] and alleviate the 3D inconsistencies [172, 217, 220, 254]. While this work showcases the utility of foundation models for 3D understanding, the features are often used within a larger system, making it unclear how much the features capture the 3D property as opposed to serving another function in the larger system. We have been inspired by this line of work to study the 3D awareness of representations. We hope that our findings help clarify some of the existing trends in the field and provide some guidance on how to select models for specific 3D tasks.

6.5 Discussion

In this chapter, we presented an exploratory study into the 3D awareness of visual models; *i.e.*, how well do the representations capture the 3D-ness of the scenes and objects? We posit that 3D awareness can be evaluated along two dimensions: (1) encoding the visible surface and (2) consistently representing the surface across views. We use trainable probes and zero-shot inference methods to evaluate the frozen features of those models.

Our results revealed some interesting trends. First, we found that most models learn representations that encode the depth and orientation of the visible surface, with vision-language models being the notable exception. While it is possible that "3D understanding" emerged for all those models, it is more likely that this performance is the result of good discriminative dense features. Future work should analyze the relationship between depth understanding and other dense tasks, such as segmentation, to clarify this point. Perhaps good dense features are all you need?

Second, we found that while models can estimate accurate correspondence across images of a similar viewpoint, they struggle with large viewpoint changes. This indicates a lack of multiview consistency, but it is unclear why this happens. One possibility is that models are learning view-dependent representations. This could be similar to the theories of shape perception proposed by Koenderink and Van Doorn [140, 141] where representations focus on 2D projection representations that are connected with an aspect graph. Another possibility is that the features are consistent and that our zero-shot probing approach is too rigid to unveil that. Future work should explore the use of probes to assess this question. A final possibility is that current models are simply good "image models" with no 3D awareness or consistency. Under this interpretation, the only thing that is tying images together is the semantics it learns from the data curation or supervisory signal. This would explain the discrepancy between semantic and 3D correspondence estimation.

The study presented in this chapter is only a first step toward understanding the 3D awareness of visual models. We hope that our findings will encourage more research on the 3D understanding of foundation models and the development of other training approaches that are more 3D aware.

Chapter 7
Discussion

This dissertation presented the idea of using cross-modal consistency for both learning and understanding visual representations. We motivated this goal by learning from the signals available to us in the environment: information we can easily obtain from the environment without requiring an explicit teacher. Our core observation is that our environments, despite their complexity, present consistent patterns across modalities. This allowed us to develop models that leverage cross-modal correspondence to learn representations in one modality using structure in another. We operationalized this idea in two settings: RGB-D video and images with free-form captions.

In RGB-D video, the idea of correspondence was very simple: image patches that depict the same 3D point in space. We integrated correspondence into an end-to-end learning pipeline along with differentiable alignment and synchronization algorithms based on more traditional algorithms. We then developed models that learned from photometric consistency (Chapter 2), geometric consistency (Chapter 3), and multiview consistency (Chapter 4). Our models performed on par with supervised approaches that relied on 3D reconstructions to obtain pose or correspondence supervision despite only relying on the raw RGB-D stream for learning. Through this line of work, we aim to demonstrate two things: First, the consistency signals inherent in our sensory stream provide a very strong signal for learning that renders explicit supervision unnecessary; second, correspondence allows us to sample learning instances from this stream by leveraging cross-modal relationships.

We then deviated from the common formulation of point correspondences in 3D and explored the idea of conceptual correspondence from language. This work was motivated by a very simple observation: when different people observe the same visual information, they tend to describe it in similar ways. Hence, similarly captioned images should be represented in similar ways. We propose language-guided visual learning and show that it outperforms self-supervised approaches that rely on image augmentations as well as language-supervised approaches that jointly embed images and text in the same space.

While we proposed a vision-language approach, we discussed in Chapter 5 how this idea is far more general and presents a new paradigm for thinking about learning from multimodal data through leveraging cross-modal relationships of unimodal representational spaces.

Finally, we took a step back and explored how correspondence could be a useful tool to analyze features. We studied the 3D awareness of visual representations to understand how well they represent the 3D structures whose images they train on. We found that most models appear to encode single image 2.5D information, such as depth and surface normals, with some degree of accuracy. Nevertheless, most of those models struggled with multiview consistency, with their errors suggesting a focus on semantics and relative location in the image plane. Those results suggest a path forward toward better understanding the strengths and limitations of visual representation learning approaches and developing models that better capture our 3D world.

Taken together, this dissertation presents a strong case for rethinking visual representation learning and moving from supervised settings or dataset-centric self-supervised approaches toward learning from signals available in the environment. To conclude this dissertation, we frame our findings in a larger framework and suggest some avenues for future work.

7.1 Towards Ecologically Plausible Perceptual Learning

In our quest to build machines that can see, we have found datasets to be incredibly useful for developing, training, and benchmarking models. Without the availability of nicely curated datasets, the work in this dissertation would not have been possible. However, datasets are snapshots of the world that often depict a simplified, limited, and often biased view of reality [286]. Ideally, models could learn to see, or see better, through their own experiences and interactions. Toward this goal, we posit that we should be developing ecologically plausible perceptual learning approaches.

Perceiving agents do not exist in isolation: their environment determines both the signals they can perceive and learn from as well as the perceptual tasks they need to perform. Consider the following three key ways that an environment affects the design of a computer vision system: (1) input data, (2) learning signal, and (3) evaluation tasks. For a system to be ecologically plausible, its input and learning signal should arise directly from the environment; *i.e.,* sensory data or interaction. This makes it difficult to justify the reliance on ground-truth segmentation masks or 3D meshes within a long-term strategy. Considering the environment also reveals the myriad sensory modalities we could leverage for learning and perception: instead of just relying on image data, we can integrate other sensory modalities such as depth, haptics, and proprioception. Finally, our evaluation tasks should capture the complexity, variety, and novelty inherent in the real world.

In the previous chapters, we focused on two signals that are a step towards more ecologically plausible learning. RGB-D video streams are a form of raw sensory data. Furthermore, most of the learning is happening between close-by frames, making it possible to deploy this directly on a robot; although it is possible that randomization is needed for the current approach to work well. Language, while not sensed directly, arises naturally in interactions with other humans. One could easily conceive of a system that extends our approach by maintaining a memory of previous interactions to learn from. This is similar to the visual memex proposed by Malisiewicz and Efros [185].

We hope that the work presented in this dissertation will motivate the community to explore such approaches and move towards more ecologically plausible learning. We finally discuss two interesting avenues that would be interesting to explore further.

Learning from Visual-Haptic Correspondence. The relationship between haptic and visual perception has been a focus of philosophical [174] and psychological inquiry [107, 258]. While there has been some recent work on visual haptic learning [28, 156], this area remains largely unexplored. One exciting question in this area is the Molyneaux Problem [174], which asks whether a person born blind who learned to differentiate between objects by touch could immediately differentiate them by sight if they regained sight. While answering this question is very difficult, Held et al. [107] showed that newly sighted individuals failed to do this upon regaining sight but could learn to match objects within a few weeks. This is an avenue that is interesting to explore through machine learning: do visual and haptic embedding spaces naturally align when learned separately? How much data is needed to learn how to map from one modality's representations to another?

Discovering Concepts from Cross-Modal Structure. Infants generalize very well to entire categories from a single instance [259]. Cognitive science researchers posited that this arises from their rich, multi-modal experience. While single-modality correlations can be confusing—*e.g.*, cats and dogs are furry, different dogs can sound the same—cross-modality correlations are often far more informative. In this dissertation, we focused on cross-modal correspondence. However, an agent that reasons over the cross-modal relationships could learn to discover novel concepts and use them to further refine their perceptual models.

Appendix A

SE(3) Camera Synchronization Algorithm

We explain the Camera Synchronization algorithm described in Chapter 4 in more detail.

Notation for SE(3) matrices. Recall that for pairs of frame $i < j$,

$$\mathbf{T}_{i,j} = \underset{\mathbf{T} \in \mathrm{SE}(3)}{\arg\min} \sum_{\text{inliers } (p,q,w) \in \mathcal{C}_{i,j}} w||\mathbf{x}_q - \mathbf{T}(\mathbf{x}_p)||_2^2$$

is our estimate for the relative transformation from camera i to camera j. We can write $\mathbf{T}_{i,j}$ as a 4x4 matrix consisting of a rotation and translation,

$$\mathbf{T}_{i,j} = \left[\begin{array}{c|c} R & 0 \\ \hline t & 1 \end{array}\right], \qquad R \in \mathsf{SO}(3),\ T \in \mathbb{R}^3.$$

$\mathbf{T}_{i,j}$ acts on points $\mathbf{x} = (x_1, x_2, x_3, 1)$ in homogeneous form by right multiplication

$$\mathbf{T}_{i,j}(\mathbf{x}) = (x_1, x_2, x_3, 1) \times \mathbf{T}_{i,j}.$$

Confidence-weighted transformations. Recall that $c_{i,j}$ is a confidence value attached to $\mathbf{T}_{i,j}$ for $i < j$. Let $\mathbf{S}_+ \subset \mathbf{R}^{4\times 4}$ denote the set of 4×4 matrices with the form:

$$\alpha \left[\begin{array}{c|c} R & 0 \\ \hline t & 1 \end{array}\right], \qquad \alpha \geq 0, R \in \mathbb{R}^{3\times 3},\ T \in \mathbb{R}^3.$$

Elements of $\mathbf{S}_+$ can be projected onto SE(3) by dividing by α, and then using SVD to project R onto SO(3).

Note that $\mathbb{S}_+$ is closed in the sense that if $A, B \in \mathbf{S}_+$ and $\alpha \geq 0$, then $A + B$, $A \times B$ and αA are all in $\mathbf{S}_+$ too.

Confidences as jump probabilities We will make two simplifying assumptions. First, we will assume that the $c_{i,j}$ have been scaled so that the rows sum to one: for all i, $\sum_j c_{i,j} = 1$. Second, we assume that for each i, $c_{i,i+1} > 0$. With these assumptions in place, $C = [c_{i,j}]$ is the stochastic matrix for an N-state Markov chain (X_t),

$$c_{i,j} = \mathbb{P}[X_{t+1} = j \mid X_t = i], \quad t = 0, 1, 2, \ldots.$$

The Markov chain is [158]:

- lazy: $\mathbb{P}[X_1 = i \mid X_0 = i] = c_{i,i} = 1/2$ as $c_{i,i} = \sum_{j \neq i} c_{i,j}$,
- connected: for all i, j, for some t sufficiently large $\mathbb{P}[X_t = j \mid X_0 = i] = (C^t)_{i,j} > 0$, and
- time-reversible: $\pi_i C_{i,j} = \pi_j C_{j,i}$ for all i, j with $\pi \in [0,1]^N$ the equiilibrium distribution.

By the Perron–Frobenius theorem and the laziness property, the eigenvalues of C can be written as

$$1 = \lambda_1 > \lambda_2 \geq ...\lambda_N \geq 0. \tag{A.1}$$

The spectral gap $1 - \lambda_2 > 0$, so convergence to equilibrium is exponential,

$$\mathbb{P}[X_t = j \mid X_0 = i] = (C^t)_{i,j} = \pi_j + \mathsf{O}(\lambda_2^t).$$

The pairwise-transformations matrix In Sec. 4.3.4, Eq. (4.3), we define a $4N \times 4N$ matrix $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} c_{1,1}\mathbf{I}_4 & c_{1,2}\mathbf{T}_{1,2} & \cdots & c_{1,N}\mathbf{T}_{1,N} \\ c_{2,1}\mathbf{T}_{2,1} & c_2\mathbf{I}_4 & \cdots & c_{2,N}\mathbf{T}_{2,N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ c_{N,1}\mathbf{T}_{N,1} & c_{N,2}\mathbf{T}_{N,2} & \cdots & c_{N,N}\mathbf{I}_4 \end{bmatrix} \in \mathbf{S}_+^{N \times N} \subset \mathbb{R}^{4N \times 4N},$$

consisting of an $N \times N$ grid of elements of $\mathbf{S}_+$. To motivate the definition of $\mathbf{A}$, we can interpret it as generating a random walk on the set $\{1, 2, \ldots, N\} \times SE(3)$,

$$\mathbb{P}[(X_{t+1}, Y_{t+1}) = (j, Y_t \times \mathbf{T}_{i,j}) \mid X_t = i] = c_{i,j}, \qquad t = 0, 1, \ldots$$

The expected value in $\mathbf{S}_+$ of the Y-component of the walk is a weighted sum of the products of pairwise transformations. Pairwise transformations with greater confidence contribute more strongly to the sum.

The solution to the synchronization problem is related to the eigenvectors of $\mathbf{A}$. *If* there is a global collection of cameras $(\mathbf{T}_i)$ such that $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$, then $\mathbf{A}$ will have four independent eigenvectors with eigenvalue one, *i.e.*

$$[\mathbf{T}_1 \dots \mathbf{T}_N] \times (\mathbf{A} - \mathbf{I}_{4N}) = \mathbf{0}, \qquad [\mathbf{T}_1 \dots \mathbf{T}_N] \in \text{SE}(3)^N \subset \mathbb{R}^{4\times 4N}.$$

All other eigenvalues λ will satisfy $|\lambda| \leq \lambda_2$ (c.f. inequality equation A.1 for the eigenvalues of Markov chain X_t, and properties of matrix determinants). As integer $k \to \infty$, each of the N rows of $\mathbf{A}^k$ will converge to a globally consistent set of cameras. The different rows will yield essentially the same solution but differing by an SE(3) transformation of the global coordinates.

More generally, if no such perfect solution exists, then we want to find

$$\begin{aligned}
&\mathop{\arg\min}_{\{\mathbf{T}_i \in \text{SE}(3):1\leq i\leq N\}} \|[\mathbf{T}_1 \dots \mathbf{T}_N] \times (\mathbf{A} - \mathbf{I}_{4N})\|_F^2 \\
&\qquad = \mathop{\arg\min}_{\{\mathbf{T}_i \in \text{SE}(3):1\leq i\leq N\}} \sum_j \left\| \mathbf{T}_j - \sum_i c_{i,j}\mathbf{T}_i\mathbf{T}_{i,j} \right\|_F^2 \\
&\qquad = \mathop{\arg\min}_{\{\mathbf{T}_i \in \text{SE}(3):1\leq i\leq N\}} \sum_{i,j} c_{i,j} \left\| \mathbf{T}_j - \mathbf{T}_i\mathbf{T}_{i,j} \right\|_F^2 .
\end{aligned}$$

The solution in [89] involves calculating an eigendecomposition directly. Let $\mathbf{A}^{\text{rot}}$ denote the $3N \times 3N$ matrix obtained from $\mathbf{A}$ by taking the top 3×3 elements from each sub-block of A. In the notation of [89, supplementary Sec. 2], our $\mathbf{A}^{\text{rot}}$ is equal to their "$L/2 + D$". Each of the $3N$ eigenvectors of $\mathbf{A}^{\text{rot}}$ (suitably padded with zeros to increase their length from $3N$ to $4N$, *e.g.*

$$[x_1, x_2, x_3, \dots, x_{3N-2}, x_{3N-1}, x_{3N}] \to [x_1, x_2, x_3, 0, \dots, x_{3N-2}, x_{3N-1}, x_{3N}, 0]),$$

becomes an eigenvectors of $\mathbf{A}$. Three of these eigenvectors with largest eigenvalues, projected onto $SO(3)$, solve [89, Eq. 5],

$$\mathop{\arg\min}_{\{R_i \in SO(3):1\leq i\leq N\}} \sum_{i,j} c_{i,j} \|R_j - R_i \times (\mathbf{T}_{i,j})_{1:3,1:3}\|_F^2$$

Rather than computing the eigendecomposition of $\mathbf{A}$ directly, we instead use power iteration. Raising $\mathbf{A}$ to large powers filters out the effect of the smaller eigenvalues. To do this efficiently, starting from $\mathbf{A}$, we repeatedly takes squares to calculate $\mathbf{A}^2$, then $\mathbf{A}^4$, and

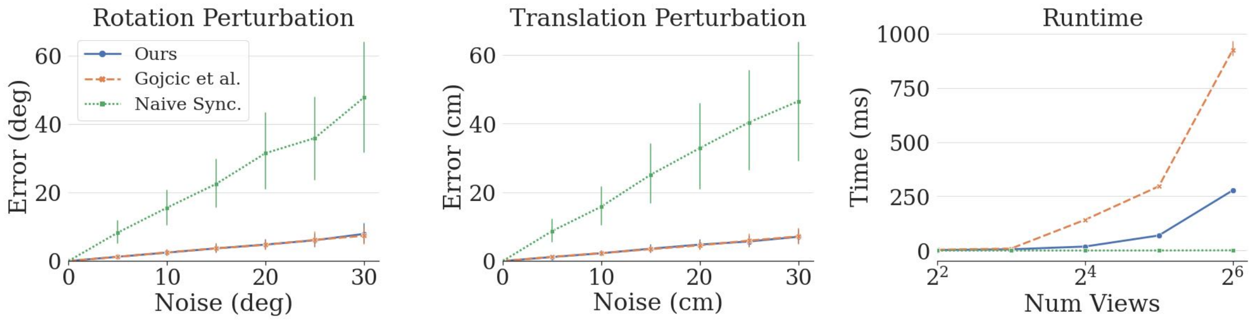


Figure A.1: **Synchronization Benchmark.** Our approach achieves the same error as Gojcic et al. [89], while being faster and more numerically stable.

so on until $\mathbf{A}^{2^t}$. Each element in $\mathbf{A}^{2^t}$ is then projected into SE(3) using SVD as described above; call the resulting matrix $\mathbb{A}$. $\mathbb{A}$ is composed of N 'rows', each with shape $4 \times 4N$; each of these rows is an approximate solution to the synchronization problem. The difference between the rows is that each row is centered around a different camera. We choose $t = O(\log N)$ so $2^t > N$; the the number of FLOPs needed to calculate $\mathbb{A}$ is thus O$(N^3 \log N)$. In practice, the time spent on synchronization is small compared to feature extraction and matching, as synchronization is independent of the image resolution. For much larger N, a database of key-frames could be used to reduce the size of N.

Numerical stability. Empirically, we find that training using synchronizations extracted from $\mathbf{A}^{2^t}$ was stable. Using an eigensolver to implement the method of [89] led to exploding gradients. The derivative with respect to a set of eigenvectors is unstable when the eigenvalues a clustered together, as is normally the case with the largest eigenvalues of $\mathbf{A}^{\text{rot}}$; when the pairwise rotations are compatible, the largest eigenvalues will be approximately equal.

Performance. We compare our synchronization approach to naive synchronization, which aggregates the transformations using adjacent views, and the eigendecomposition approach proposed by Gojcic et al. [89]. We compare the three algorithms on their ability to handle rotation and translation perturbation in the pairwise estimates as well as their runtime. As seen in Fig. A.1, our approach achieves the same performance as the eigendecomposition approach while being faster. Both approaches greatly naive synchronization since they are able to use information from all pairs. Furthermore, since our approach only relies on power iteration, it does not suffer from the numerical instability in the backward gradient discussed above.

Appendix B

Language-Guided Learning Experimental Setup

B.1 Evaluation Tasks

We compare all models by evaluating the encoder's frozen features on two downstream classification tasks: linear probe and few-shot. We chose to use simple classifiers since they allow us either to evaluate the features as is with non-parametric methods or to perform a comprehensive hyperparameter sweep (just one hyperparameter for logistic regression) to ensure a fair comparison. We explain the two evaluation setups below. The code has been made publicly available to simplify these comparisons in the future.

Linear Probe Classification

We follow the linear probe evaluation proposed by Kornblith et al. [144]: train a single classification layer using the L-BFGS optimizer [170]. We follow prior work [144, 219] and perform a hyperparameter sweep over the cost values in the logistic regression loss. We sweep over 96 values in log space from 10^{-6} to 10^{6}. During the hyper-parameter sweep, we train on the train split and evaluate on the valid split. We choose the cost value with the best validation performance and train a final classifier on the combined train and valid instances. We use the PyTorch [211] implementation of L-BFGS with all the default parameters except for the maximum number of iterations, which is set to 1000 similar to CLIP [219]. Our evaluation metric depends on the dataset, as shown in Tab. B.1, to account for class imbalance.

Few-Shot Classification

We also propose using fewshot classification as an evaluation for frozen features. Previous work [281, 303] has shown that simple classifiers on top of frozen features are strong baselines for fewshot classification. More specifically, Wang et al. [303] shows that when features are normalized (mean subtraction and L2 normalization), a nearest neighbor classifier is a very effective and strong baseline. Inspired by those results, we use simple weighted nearest neighbor classifiers as an evaluation for pre-trained frozen features.

Table B.1: **Evaluation Datasets.** Orange rows indicate datasets that do not have an official validation split; we constructed one by randomly holding out 20% of the official train split. Blue rows indicate datasets that do not officially define splits; we randomly sample instances to construct non-overlapping splits.

Dataset	Classes	Train	Val	Test	Metric
Food-101 [25]	101	60600	15150	25250	accuracy
CIFAR-10 [146]	10	40000	10000	10000	accuracy
CIFAR-100 [146]	100	40000	10000	10000	accuracy
CUB-2011 [307]	200	5795	1199	5794	accuracy
SUN397 [317]	397	15880	3970	19849	accuracy
Stanford Cars [145]	196	6515	1629	8041	accuracy
FGVC Aircraft [184]	100	3334	3333	3333	mean-per-cls
DTD [53]	47	1880	1880	1880	accuracy
Oxford-IIIT Pets [210]	37	2944	736	3669	mean-per-cls
Caltech-101 [81]	102	2448	612	6084	mean-per-cls
Oxford Flowers [203]	102	1020	1020	6149	mean-per-cls
STL-10 [54]	10	4000	1000	8000	accuracy
EuroSAT [106]	10	5000	5000	5000	accuracy
RESISC45 [47]	45	3150	3150	25200	accuracy
Patch Camelyon [296]	2	262144	32768	32768	accuracy

We set k to be the size of the support set and classify the features as follows:

$$y' = \arg\max_{v} \sum_{(I,y)\in\mathbb{D}_{\text{support}}} \mathbb{1}_{[v=y]}\text{sim}(f(I'), f(I)) \tag{B.1}$$

where $\mathbb{1}_{[v=y]}$ is an indicator variable that is 1 if y is the same class as v and 0 other wise, $\text{sim}(\cdot,\cdot)$ is cosine similarity between two vectors, f is the visual encoder, I' is the target image, $\mathbb{D}_{\text{support}}$ is the support set.

We adopt 5-way, 5-shot classification as our fewshot classification task. For each episode, we sample five random classes and then sample five images for each class for the training set, resulting in 25 labeled training images. We also sample 5 images for each class from the test set as our test images. For classes that have less than five test images, we use all available test images for that class. This is primarily an issue for Caltech-101 [81]. We sample 5000 episodes and compute the average test accuracy across all episodes. We experimented with increasing the number of episodes to 50000 to improve evaluation but noticed little change in the mean performance.

B.2 Evaluation Datasets

For evaluation, we use the datasets shown in Tab. B.1. We make use of TensorFlow datasets for evaluation and easy of replication [277]. For all datasets, we preprocess the images by resizing the image so that its smaller dimension is 224 using bilinear interpolation followed by a center crop to 224×224. We use bilinear interpolation since it can improve performance on low-resolution datasets such as CIFAR-10 and CIFAR-100. We normalize the images using ImageNet's mean and standard deviation for pixel values for all models except for pre-trained CLIP. For CLIP, we use their provided mean and standard deviation values as they have a large impact on performance: an average gain of $\tilde{4}\%$ for linear probes. We exclude Patch Camelyon from the fewshot evaluation since we do 5-shot, 5-way classification, and Patch Camelyon is a binary classification dataset.

B.3 Baselines

For fair evaluation, we retrained previous methods from scratch with several methods reimplemented. We also provided several system-level comparisons using pretrained checkpoints provided by prior work. Below, we provide additional details on our baselines.

B.3.1 Pre-trained Checkpoints

We use pretrained checkpoints for both sampling and as a basis for comparison. We list the source of the checkpoints below.

- **SBERT:** We use two checkpoints from SBERT [225].[1] The first checkpoint is `all-mpnet-base-v2` which uses the MPNet [262] backbone. The second second checkpoints is `all-MiniLM-L12-v2` uses the MiniLM [301] backbone.
- **CLIP:** We use the official checkpoints provided by CLIP for both system-level comparisons and sampling.[2] We use the `RN50` checkpoint in the system-level comparisons to match the backbone for other models. We use the `ViT-B/32` checkpoint for sampling to provide the strongest performance for visual sampling in evaluating different modalities for sampling.
- **ImageNet pretrained model:** We use the checkpoints provided by torchvision[3] for all ImageNet pre-trained models. For system-level comparisons, we use the ResNet-50 with

[1] https://www.sbert.net/

[2] https://github.com/openai/CLIP

[3] https://pytorch.org/vision/stable/models.html

IMAGENET1K_V2 checkpoint as it achieves better performance than the original ResNet-50 checkpoint. For consistency, we use the ViT-base-32 model for sampling to match the CLIP sampling strategy and provide a strong baseline for utility of language as a sampling modality.

- **SimCLR:** We use the SimCLR checkpoint provided by PyTorch Lightning Bolts.[4] While SimCLR released some checkpoints for TensorFlow, we found that converting them to PyTorch using the recommended tools resulted in lower performance. We use the same checkpoint for both sampling and system-level comparison. SimCLR only released models trained for 800 epochs.
- **SimSiam:** We use the official checkpoint released by SimSiam.[5] We use the checkpoint trained with a batch size of 512 as it more closely matches our training setup.
- **MoCo:** We use the official checkpoint for MoCo v3.[6] We use the checkpoint for the model trained for 100 epochs to match other checkpoints more closely.
- **SwAV:** We use the official SwAV checkpoint.[7] We use the checkpoint trained for 100 epochs to match the training duration of other methods. Unlike our implementation, the method is *full* SwAV which includes the Multi-Crop strategy.

B.3.2 Retrained models

We re-implement and retrain all baselines. When an official implementation was available, we adapted their code to fit within our pipeline.

For all models, we use a ResNet-50 backbone from torchvision with random initialization and a feature dimension of 2048 (the fc layer is removed). For projection layers, we either use a linear layer or a multi-layer perceptron (MLP) where every layer except for the last is followed by batch normalization and a ReLU non-linearity. We describe an N-layer MLP with $N+1$ numbers depicting the input dimension for the first layer followed by the output dimension for all layers.

We use two forms of augmentation: SimCLR or global crop. Global crop consists of a random resized square crop with a scale of (0.5, 1.0) to an image size of 224×224. SimCLR augmentations consist of random resized square crop, color jittering, random gray scale, random horizontal flipping, and Gaussian blur. We use the same augmentation parameters as prior work [41, 44]. All images are normalized using ImageNet's mean and standard

[4] https://lightning-bolts.readthedocs.io/

[5] https://github.com/facebookresearch/simsiam

[6] https://github.com/facebookresearch/moco-v3

[7] https://github.com/facebookresearch/swav

deviation statistics.

Below we provide some details for each of the baseline methods:

- **SimCLR:** We use the backbone with a 3-layer MLP as a projection layer with feature dimensions (2048, 2048, 2048, 128) similar to the original paper. We use the SimCLR loss implementation from Mu et al. [198]. We adapt it to the distributed setting using synchronized batch norm, as well as synchronizing the features and gradients for the loss. We use SimCLR augmentations for SimCLR and global crop augmentations for LGSimCLR. We experimented with mixing SimCLR augmentation and language sampled pairs and found that it results in slightly inferior performance: adding augmentations reduces the linear probe average accuracy from 78.3 to 77.9.

- **CLIP:** We use the backbone with a linear projection layer to a feature dimension of 512. We use the smallest CLIP language encoder, similar to SLIP [198], with a feature dimension of 512 and a linear language projection layer. We use the loss implementation from SLIP [198], but adapt it to share the loss gradients similar to the SimCLR loss. We use global crop augmentation for CLIP since SLIP [198] reported that it performs better than CLIP's original center crop preprocessing.

- **SLIP:** We follow SLIP's implementation and combine the augmentations, projections, and losses from SimCLR and CLIP. We use the same language transformer as our CLIP implementation. We generate two augmented views with SimCLR augmentation for the SimCLR loss and one with global cropping for the CLIP loss. Those views are passed through their respective projections (3-layer MLP for SimCLR and linear projection for CLIP) and losses. For LGSLIP, we only use the global crop augmentation, resulting in only 2 augmented views. We apply the SimCLR loss between the language-sampled image pair and the CLIP loss between only one of the images and its caption.

- **SimSiam:** We follow the original SimSiam implementation and use a 3-layer MLP as our projection head (2048, 2048, 2048, 2048) and a 2 layer MLP as our prediction head (2048, 512, 2048). We use the loss formulation from the original paper. For LGSimSiam, we use the same formulation but use global crop instead of SimCLR augmentations.

- **SwAV.** We follow the original SwaV implementation and use a 2 layer MLP (2048, 2048, 128) as our projection head and a linear layer as our prototype head with an output dimension of 3000. The prototypes are frozen for the first epoch. We use the distributed Sinkhorn clustering implementation from the official code release.

- **NNCLR.** We rely on the implementation of NNCLR provided by Lightly [273] since NNCLR [71] did not release an official implementation. Specifically, we use the memory bank implementation from Lightly and reimplement NNCLR. We find that while our NNCLR implementation outperforms SimCLR on ImageNet, it under-performs on RedCaps. We use a 3 layer MLP (2048, 2048, 2048, 256) as our projection head and a 2 layer MLP (256, 4098, 256) as our prediction head. We also use a queue of length 16384 (equivalent to 32 batches) for the memory bank. For Language NNCLR, we use the memory bank from Lightly as well, similar to DeCLIP [166]. We adapt the CLIP implementation listed above with a memory bank for the language encoder. We use a weighting of 0.8 for the CLIP loss and 0.2 for the language NNS loss, similar to DeCLIP [166].

B.4 Complete Results

For clarity, we have chosen to provide averaged results in Chapter 5. However, we provide the full list of results here for the sake of completeness. We also report the complete performance breakdown for all methods on linear probe in Tab. B.2 and fewshot classification Tab. B.3. For fewshot classification, we also report the 95% confidence interval as a subscript.

Table B.2: **Linear Probe Evaluations.** We train models on RedCaps and report performance of logistic regression using frozen features on 15 downstream tasks. Models are split based on whether or not they require captions for training. The results show a strong performance gain for language-guided sampling over previous approaches with strong performance gains for fine-grained classification datasets.

Model	**Dataset**	**Sampling Space**	**Food-101**	**CIFAR-10**	**CIFAR-100**	**CUB**	**SUN397**	**Cars**	**Aircraft**	**DTD**	**Pets**	**Caltech-101**	**Flowers**	**STL-10**	**EuroSAT**	**RESISC45**	**PCAM**	**Avg.**
Pre-trained Checkpoints																		
Supervised [309]	ImageNet	-	71.0	93.2	77.0	68.4	63.0	48.7	41.0	73.0	92.6	91.9	88.3	98.2	95.8	85.3	82.4	78.0
SimSiam [42]	ImageNet	-	70.6	92.2	74.9	47.1	0.3	52.4	52.6	74.4	83.3	89.3	91.8	95.9	96.7	86.4	85.2	72.9
MoCo v3 [44]	ImageNet	-	71.4	93.3	77.9	51.5	60.4	52.4	53.0	73.6	85.8	90.4	92.1	96.8	96.3	85.0	85.0	77.7
SwAV [32]	ImageNet	-	72.8	93.0	77.5	48.8	63.2	55.5	52.7	77.2	84.5	89.9	93.4	97.2	96.7	86.9	83.7	78.2
SimCLR [41]	ImageNet	-	71.4	91.3	73.9	44.3	60.3	44.6	46.7	74.9	83.9	87.4	90.2	96.2	95.9	84.4	85.1	75.4
CLIP [219]	CLIP (400M)	-	86.4	88.7	70.2	69.8	72.5	78.4	49.4	76.3	88.0	88.9	96.1	97.2	94.7	87.9	82.7	81.8
RedCaps-trained Baselines																		
SwAV	RedCaps	-	63.6	81.3	57.5	21.6	47.5	22.9	35.4	68.1	61.1	70.5	78.0	87.7	94.3	79.9	84.3	63.6
SimSiam	RedCaps	-	64.1	79.9	56.1	28.2	48.3	29.5	41.2	66.2	69.1	73.6	83.6	85.7	94.4	82.1	83.3	65.7
SimCLR	RedCaps	-	69.0	82.9	61.6	30.6	52.6	33.7	43.7	69.8	70.5	74.1	86.9	88.0	95.4	84.6	84.4	68.5
Visual NNCLR	RedCaps	-	65.4	82.8	60.2	26.6	50.0	26.6	40.9	68.0	65.2	75.4	83.5	88.5	95.3	82.2	83.8	66.3
CLIP	RedCaps	-	80.9	84.7	62.7	50.4	57.4	45.8	36.7	67.6	79.8	84.0	91.0	93.5	93.9	82.2	82.6	72.9
CLIP (SBERT Encoder)	RedCaps	-	80.5	81.3	59.4	50.6	56.9	45.9	35.7	69.1	76.7	81.7	90.2	93.6	92.9	81.1	81.3	71.8
Language NNCLR	RedCaps	-	81.2	83.1	61.9	48.6	56.5	45.1	37.2	68.8	78.1	82.0	90.2	93.4	92.5	81.1	80.7	72.0
SLIP	RedCaps	-	77.7	87.2	67.0	42.4	58.1	48.7	45.2	72.3	79.5	82.7	92.1	92.7	95.6	85.5	83.4	74.0
Sampling Space - Language																		
LGSimCLR	RedCaps	SBERT (MiniLM)	83.2	88.0	69.3	60.4	59.7	64.0	54.0	72.7	82.6	88.5	95.7	94.1	96.4	88.1	82.2	78.6
LGSimCLR	RedCaps	CLIP (400M)	83.3	87.6	68.9	60.1	59.9	62.9	53.7	70.5	82.6	88.7	95.6	94.3	96.2	88.2	82.0	78.3
LGSimCLR	RedCaps	CLIP (RedCaps)	83.7	88.0	67.8	59.6	60.7	60.8	53.7	71.4	82.4	89.1	95.9	93.8	96.1	88.4	82.7	78.3
LGSimCLR	RedCaps	FastText BoW	80.8	85.5	66.7	54.2	58.7	56.6	51.1	69.9	78.3	88.0	94.4	92.5	96.2	87.7	81.5	76.1
Sampling Space - Visual																		
LGSimCLR	RedCaps	ImageNet Supervised	75.7	92.2	75.4	57.5	60.2	53.7	52.2	71.7	90.3	90.2	93.1	95.5	96.8	87.7	83.0	78.3
LGSimCLR	RedCaps	SimCLR	71.4	87.0	67.5	36.8	57.9	41.8	46.3	74.2	82.8	82.6	90.7	93.4	95.7	85.2	83.2	73.1
LGSimCLR	RedCaps	CLIP (400M)	83.6	90.7	72.1	58.3	62.5	59.2	51.3	75.5	88.7	90.3	95.2	95.4	96.2	88.6	82.9	79.4
Sampling Scope																		
LGSimCLR	RedCaps	SBERT - Year	82.6	85.8	66.1	58.1	59.0	57.1	52.8	71.9	80.7	88.1	95.6	93.1	96.1	88.0	82.8	77.2
LGSimCLR	RedCaps	SBERT - Sub-Year	82.4	86.8	66.9	56.4	59.5	54.5	51.4	72.1	89.0	89.8	94.9	95.2	95.8	88.1	81.9	77.6
LGSimCLR	RedCaps	SBERT - Sub	83.2	88.7	69.5	60.4	59.9	60.0	53.0	72.4	89.7	90.3	96.2	94.8	96.2	88.4	82.2	79.0
Pre-training Datasets																		
LGSimCLR	CC3M	SBERT (MPNet)	64.4	84.6	65.4	44.4	59.1	41.9	46.8	66.0	70.7	83.9	91.2	91.6	95.6	86.1	80.5	71.5
LGSimCLR	CC12M	SBERT (MPNet)	73.4	88.6	70.1	50.4	66.0	58.7	52.4	72.6	79.0	88.3	92.6	94.5	95.6	87.5	81.6	76.8
LGSimCLR	RC-20	SBERT (MPNet)	77.8	84.3	64.5	53.9	53.9	51.7	48.1	66.4	76.2	83.9	93.9	89.9	95.4	86.4	81.4	73.8
Batch Size Scaling																		
SimCLR (256)	RedCaps	-	67.1	83.1	60.5	28.5	51.0	32.5	42.4	70.0	68.3	73.9	85.8	86.5	96.2	84.2	83.1	67.5
SimCLR (1024)	RedCaps	-	70.0	84.4	62.8	31.9	52.4	35.8	44.2	70.9	72.7	74.7	87.9	88.3	95.6	84.8	83.3	69.3
SimCLR (2048)	RedCaps	-	70.4	83.9	62.6	32.5	53.3	36.7	44.9	70.9	73.1	75.5	88.1	88.8	96.5	85.1	84.2	69.8
LGSimCLR (256)	RedCaps	SBERT (MPNet)	82.9	87.3	68.0	58.7	60.2	58.2	52.6	73.2	81.1	88.2	95.2	94.0	96.1	87.7	81.8	77.7
LGSimCLR (1024)	RedCaps	SBERT (MPNet)	83.7	87.1	68.1	62.0	60.7	63.4	53.7	73.4	80.8	89.8	95.7	93.7	95.7	88.3	82.6	78.6
LGSimCLR (2048)	RedCaps	SBERT (MPNet)	84.2	88.2	69.1	63.2	60.9	65.2	55.5	71.4	81.7	89.7	96.0	94.4	96.3	88.5	82.1	79.1
Alternative Formulations																		
LGSimCLR	RedCaps	SBERT (MPNet)	83.2	87.8	69.0	59.3	60.3	62.3	53.4	71.2	81.8	89.4	95.9	94.0	95.6	88.0	81.1	78.2
LGSimSiam	RedCaps	SBERT (MPNet)	73.8	83.4	62.6	40.6	54.6	41.1	47.3	68.6	66.5	85.2	90.3	90.8	95.7	85.6	81.3	71.2
LGSLIP	RedCaps	SBERT (MPNet)	84.5	87.4	69.2	60.7	62.3	62.2	52.5	73.1	83.1	90.2	96.3	94.8	95.3	88.4	82.7	78.8

Table B.3: **Few-Shot Evaluations.** We train models on RedCaps and report the performance of 5-way, 5-shot classification using frozen features on 14 downstream tasks. Models are split based on whether or not they require captions for training.

Model	Dataset	Sampling Space	Food-101	CIFAR-10	CIFAR-100	CUB	SUN397	Cars	Aircraft	DTD	Pets	Caltech-101	Flowers	STL-10	EuroSAT	RESISC45	Avg.
Pre-trained Checkpoints																	
Supervised [309]	ImageNet	-	$81.6_{0.3}$	$84.1_{0.2}$	$87.8_{0.2}$	$91.9_{0.2}$	$95.0_{0.1}$	$75.7_{0.3}$	$53.1_{0.4}$	$80.4_{0.3}$	$97.6_{0.1}$	$97.4_{0.1}$	$91.4_{0.2}$	$95.2_{0.1}$	$83.6_{0.2}$	$85.2_{0.3}$	85.7
SimSiam [42]	ImageNet	-	$70.5_{0.3}$	$77.5_{0.3}$	$82.0_{0.3}$	$67.4_{0.4}$	$92.1_{0.2}$	$51.9_{0.3}$	$43.7_{0.4}$	$81.8_{0.3}$	$86.7_{0.3}$	$94.9_{0.2}$	$93.7_{0.2}$	$89.7_{0.2}$	$87.7_{0.2}$	$82.2_{0.3}$	78.7
MoCo v3 [44]	ImageNet	-	$72.7_{0.3}$	$82.3_{0.2}$	$84.9_{0.3}$	$74.7_{0.3}$	$92.5_{0.2}$	$52.0_{0.4}$	$42.4_{0.4}$	$80.7_{0.3}$	$89.1_{0.2}$	$95.8_{0.1}$	$93.6_{0.2}$	$91.5_{0.2}$	$86.3_{0.2}$	$83.0_{0.3}$	80.1
SwAV [32]	ImageNet	-	$68.3_{0.3}$	$78.1_{0.3}$	$82.1_{0.3}$	$65.4_{0.4}$	$93.7_{0.2}$	$52.7_{0.4}$	$40.3_{0.4}$	$83.8_{0.2}$	$83.8_{0.3}$	$94.5_{0.2}$	$93.4_{0.2}$	$91.2_{0.2}$	$88.0_{0.2}$	$83.8_{0.3}$	78.5
SimCLR [41]	ImageNet	-	$70.0_{0.3}$	$76.9_{0.3}$	$80.9_{0.3}$	$67.5_{0.4}$	$92.5_{0.2}$	$51.9_{0.3}$	$42.1_{0.4}$	$82.2_{0.3}$	$85.0_{0.3}$	$93.0_{0.2}$	$90.3_{0.2}$	$88.8_{0.2}$	$83.6_{0.3}$	$78.5_{0.3}$	77.4
CLIP [219]	Web400M	-	$92.1_{0.2}$	$76.3_{0.3}$	$79.2_{0.3}$	$92.9_{0.2}$	$96.9_{0.1}$	$93.3_{0.2}$	$73.2_{0.4}$	$81.8_{0.3}$	$86.1_{0.3}$	$95.9_{0.1}$	$97.9_{0.1}$	$95.6_{0.1}$	$77.5_{0.3}$	$90.5_{0.2}$	87.8
RedCaps-trained Baselines																	
SwAV	RedCaps	-	$64.5_{0.4}$	$54.0_{0.3}$	$61.8_{0.3}$	$45.8_{0.4}$	$84.9_{0.3}$	$36.5_{0.3}$	$34.1_{0.3}$	$74.8_{0.3}$	$66.5_{0.4}$	$78.1_{0.3}$	$75.5_{0.3}$	$72.6_{0.3}$	$80.4_{0.3}$	$72.9_{0.4}$	64.5
SimSiam	RedCaps	-	$63.9_{0.3}$	$49.9_{0.3}$	$57.2_{0.3}$	$49.5_{0.4}$	$84.5_{0.3}$	$39.3_{0.3}$	$37.9_{0.3}$	$75.7_{0.3}$	$67.8_{0.4}$	$79.7_{0.3}$	$81.5_{0.3}$	$69.6_{0.3}$	$80.6_{0.3}$	$79.4_{0.3}$	65.5
SimCLR	RedCaps	-	$66.9_{0.3}$	$45.7_{0.3}$	$51.0_{0.3}$	$51.5_{0.4}$	$87.1_{0.2}$	$44.0_{0.3}$	$38.4_{0.3}$	$77.6_{0.3}$	$70.1_{0.3}$	$80.0_{0.3}$	$86.9_{0.2}$	$69.6_{0.3}$	$83.5_{0.3}$	$81.3_{0.3}$	66.7
Visual NNCLR	RedCaps	-	$65.6_{0.3}$	$54.1_{0.3}$	$61.7_{0.3}$	$45.8_{0.3}$	$85.3_{0.3}$	$37.9_{0.3}$	$34.9_{0.3}$	$75.2_{0.3}$	$67.3_{0.4}$	$81.1_{0.3}$	$75.4_{0.3}$	$74.3_{0.3}$	$83.6_{0.3}$	$76.7_{0.3}$	65.6
CLIP	RedCaps	-	$88.9_{0.2}$	$64.6_{0.3}$	$73.1_{0.3}$	$78.3_{0.3}$	$90.9_{0.2}$	$69.7_{0.3}$	$40.7_{0.3}$	$75.7_{0.3}$	$77.5_{0.3}$	$91.6_{0.2}$	$94.7_{0.2}$	$89.8_{0.2}$	$75.3_{0.3}$	$74.8_{0.3}$	77.5
CLIP (SBERT)	RedCaps	-	$89.9_{0.2}$	$59.9_{0.3}$	$67.9_{0.3}$	$83.2_{0.3}$	$91.1_{0.2}$	$70.2_{0.3}$	$41.0_{0.3}$	$75.0_{0.3}$	$79.4_{0.3}$	$91.2_{0.2}$	$94.5_{0.2}$	$89.4_{0.2}$	$72.3_{0.3}$	$74.9_{0.3}$	77.1
Language NNCLR	RedCaps	-	$89.3_{0.2}$	$65.3_{0.3}$	$73.4_{0.3}$	$78.6_{0.3}$	$90.8_{0.2}$	$68.4_{0.3}$	$40.4_{0.3}$	$75.2_{0.3}$	$78.8_{0.3}$	$90.9_{0.2}$	$94.3_{0.2}$	$89.6_{0.2}$	$75.2_{0.3}$	$71.9_{0.3}$	77.3
SLIP	RedCaps	-	$81.5_{0.3}$	$63.5_{0.3}$	$70.8_{0.3}$	$63.1_{0.4}$	$91.3_{0.2}$	$62.9_{0.3}$	$42.1_{0.4}$	$79.6_{0.3}$	$76.4_{0.3}$	$88.4_{0.2}$	$92.2_{0.2}$	$83.4_{0.2}$	$82.7_{0.3}$	$80.8_{0.3}$	75.6
Sampling Space - Language																	
LGSimCLR	RedCaps	SBERT (MiniLM)	$90.4_{0.2}$	$67.1_{0.3}$	$76.7_{0.3}$	$83.9_{0.3}$	$92.7_{0.2}$	$79.2_{0.3}$	$52.1_{0.4}$	$81.2_{0.3}$	$86.2_{0.3}$	$95.5_{0.1}$	$97.6_{0.1}$	$87.4_{0.2}$	$86.9_{0.2}$	$89.0_{0.2}$	83.3
LGSimCLR	RedCaps	CLIP (400M)	$90.7_{0.2}$	$65.8_{0.3}$	$75.6_{0.3}$	$83.8_{0.3}$	$92.8_{0.2}$	$80.9_{0.3}$	$52.0_{0.4}$	$81.4_{0.3}$	$85.6_{0.3}$	$95.5_{0.1}$	$97.5_{0.1}$	$87.3_{0.2}$	$84.8_{0.2}$	$89.3_{0.2}$	83.1
LGSimCLR	RedCaps	CLIP (RedCaps)	$90.4_{0.2}$	$64.8_{0.3}$	$75.3_{0.3}$	$82.2_{0.3}$	$92.8_{0.2}$	$76.6_{0.3}$	$50.4_{0.4}$	$81.3_{0.3}$	$84.6_{0.3}$	$95.2_{0.2}$	$97.7_{0.1}$	$86.9_{0.2}$	$86.5_{0.2}$	$89.1_{0.2}$	82.4
LGSimCLR	RedCaps	FastText BoW	$88.4_{0.2}$	$62.1_{0.3}$	$73.7_{0.3}$	$79.3_{0.3}$	$92.2_{0.2}$	$74.0_{0.3}$	$52.5_{0.4}$	$79.4_{0.3}$	$82.7_{0.3}$	$94.3_{0.2}$	$97.5_{0.1}$	$83.0_{0.2}$	$85.4_{0.2}$	$88.5_{0.2}$	80.9
Sampling Space - Visual																	
LGSimCLR	RedCaps	ImageNet Supervised	$79.6_{0.3}$	$75.6_{0.3}$	$83.0_{0.3}$	$76.6_{0.3}$	$92.5_{0.2}$	$64.6_{0.4}$	$46.1_{0.4}$	$80.7_{0.3}$	$94.3_{0.2}$	$96.3_{0.1}$	$94.8_{0.2}$	$87.4_{0.2}$	$86.4_{0.2}$	$87.3_{0.2}$	81.8
LGSimCLR	RedCaps	SimCLR	$72.0_{0.3}$	$62.9_{0.3}$	$71.9_{0.3}$	$58.9_{0.4}$	$90.8_{0.2}$	$51.3_{0.3}$	$38.7_{0.3}$	$81.8_{0.3}$	$86.4_{0.3}$	$91.3_{0.2}$	$90.7_{0.2}$	$83.8_{0.2}$	$85.5_{0.2}$	$78.6_{0.3}$	74.6
LGSimCLR	RedCaps	CLIP (400M)	$88.8_{0.2}$	$72.5_{0.3}$	$79.7_{0.3}$	$77.6_{0.3}$	$93.1_{0.2}$	$73.3_{0.3}$	$45.6_{0.4}$	$82.2_{0.3}$	$90.9_{0.2}$	$94.6_{0.2}$	$96.3_{0.1}$	$89.2_{0.2}$	$84.6_{0.2}$	$87.4_{0.2}$	82.6
Sampling Scope																	
LGSimCLR	RedCaps	SBERT - Year	$89.7_{0.2}$	$61.2_{0.3}$	$72.2_{0.3}$	$81.2_{0.3}$	$92.0_{0.2}$	$71.9_{0.3}$	$48.4_{0.4}$	$80.1_{0.3}$	$79.3_{0.3}$	$93.8_{0.2}$	$97.4_{0.1}$	$84.5_{0.2}$	$84.0_{0.2}$	$87.8_{0.2}$	80.2
LGSimCLR	RedCaps	SBERT - Sub-Year	$88.0_{0.2}$	$59.0_{0.3}$	$66.6_{0.3}$	$76.5_{0.3}$	$90.6_{0.2}$	$63.6_{0.4}$	$45.2_{0.3}$	$78.5_{0.3}$	$80.7_{0.3}$	$94.8_{0.2}$	$97.4_{0.1}$	$83.2_{0.2}$	$82.2_{0.3}$	$88.7_{0.2}$	78.2
LGSimCLR	RedCaps	SBERT - Sub	$88.7_{0.2}$	$70.4_{0.3}$	$78.4_{0.3}$	$75.3_{0.3}$	$90.7_{0.2}$	$67.3_{0.3}$	$47.6_{0.4}$	$78.3_{0.3}$	$76.6_{0.3}$	$95.4_{0.1}$	$97.8_{0.1}$	$86.5_{0.2}$	$85.3_{0.2}$	$89.1_{0.2}$	80.5
Pre-training Datasets																	
LGSimCLR	CC3M	SBERT (MPNet)	$69.2_{0.3}$	$60.6_{0.3}$	$71.7_{0.3}$	$72.0_{0.3}$	$92.8_{0.2}$	$58.8_{0.3}$	$48.9_{0.4}$	$77.4_{0.3}$	$77.8_{0.3}$	$92.9_{0.2}$	$95.0_{0.2}$	$83.0_{0.3}$	$82.4_{0.3}$	$86.3_{0.3}$	76.3
LGSimCLR	CC12M	SBERT (MPNet)	$79.6_{0.3}$	$72.0_{0.3}$	$78.5_{0.3}$	$71.2_{0.3}$	$95.2_{0.1}$	$78.2_{0.3}$	$55.5_{0.4}$	$81.8_{0.3}$	$82.1_{0.3}$	$96.2_{0.1}$	$95.3_{0.1}$	$90.0_{0.2}$	$83.6_{0.3}$	$88.0_{0.2}$	81.9
LGSimCLR	RC-20	SBERT (MPNet)	$86.0_{0.2}$	$60.3_{0.3}$	$70.5_{0.3}$	$79.9_{0.3}$	$90.1_{0.2}$	$69.8_{0.3}$	$48.6_{0.4}$	$77.0_{0.3}$	$81.0_{0.3}$	$92.3_{0.2}$	$96.8_{0.1}$	$78.8_{0.3}$	$84.7_{0.2}$	$87.0_{0.2}$	78.8
Batch Size Scaling																	
SimCLR (256)	RedCaps	-	$64.9_{0.3}$	$52.7_{0.3}$	$57.9_{0.3}$	$51.4_{0.4}$	$86.6_{0.2}$	$43.6_{0.3}$	$38.3_{0.3}$	$77.1_{0.3}$	$68.7_{0.3}$	$79.2_{0.3}$	$85.9_{0.3}$	$69.7_{0.3}$	$84.1_{0.3}$	$81.3_{0.3}$	67.2
SimCLR (1024)	RedCaps	-	$67.5_{0.3}$	$54.0_{0.3}$	$59.2_{0.3}$	$53.0_{0.4}$	$87.2_{0.2}$	$44.7_{0.3}$	$38.9_{0.3}$	$77.9_{0.3}$	$71.5_{0.3}$	$80.4_{0.3}$	$87.9_{0.2}$	$71.8_{0.3}$	$82.5_{0.3}$	$81.8_{0.3}$	68.4
SimCLR (2048)	RedCaps	-	$68.4_{0.3}$	$51.8_{0.3}$	$57.8_{0.3}$	$53.3_{0.4}$	$87.3_{0.2}$	$45.4_{0.3}$	$38.8_{0.3}$	$77.8_{0.3}$	$73.2_{0.3}$	$81.6_{0.3}$	$87.9_{0.2}$	$71.6_{0.3}$	$84.0_{0.3}$	$81.6_{0.3}$	68.6
LGSimCLR (256)	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$66.6_{0.3}$	$75.8_{0.3}$	$81.6_{0.3}$	$92.6_{0.2}$	$75.3_{0.3}$	$50.5_{0.4}$	$81.6_{0.3}$	$83.2_{0.3}$	$95.3_{0.1}$	$97.6_{0.1}$	$86.8_{0.2}$	$86.4_{0.2}$	$88.9_{0.2}$	82.3
LGSimCLR (1024)	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$64.9_{0.3}$	$75.7_{0.3}$	$83.8_{0.3}$	$92.6_{0.2}$	$78.2_{0.3}$	$52.6_{0.4}$	$80.9_{0.3}$	$83.6_{0.3}$	$95.6_{0.1}$	$97.6_{0.1}$	$86.7_{0.2}$	$86.0_{0.2}$	$88.6_{0.2}$	82.6
LGSimCLR (2048)	RedCaps	SBERT (MPNet)	$90.6_{0.2}$	$67.5_{0.3}$	$76.6_{0.3}$	$83.9_{0.3}$	$92.6_{0.2}$	$79.7_{0.3}$	$51.5_{0.4}$	$80.6_{0.3}$	$83.8_{0.3}$	$95.8_{0.1}$	$97.6_{0.1}$	$87.1_{0.2}$	$86.6_{0.2}$	$89.2_{0.2}$	83.1
Alternative Formulations																	
LGSimCLR	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$66.3_{0.3}$	$75.5_{0.3}$	$83.1_{0.3}$	$92.7_{0.2}$	$77.6_{0.3}$	$50.6_{0.4}$	$81.1_{0.3}$	$84.1_{0.3}$	$95.4_{0.1}$	$97.6_{0.1}$	$86.5_{0.2}$	$85.0_{0.2}$	$89.0_{0.2}$	82.5
LGSimSiam	RedCaps	SBERT (MPNet)	$81.2_{0.3}$	$61.6_{0.3}$	$71.2_{0.3}$	$63.1_{0.4}$	$90.2_{0.2}$	$60.9_{0.3}$	$44.6_{0.4}$	$78.8_{0.3}$	$68.0_{0.3}$	$92.8_{0.2}$	$93.7_{0.2}$	$81.2_{0.3}$	$85.1_{0.2}$	$86.7_{0.2}$	75.7
LGSLIP	RedCaps	SBERT (MPNet)	$91.3_{0.2}$	$67.2_{0.3}$	$77.2_{0.3}$	$81.8_{0.3}$	$92.6_{0.2}$	$77.3_{0.3}$	$50.4_{0.4}$	$81.8_{0.3}$	$81.8_{0.3}$	$96.1_{0.1}$	$97.8_{0.1}$	$89.2_{0.2}$	$85.3_{0.2}$	$89.1_{0.2}$	82.8

Appendix C

3D Awareness Experimental Setup

We only provide high-level details in the chapter and omit details to enhance readability. We provide more details regarding the experimental setup below and explain the rationale behind our design choices.

C.1 Pre-trained Vision Models

We consider 16 checkpoints that represent models trained with 6 different kinds of supervision. The models were chosen with two criteria in mind: (1) coverage of major approaches used for large-scale training, and (2) comparable model and training scale to allow for comparisons. We only use publicly available checkpoints to understand the 3D awareness of the models that are commonly used. We list all models considered below:

MAE. He et al. [103] proposed masked auto-encoding as a pre-training recipe for vision transformers. Such models are trained with a large masking ratio; *e.g.*, 75% of the input image patches are masked. In our experiments, we use the ViT-B/16 model trained on ImageNet-1k. We use the checkpoint[1] available on the Transformers library [312].

FCMAE. Fully-convolutional masked autoencoders (FCMAE) extend the MAE approach similarly trained to reconstruct images. However, unlike MAE, they use a ConvNeXtv2 [313] backbone instead of ViT. In our experiments, we use the ConvNeXtv2 base architecture which has a comparable model capability to the base visual transformer architectures. Following the analysis of Goldblum et al. [90], we use the model pre-trained on ImageNet-22k to allow a more comparable training data to other models. We use the checkpoint[2] available on the timm library [308].

[1] https://huggingface.co/facebook/vit-mae-base

[2] https://huggingface.co/timm/convnextv2_base.fcmae_ft_in22k_in1k_384

DINO. Caron et al. [33] proposed a self-distillation approach for model pre-training. The proposed approach trains a student network to generate features similar to a teacher network, where the teacher is an exponential moving average of the student network. At its core, this approach relies on instance discrimination as the model is trained to learn to generate similar embeddings for different crops of the same image instance. In our work, we use the ViT-B/16 architecture trained on ImageNet-1k. We use the checkpoint released by the authors.[3]

iBOT. Zhou et al. [352] combine ideas from DINO and MAE by training a model to reconstruct the masked dense features based on a teacher network. iBOT uses both an image-level and a dense distillation objective. In our work, we use the ViT-B/16 architecture trained on ImageNet-1k. We use the checkpoint released by the authors.[4]

DINOv2. Oquab et al. [206] scale up the hybrid approach proposed by Zhou et al. [352] while improving the training recipe and incorporating improved losses and regularizers. Furthermore, the training data and recipe are both scaled up in magnitude resulting in much better performance. This includes the collection of a large curated private dataset called LVD-142M, which is curated through the use of the clustered features of a pre-trained self-supervised model using several downstream datasets, including NYUv2 [257]. While DINOv2 was trained on ImageNet-22k, those weights are not publicly available. We discuss the impact of these curated datasets in Sec. 6.3. We also consider the new DINOv2+reg, which incorporates register tokens [59]. However, we find that it results in slightly worse performance than the classic DINOv2 model. Similar to other models, we use the base-model visual transformer. However, DINOv2 is trained with a smaller patch size of 14 instead of 16. We use the checkpoints released by the authors.[5]

DeiT III. Touvron et al. [287] propose an updated training recipe of supervised vision transformers that incorporate recent best practices from self-supervised learning. The result is a much stronger supervised transformer compared to previous training recipes. Similar to other approaches, we use the ViT-B/16 architecture trained on ImageNet-22k. We use the checkpoint released by the authors.[6]

[3] https://github.com/facebookresearch/dino

[4] https://github.com/bytedance/ibot

[5] https://github.com/facebookresearch/dinov2

[6] https://github.com/facebookresearch/deit

CLIP. Vision and language models are trained to generate aligned feature embeddings using a contrastive objective. The original CLIP family of models was proposed by Radford et al. [219] and included a large variety of architectures on a private dataset of 400M image-text pairs called WIT. More recently, Ilharco et al. [119] trained several CLIP models using several architectures trained on publicly available datasets. We consider four different CLIP models. First, we consider two ViT-B/16 models trained on WIT or LAION [245]. This provides us with both the original CLIP, which is widely used, as well as a variant trained with public datasets. We also consider two ConvNeXt-base [173] models trained with and without additional augmentations [269], which were trained on LAION as well. For all models, we use the checkpoints available through OpenCLIP [119].[7]

SigLIP. SigLIP modifies CLIP by replacing the contrastive objective with an instance-wise sigmoid loss. The sigmoid loss does not require the computation of all pairs across the batch, as it only relies on the image and text embedding. This simplifies the objective while enabling further scaling up of the batch size for training. The publicly available checkpoints were trained on WebLI, a private dataset. We use the checkpoint available through OpenCLIP [119].[7]

StableDiffusion. StableDiffusion [229] is trained using on text-conditioned image generation using a denoising objective. This family of models have achieved a remarkable generation performance. In our experiments, we use the text-conditioned checkpoint of StableDiffusion v2-1.[8] Following prior work [323, 341, 349], we extract features from the decoding blocks of the UNet. However, we deviate from prior work in two ways. First, some prior work [275, 340] computes features of the image with different sampled noise and then averages them. While this form of ensembling is unique to Diffusion-based models, it is possible to compute features based on image crops and similarly average them. To enable fair comparison with other models, we simply compute features once for each image and instead experiment with different noise levels: $t = \{1, 150, 250\}$. Second, prior work often computes features using both the image and some auxiliary information; *e.g.*, VPD [349] uses the image class to generate prompts for feature extraction. Such auxiliary information is not used by other models, nor is it available in many settings. Instead, we use an empty vector as the prompt similar to Zhan et al. [340].

[7] https://github.com/mlfoundations/open_clip

[8] https://huggingface.co/stabilityai/stable-diffusion-2-1

MiDaS. MiDaS is a family of models trained on a collection of monocular depth datasets using a scale-invariant depth estimation objective [222]. In this work, we consider MiDaS 3.0 [223] DPT Large, which trains a dense prediction transformer (DPT) head on top of the features extracted from a ViT-L/16. While newer iterations of MiDaS 3.1 and ZoeDepth [19] include base-size transformers, we are unable to use them due to their reliance on relative positional biases. Specifically, most ViTs rely on absolute learned or heuristic positional encoding, which can be easily interpolated to handle variable image sizes with minimal performance deterioration. However, we find that interpolating relative positional biases severely deteriorates performance. As a result, we instead use MiDaS 3.0 which used absolute positional embeddings. We note that the use of a larger backbone likely exaggerates the performance of MiDaS in our analysis. We use the checkpoint released by the authors.[9]

SAM. Kirillov et al. [138] recently proposed interactive class-agnostic segmentation as a training objective to enable generalizable, open-world segmentation. The model is trained on a novel dataset of 10M images with 1 billion masks [138]. While the SAM architecture uses a mask decoder and a prompt encoder, the features are computed by a visual transformer backbone. We use the backbone from the SAM base model which is a ViT-B/16 backbone. We use the checkpoint released by the authors.[10]

C.2 Evaluation Datasets

NAVI. NAVI is a dataset of objects annotated with high-quality 3D information, which was proposed by Jampani et al. [122]. The dataset depicts a set of N objects in a wide range of poses and environments. High-quality object meshes are aligned to each image which provides accurate depth and pose annotation. We extend the dataset by generating surface normal annotation for each image. The dataset is organized into multiview image collections, which include a larger number of multiview images of the object in the same pose and scene, as well as a wild set that depicts the object in different poses and environments. We use the full dataset and treat the multiview images for training and validation and the wild set for testing. Furthermore, we exclude 2 objects from the dataset as they do not have multiview and wildset images. For correspondence estimation experiments, we only use the wild set images and for each image, we sample a pair that has a relative rotation between 0 and 120 degrees.

[9] https://github.com/isl-org/MiDaS

[10] https://github.com/facebookresearch/segment-anything

NYU v2. The NYU Depth v2 dataset is a dataset of indoor scenes proposed by Silberman et al. [257]. The dataset consists of RGB-D video collected using a Microsoft Kinect camera and includes dense annotation for both depth and semantic segmentation. Furthermore, Ladickỳ et al. [150] provided surface normal annotations for the labeled set of 1449 images. We use the original train/test split for surface normal estimation. For depth estimation, we further include the unlabeled instances providing us with a total of 24231 images for training.

ScanNet Pairs. ScanNet [56] is a large dataset of RGB-D videos depicting indoor scenes. Sarlin et al. [241] extracted a small subset of 1500 image pairs as a benchmark for correspondence estimation. Since our correspondence estimation experiments require no training, we use all pairs as a test set.

SPair 71k. SPair-71k [195] is a dataset of image pairs that were extracted from the PASCAL datasets [80, 316]. The image pairs depict a set of 18 categories and depict different object instances of the same class. Furthermore, 8 of the categories depict non-rigid objects; *e.g.*, cats, cows, humans. All images are annotated with class-specific key points, and image pairs are further annotated with auxiliary information such as viewpoint variation. We follow the experimental setup of Zhang et al. [341] of sampling an equal number of image pairs for each class, but instead, use a larger number of 200 image pairs per class. We extend this setup by separating the sampled pairs based on the annotated viewpoint variation, which is a subjective measure of how much the viewpoint changed between the two instances and is annotated with 0, 1, or 2.

C.3 Evaluation Tasks

We evaluate all models on three tasks: monocular depth estimation, surface normal estimation, and correspondence estimation. We chose those tasks as they evaluate the model along the two dimensions of 3D awareness we are interested in: single-image 3D understanding and multiview consistency. While we train the models for depth and surface normal estimation, we directly evaluate the features for correspondence. We note that while we use the same setup for evaluating correspondence for NAVI and NYU, SPair follows a different setup due to the existence of key points, which we describe separately. We describe each of the tasks and their evaluation procedure below.

C.3.1 Monocular Depth Estimation

Task Definition: Given an image, estimate a depth value for each pixel in the image. This problem is ill-posed as it suffers from scale ambiguity; *i.e.*, a larger object that is further away will produce the same image as a smaller object that is closer to the camera. While the regularity of our environment still allows objects to learn accurate metric depth for specific image collections, such models struggle to generalize to other image collections as different camera intrinsics or image augmentations can introduce effects similar to scale ambiguity [331]. An alternative approach is to predict depth up to scale and then scale it appropriately.

We use metric depth estimation for NYU due to the regularity of the data and to enable direct comparison to prior work. However, we observe that scale-invariant is more appropriate for NAVI due to the larger variance in cameras as well as the relatively small depth variation in the object surface relative to how far the object is. As a result, we scale the depth for NAVI objects between 0 and 1 for a scale-invariant depth estimation task where 0 means the closest pixel to the camera and 1 means the furthest point on the object from the camera. This variation still enables models to learn accurate depth as shown in Chapter 6 and allows us to use standard depth evaluation metrics.

We use the AdaBins [18] parameterization of depth estimation due to its relatively strong performance. Rather than regressing depth values, Bhat et al. [18] propose dividing the depth range into several bins and estimating the probability of each. The final depth value is the weighted sum of the bin probabilities and bin center values. Similar to Oquab et al. [206], we use 256 uniformly distributed bins and only estimate the bin probabilities. We use a depth range of 0-10m for NYU and 0-1 for NAVI.

Losses: We use a combination of the scale-invariant sigmoid depth loss [73] and the gradient matching loss [167] similar to Oquab et al. [206].

Evaluation Metrics: We follow the evaluation setup of Eigen et al. [73] and compute the root-mean-square error and prediction accuracy at different thresholds. The accuracy, δ_i, is computed as the number of pixels whose ratio of depth prediction to ground truth is less than 1.25^i:

$$\delta_i(d^{pr}, d^{gt}) = \frac{1}{N} \sum_{j \in N} \max(\frac{d^{pr}_j}{d^{gt}_j}, \frac{d^{gt}_j}{d^{pr}_j}) < 1.25^i \tag{C.1}$$

where d^{pr} is predicted depth and d^{gt} is ground-truth depth.

Probe: We use a non-linear multi-scale convolutional probe. The probe takes as input multi-scale features that are extracted from several stages in the network. Prior work has shown that vision transformer features focus on different objects at different layers [1, 341] and that the granularity is not consistent across models [299, 340]. Instead of using a probe at a single layer, we train a multi-stage probe on features extracted from several layers. ConvNeXt architectures often group their layers into four stages. We follow this delineation and extract features after every stage. For ViTs, we split the layers into 4 equally sized blocks and extract features after each block; *e.g.*, for ViT-B, this is after layers 3, 6, 9, and 12. For StableDiffusion, the decoding portion of the UNet similarly consists of 4 blocks. We extract features after each of those blocks. Since prior work has found that the earliest stage features are often not useful, we only train the model on the latter three stages. This is flipped for StableDiffusion (earlier three stages) as we're sampling from the decoding part of the UNet.

Given a set of feature maps, we first use a single convolution layer to map each feature map into the same feature dimension f and concatenate them together. Next, we up-sample the features by a factor of S and apply three convolutional layers. Finally, we up-sample the features again by a factor of S and apply two final convolutional layers. All convolutional layers are followed by a ReLU activation and have a feature dimension of f, except for the final layer, whose output dimension matches the number of bins. This results in an up-sampling factor of 16. In our experiments, we set $f{=}512$, $S{=}4$, and the number of bins to 256.

Optimization: We train models for 10 epochs with a linear warm-up of the learning rate for 1.5 epochs and cosine decay to 0. We use the AdamW optimizer [171] with a linear rate of 0.001 and a weight decay of 0.01.

C.3.2 Surface Normal Estimation

Task Definition: Given an image, our goal is to estimate the direction of the surface at every pixel. The direction is predicted as a unit norm that is orthogonal to the surface at the point.

Training Loss: The cosine distance is a commonly used loss due to its relative simplicity. However, Bae et al. [11] observe that the model can be heavily penalized by areas that are ambiguous. As a result, they propose predicting a fourth value that captures the uncertainty and calibrating the loss using that value. The loss uses the estimated uncertainty of weighing the loss at each pixel while encouraging the model to minimize its uncertainty. We use the loss formulation proposed by Bae et al. [11] in our experiments.

Evaluation Metrics: For each pixel, we compute the error as the relative angle between the predicted and ground-truth surface normals in degrees. Similar to depth estimation, we compute the RMSE for each image as well as the accuracy at different thresholds. However, instead of using the ratio as done in depth, we simply compute the accuracy at different angular thresholds ($11.25°$, $22.5°$, $30°$) similar to prior work [11, 85, 214].

Probe: We use the same probe design as depth estimation with the main difference of the final layer output dimensionality being 4 instead of 256. The four values correspond to the x-, y-, and z-components of the surface normal direction the uncertainty value used in the loss computation. We normalize the 3 directional components to a unit normal.

Optimization: The optimization procedure is identical to that used for depth estimation.

C.3.3 3D Correspondence Estimation

Task Definition: Given two images that depict the same object or scene from different views, the goal is to identify pairs of pixels across images that depict the same 3D point in space. We consider two settings: object-centric and scene-centric. For the scene-centric evaluation, we allow correspondences to be computed for all pixels across the images. However, for object-centric evaluation, we only consider pixels that lie on the object mask.

Inference Procedure: Given two images, we first extract a feature map for each image. We then estimate correspondence using nearest neighbors in feature space. This provides us with correspondence for each pixel, many of which will be inaccurate. We filter the correspondences using Lowe's ratio test [179], which aims to find unique matches by discounting points that have more than 1 strong correspondence. For each point p, we find its first and second nearest neighbors: q_0 and q_1. We then compute the ratio r as follows:

$$r = 1 - \frac{D(p, q_0)}{D(p, q_1)} \tag{C.2}$$

where $D(x, y)$ is the cosine distance between x and y. We rank the correspondences using the ratio test and keep the top 1000 correspondences.

Evaluation: Correspondences are evaluated based on either 2D projection error or 3D error. Given an estimated correspondence between pixel locations p in image 1 and q in image 2, the 2D projection distance is computed by first projecting point p into 3D space using known depth and intrinsics and then projecting it into image 2 using known camera intrinsics and the relative viewpoint between the two images. This allows us to find the actual location of point p when projected into image 2: p'. The 2D correspondence error can be computed as distance between p' and q in the image plane. This works very well for scenes, but can be problematic for objects where points that are no invisible can still be projected into the image plane. While it is possible to omit surface points that are not visible, approach ignore a lot of points on thin structures; *e.g.*, points on a wire. Instead, we can simply compute the 3D correspondence error by projection both points p and q into a shared 3D space and compute the distance between them. We use the 2D projection error for scenes and 3D error for objects.

We evaluate performance using the percentage of correspondences whose error is below a specific threshold. Since we're interested in the consistency of representation, we split the image pairs based on the viewpoint change between them where θ_i^j means the error for image pairs whose relative viewpoint angle is between i and j degrees. One thing to note is that while two views with a relative angle of $180°$ depict the opposite side of the object with no mutually visible surfaces, a room viewed from the opposite corner has a relative viewpoint change of $180°$ with a large portion of the images being mutually visible. Hence, while increasing relative viewpoints imply increasing difficulty, the numbers are not directly comparable as one is viewing the scene from the inside of it but viewing the object from the outside.

C.3.4 Semantic Correspondence Estimation

Task Definition: Given two images and a set of semantic key points in image one, the goal is to find the pixel location belonging to those key points in the second image. Key points are often semantic parts; *e.g.*, a cat's left ear or the front right wheel of the car. Unlike the previous task, where one has to find a set of points in both images that match each other, the set of points in the first image are already specified. Furthermore, while the previous task is matching points belonging to the same scene or the same object instance, semantic keypoints are defined at the class level so the images often depict two different instance; *e.g.*, two different cats or two different cars.

Inference Procedure: We follow prior work [275, 341] and simply use nearest neighbors. There is no need for filtering as the goal is to just find the point in the second image that is most similar to the keypoint.

Evaluation Metrics: The evaluation is often based one percentage recall of keypoints withing a pixel threshold; *i.e.*, the percentage of predicted keypoints within N pixels of the ground-truth match. The evaluation is based on the assumption that each key point has a single valid match in the second image. This results in each evaluation only considering the predicted key point and its ground-truth location and ignoring everything else. The experiments reported in Appendix C.5, but we also consider an alternative evaluation as discussed in Chapter 6.

An alternative way to evaluate the prediction is to compare all the key points in that image. Instead of asking how close the prediction is to the ground-truth for the same keypoint type, we can ask which ground-truth keypoint is closest to the prediction. This allows us to understand which key points are getting confused with each other rather than how many key points are being correctly classified. This is important since the threshold is usually 10-20% of the bounding box size, which can include several different keypoints.

Prior work reports the average performance for all pairs. Instead, we separate the performance for image pairs of different viewpoint changes. Specifically, we use the viewpoint variation annotation provided by SPair [195] and report the performance for different viewpoint difficulties.

C.4 Performance Correlation

One question tackled in Chapter 6 is how well the performance is correlated across tasks. If several tasks are measuring the same capability, we would expect their performance to be well-correlated. Although a high correlation could be caused by a variety of other factors. Hence, while a high correlation provides some evidence that the tasks are measuring the same capability, a very low correlation would imply that the tasks are not related.

We compute the correlation of model performance across different tasks and task domains. Specifically, we compute the Pearson correlation coefficient, which assumes a linear relationship between models. One possibility is that the relationships across tasks may not be linear and that a rank correlation might be well-suited. Empirically, we observe that both statistical measures often result in similar trends with some minor exceptions. When considering cases where the correlations deviate from each other, we find that they are caused by fluctuations in the rankings that arise for very small changes in performance for similarly performing models. As a result, we choose to report the Pearson correlation coefficient as a descriptive statistic of the relationships between model performance.

When considering the overall model performance, such as Figure 1, we aggregate performance across all tasks. Since the absolute performance values are not directly comparable, we instead rely on model rank. While a model's ranking can fluctuate due to minor differences, such fluctuations tend to get canceled out when averaging the ranking across multiple tasks. We convert the rankings to a normalized rating where 1 means the best-performing model and 0 means the worst performing model. The overall ratings are shown in Figure 1. We emphasize that such ratings represent the relative, not absolute, model performance. Hence, a rating of 1 does not mean the representations are 3D aware but rather that they are more 3D aware than the other models considered.

C.5 Complete Results

We chose to focus on overall performance trends and salient comparisons in the main body of the paper. In the supplemental, we report the complete results for all models and tasks considered in Tables C.1 to C.5. We discuss each of the results below and provide some additional analysis.

What explains CLIP's low performance? One interesting finding is that CLIP appears to perform poorly across all tasks. This supports prior work which shows that CLIP even struggles with 2D spatial relationships and behaves like a bag-of-words model [159]. We considered different possibilities: training data, training objective, model architecture, and augmentations. One possibility is that CLIP's WIT data does not capture such relationships. However, we note that the OpenCLIP checkpoint trained on LAION achieves a very similar performance across all tasks. It is worth noting that StableDiffusion is one of the strongest performing models and is trained using LAION. Another possibility is that this is caused by the training objective; *i.e.*, the contrastive objective discourages such relationships. We compared the CLIP model to SigLIP which is trained with a non–contrastive objective. While SigLIP outperforms CLIP on most tasks, its performance is still much lower than other models.

The largest improvement comes from changing the backbone from ViT to ConvNeXt. This change results in a qualitative change in CLIP's performance; *e.g.*, from predicting flat surfaces for depth to generating something that looks like a depth map. We note that this is not because ConvNeXt is a strictly superior architecture; *e.g.*, we find that DeiT's ViT performs better than ConvNeXt for supervised training on ImageNet-22k. This suggests that training signal or model architecture do not independently determine the 3D awareness of features, although more controlled experiments are needed to confirm this claim.

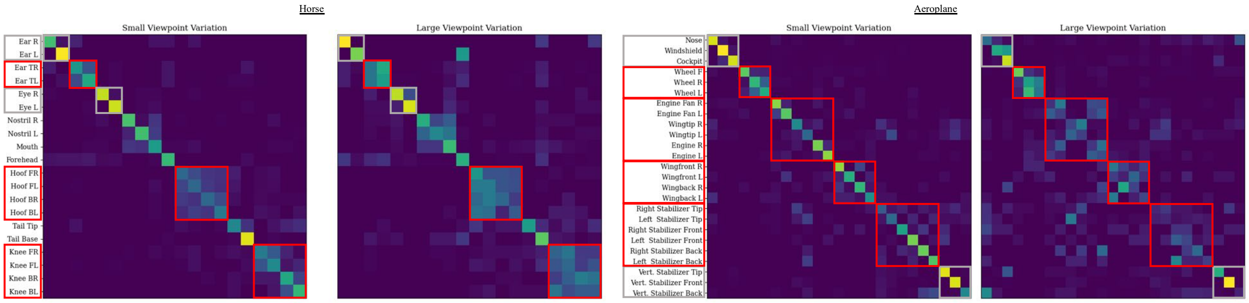


Figure C.1: **Keypoint confusion matrices for horses and aeroplanes.** We find similar confusion patters in other SPair classes where large viewpoint changes result in high confusion between semantically related classes (highlighted in red). Furthermore, we find that keypoints that experience a lot of deformation (*e.g.*, knees and hooves) are confused for all image pairs as they appear in different 2D locatons relative to each other. However, we find that classes (highlighted in grey) that often appear in the same 2D configuration do not suffer from this effect.

How does noise level affect StableDiffusion? There has been a recent surge of papers on using StableDiffusion for, or analyzing its features on, a range of 3D tasks [46, 275, 340, 341]. One common variation is how to prompt StableDiffusion. Some models use a text prompt to condition the denoising approach during feature extraction [46, 275, 349]. While this can improve performance, such prompts are not always available, so we use an empty string for conditioning similar to Zhan et al. [340]. Another common design choice is to extract features for multiple noise samples and average them. This is often combined with a larger noise level. However, this implicitly converts the model into a classifier ensemble [205] which is known to improve performance. Furthermore, one could similarly generate such ensembles by running other models on different augmented versions of the same image. We extract features for a single pass of the network.

In our experiments, we analyze the impact of two factors: where to sample features from and what noise level to use. For feature extraction, our results are consistent with prior work that find that the output of the second decoding block results in the best performance. We observe this across correspondence tasks, as well as with linear probes in preliminary experiments. On the other hand, the results for noise level are less consistent and depend on the task. However, the variance is typically small, suggesting that the model is able to extract good features for different noise levels. We note that while our results are consistent with Chen et al. [46] who also find that depth representations even for larger noise levels, we observe a larger variance in performance when extracting representations from different layers.

Additional SPair Confusion Matrices. In Chapter 6, we visualized the confusion matrices for the chair class under different viewpoint variations. The chair class is most representative of this distinction for two reasons: (1) its keypoints neatly segment into semantic groups that only differ based on their relative location in the chair's canonical frame of references; and (2) semantically similar keypoints can be visible in the same image in different relative orientations for each other. Many other classes do not fulfill those criteria, especially the second point. For example, several keypoints for humans and animals are unique; *e.g.*, mouth, nose, tail, forehead. In other cases, semantically related keypoints almost always appear in the same 2D configuration. For example, eyes and ears often appear in the same 2D configuration when they are both visible. As a result, their 2D relative locations are very strongly correlated with their identity making it difficult to assess the 3D awareness of the model. This is confounded by the additional bias in the dataset of most animals and humans pictured facing the camera. Finally, many symmetric key points are almost never co-visible. It is very rare for both front left and front right wheels to be co-visible in the same image. Most car pictures featuring more than 1 visible wheel are side views. Hence, for most car view pairs, a combination of coarse semantic class (*i.e.*, car wheel) and the relative location in the image can result in accurate correspondence.

We observe those findings in the data. We present the confusion matrices StableDiffusion features on two additional classes: horses and aeroplanes. The results are shown in Fig. C.1. First, we find that unique classes such as tail tip and tail base have similar accuracy across viewpoint variations. Furthermore, classes that appear in similar orientations such as eyes or ears are similarly unaffected. We note that the ear annotation refers to the front of the ear for horses. Meanwhile, the top of the ear, which is also visible when the horse is looking away exhibits a different behavior where it is far more confused when for larger viewpoint changes.

We observe similar patterns for airplanes where unique key points such as the airplane noses, windshields, and cockpits are all accurately predicted for large viewpoint changes. Meanwhile, wheels, wings, and stabilizers are all confused for larger viewpoint changes. To support that this is not simply due to some key points being more difficult, one can compare the performance of vertical stabilizers to the performance of right and left stabilizers. While the right and left stabilizers are confused, the vertical stabilizer, which looks the same but has a very different orientation, is still accurately predicted for larger viewpoint changes.

Finally, we observe interesting trends for knees and hooves, which appear to be confused regardless of viewpoint change. Since animals can deform, hooves and legs consistently appear in different orientations, especially for horses, which are often pictured while moving. Nevertheless, the confusion is strongly restricted to the semantically equivalent classes. This strongly suggests that while the model understands semantics, it lacks 3D awareness.

Table C.1: **Depth Estimation Results.** We present the depth estimation results for all models. Models are grouped based on the supervisory signal.

			NYU				NAVI			
Model	**Architecture**	**Dataset**	δ_1	δ_2	δ_3	RMSE	δ_1	δ_2	δ_3	RMSE
Self-Supervised Models										
MAE [103]	ViT-B16	IN-1k	65.86	90.79	97.71	0.6827	45.40	71.83	84.84	0.1336
FCMAE [313]	ConvNeXtv2-B	IN-22k	78.83	96.25	99.27	0.5419	58.58	82.13	91.30	0.1014
iBOT [352]	ViT-B16	IN-1k	82.15	97.08	99.36	0.4903	57.15	81.16	90.72	0.1037
DINO[33]	ViT-B16	IN-1k	81.08	96.68	99.28	0.5059	61.49	83.82	92.12	0.0946
DINOv2 [206]	ViT-B14	LVD	93.67	99.40	99.90	0.3245	69.59	88.97	95.19	0.0765
DINOv2+reg [59]	ViT-B14	LVD	93.40	99.34	99.90	0.3352	66.63	87.54	94.48	0.0823
Densely-Supervised Models										
SAM [138]	ViT-B16	SA-1B	75.06	94.93	99.04	0.5781	56.52	80.84	90.57	0.1047
MiDaS [223]	ViT-L16	MIX 6	79.35	95.85	98.95	0.5216	60.53	83.20	91.74	0.0974
Classification-Supervised Models										
DeiT III [287]	ViT-B16	IN-22k	86.03	97.99	99.69	0.4390	66.58	87.16	94.04	0.0839
ConvNeXT [173]	ConvNeXt-B	IN-22k	80.49	96.71	99.44	0.5111	57.51	81.21	90.63	0.1049
Language-Supervised Models										
SigLIP [339]	ViT-B16	WebLI	61.81	88.70	97.11	0.7421	37.53	63.90	79.36	0.1558
CLIP [219]	ViT-B16	WIT	49.76	79.88	92.90	0.9564	25.45	49.37	68.81	0.1988
CLIP [119]	ViT-B16	LAION	50.22	80.24	93.00	0.9478	24.57	48.05	67.70	0.2014
CLIP [119]	ConvNeXt-B	LAION	80.05	96.51	99.37	0.5106	55.55	80.65	90.52	0.1067
CLIP [119] + AugReg	ConvNeXt-B	LAION	82.68	97.14	99.49	0.4820	57.45	81.79	91.19	0.1027
Text-Conditioned Image Generation Models										
StableDiffusion [229] (t=1)	UNet	LAION	84.37	97.38	99.54	0.4494	57.79	82.03	91.24	0.0999
StableDiffusion [229] (t=150)	UNet	LAION	85.02	97.50	99.55	0.4461	59.13	82.80	91.82	0.0979
StableDiffusion [229] (t=250)	UNet	LAION	82.90	96.81	99.36	0.4835	57.57	81.69	91.13	0.1024

Table C.2: **Surface Normal Estimation Results.** We present the surface normal estimation results for all models. Models are grouped based on the supervisory signal.

			NYU				NAVI			
Model	**Architecture**	**Dataset**	11.25°	22.5°	30°	RMSE	11.25°	22.5°	30°	RMSE
Self-Supervised Models										
MAE [103]	ViT-B16	IN-1k	39.36	60.27	68.67	35.42	23.99	50.41	63.38	35.78
FCMAE [313]	ConvNeXtv2-B	IN-22k	32.04	53.44	63.30	38.01	34.14	60.48	71.31	32.18
iBOT [352]	ViT-B16	IN-1k	46.28	66.58	74.00	32.11	35.14	60.95	71.84	31.50
DINO[33]	ViT-B16	IN-1k	42.14	62.75	71.15	33.68	34.21	60.34	71.47	31.68
DINOv2 [206]	ViT-B14	LVD	60.36	78.42	84.26	24.64	43.34	69.12	78.80	27.32
DINOv2+reg [59]	ViT-B14	LVD	59.98	78.22	84.05	24.72	41.65	68.02	78.01	27.72
Densely-Supervised Models										
SAM [138]	ViT-B16	SA-1B	42.48	65.87	75.00	30.65	32.96	59.92	71.46	31.41
MiDaS [223]	ViT-L16	MIX 6	43.66	64.40	72.22	33.20	35.25	61.30	72.12	31.41
Classification-Supervised Models										
DeiT III [287]	ViT-B16	IN-22k	45.58	67.13	75.03	31.04	38.08	64.00	74.41	30.04
ConvNeXT [173]	ConvNeXt-B	IN-22k	36.81	59.46	69.15	34.25	29.01	55.77	67.69	33.76
Language-Supervised Models										
SigLIP [339]	ViT-B16	WebLI	26.05	45.37	55.86	39.95	16.03	40.21	54.09	40.05
CLIP [219]	ViT-B16	WIT	22.90	42.65	53.42	40.27	10.67	31.83	45.75	44.41
CLIP [119]	ViT-B16	LAION	23.32	43.01	53.70	40.41	12.37	34.52	48.39	43.34
CLIP [119]	ConvNeXt-B	LAION	36.93	59.29	69.03	34.08	29.46	56.00	67.90	33.65
CLIP [119] + AugReg	ConvNeXt-B	LAION	38.51	61.28	70.79	33.10	31.18	57.96	69.48	32.87
Text-Conditioned Image Generation Models										
StableDiffusion [229] (t=1)	UNet	LAION	58.77	76.39	82.21	26.55	38.69	65.52	76.10	28.91
StableDiffusion [229] (t=150)	UNet	LAION	58.22	75.58	81.39	27.01	39.53	65.59	75.96	28.89
StableDiffusion [229] (t=250)	UNet	LAION	55.41	72.95	79.03	28.75	37.93	63.67	74.24	29.93

Table C.3: **Correspondence Estimation Results for ScanNet.** We present the ScanNet correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned for different relative viewpoint changes between image pairs. The highest performing set of results for each model are highlighted and bolded.

			Block_0				Block_1				Block_2				Block_3			
Model	**Architecture**	**Dataset**	θ_{0}^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_{0}^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_{0}^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_{0}^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}
Self-Supervised Models																		
MAE [103]	ViT-B16	IN-1k	3.4	2.8	3.9	2.4	4.7	3.6	3.9	2.6	7.8	5.5	4.9	3.1	**13.5**	**8.2**	**6.0**	**3.6**
FCMAE [313]	ConvNeXtv2-B	IN-22k	26.8	20.1	12.2	5.6	**60.8**	**49.8**	**28.1**	**9.9**	31.8	22.6	13.3	6.0	36.6	26.6	14.9	7.3
iBOT [352]	ViT-B16	IN-1k	7.0	5.1	5.1	3.0	11.9	8.0	6.8	3.3	19.5	13.1	9.6	4.1	**27.3**	**18.3**	**12.8**	**5.7**
DINO[33]	ViT-B16	IN-1k	14.3	9.6	7.9	4.0	42.9	32.4	21.1	9.1	44.5	34.1	22.3	9.8	**45.0**	**34.4**	**22.6**	**10.7**
DINOv2 [206]	ViT-B14	LVD	25.4	19.4	12.7	4.9	**47.0**	**36.4**	**22.4**	**8.5**	37.6	26.7	16.8	7.5	37.0	27.5	19.7	11.2
DINOv2+reg [59]	ViT-B14	LVD	29.1	24.7	15.1	5.7	**56.1**	**47.4**	**29.5**	**10.3**	48.4	37.9	24.2	9.9	41.9	33.6	23.3	12.2
Densely-Supervised Models																		
SAM [138]	ViT-B16	SA-1B	8.3	6.0	5.7	2.9	35.3	26.5	17.5	5.3	53.1	44.9	29.4	8.9	**55.4**	**47.1**	**30.4**	**9.5**
MiDaS [223]	ViT-L16	MIX 6	50.5	39.1	24.5	11.2	**56.5**	**47.4**	**31.6**	**13.9**	55.6	46.1	30.8	14.3	52.5	42.2	27.6	13.2
Classification-Supervised Models																		
DeiT III [287]	ViT-B16	IN-22k	17.6	12.2	8.8	3.3	**38.3**	**29.8**	**17.8**	**7.2**	28.5	21.3	13.9	6.7	20.6	13.8	9.2	5.0
ConvNeXT [173]	ConvNeXt-B	IN-22k	23.4	17.3	10.2	4.8	**43.3**	**33.3**	**18.0**	**6.9**	10.0	6.4	3.9	2.8	14.4	10.1	5.5	3.6
Language-Supervised Models																		
SigLIP [339]	ViT-B16	WebLI	5.7	4.1	4.4	2.3	12.0	7.0	5.6	3.0	12.0	7.6	5.6	3.3	**14.2**	**10.4**	**8.0**	**5.2**
CLIP [219]	ViT-B16	WIT	**15.5**	**11.3**	**8.2**	**4.3**	11.0	7.4	6.1	3.9	5.9	4.0	3.4	2.4	4.0	2.7	2.4	1.8
CLIP [119]	ViT-B16	LAION	**20.9**	**15.4**	**10.4**	**5.2**	13.1	8.9	6.6	3.9	5.4	3.7	3.2	2.4	3.4	2.4	2.2	1.8
CLIP [119]	ConvNeXt-B	LAION	27.4	21.2	13.4	6.6	31.3	22.5	12.3	5.3	**46.8**	**38.0**	**22.3**	**8.9**	36.9	28.5	16.4	7.9
CLIP [119] + AugReg	ConvNeXt-B	LAION	24.6	19.3	12.6	6.3	**37.7**	**27.5**	**14.5**	**6.0**	32.2	22.8	12.8	5.5	31.1	22.4	12.8	7.0
Text-Conditioned Image Generation Models																		
StableDiffusion [229] (t=1)	UNet	LAION	10.2	5.1	3.0	1.3	**66.5**	**55.0**	**31.1**	**8.3**	63.3	50.8	29.3	9.4	31.0	23.5	14.5	6.7
StableDiffusion [229] (t=150)	UNet	LAION	14.3	7.8	4.0	1.5	**62.8**	**53.1**	**32.6**	**8.4**	54.4	43.2	24.0	7.2	21.9	17.0	11.1	5.5
StableDiffusion [229] (t=250)	UNet	LAION	13.7	7.8	4.0	1.7	**56.8**	**47.5**	**28.2**	**7.2**	47.7	36.6	19.9	6.3	19.2	14.8	10.0	5.1

Table C.4: **Correspondence Estimation Results for NAVI.** We present the NAVI correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned for different relative viewpoint changes between image pairs. The highest performing set of results for each model are highlighted and bolded.

			Block_0				Block_1				Block_2				Block_3			
Model	**Architecture**	**Dataset**	θ_{0}^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_{0}^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_{0}^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_{0}^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}
Self-Supervised Models																		
MAE [103]	ViT-B16	IN-1k	75.7	39.5	18.8	10.8	79.0	41.3	19.5	11.1	**79.6**	**41.9**	**19.6**	**11.3**	76.6	39.9	18.9	11.0
FCMAE [313]	ConvNeXtv2-B	IN-22k	40.9	24.5	15.9	11.1	67.4	36.1	19.5	11.9	73.8	47.0	28.9	18.3	**82.4**	**57.7**	**39.5**	**26.9**
iBOT [352]	ViT-B16	IN-1k	70.2	36.4	18.4	11.2	81.8	42.9	20.4	11.6	87.3	47.9	21.8	11.9	**89.1**	**50.3**	**23.5**	**13.0**
DINO[33]	ViT-B16	IN-1k	87.5	48.7	22.3	12.5	90.0	56.9	29.4	18.4	**87.9**	**56.8**	**31.0**	**20.5**	86.8	55.5	30.7	19.9
DINOv2 [206]	ViT-B14	LVD	81.4	44.6	21.6	12.5	94.2	62.4	29.0	14.8	94.5	68.6	36.3	20.3	**90.4**	**69.7**	**45.4**	**32.2**
DINOv2+reg [59]	ViT-B14	LVD	75.4	40.7	20.9	12.3	93.2	61.0	28.5	14.5	94.5	69.6	37.8	21.3	**88.4**	**67.3**	**44.3**	**31.2**
Densely-Supervised Models																		
SAM [138]	ViT-B16	SA-1B	80.2	41.7	19.6	11.4	84.4	47.4	22.0	12.0	89.7	55.1	25.0	12.7	**89.5**	**56.0**	**25.1**	**12.7**
MiDaS [223]	ViT-L16	MIX 6	81.4	48.0	24.1	14.5	83.5	55.8	30.7	21.0	**82.5**	**55.8**	**31.9**	**22.3**	79.8	52.7	30.0	21.1
Classification-Supervised Models																		
DeiT III [287]	ViT-B16	IN-22k	89.9	49.9	22.7	12.2	**91.7**	**62.3**	**33.9**	**21.7**	84.0	58.3	38.1	26.6	62.6	38.1	25.2	17.1
ConvNeXT [173]	ConvNeXt-B	IN-22k	41.1	24.2	15.6	10.2	50.0	25.4	14.6	9.3	74.4	46.2	25.6	16.7	**79.0**	**50.7**	**30.8**	**21.1**
Language-Supervised Models																		
SigLIP [339]	ViT-B16	WebLI	**73.5**	**37.7**	**18.1**	**11.3**	69.3	38.0	19.7	12.2	45.1	26.9	16.8	11.5	37.7	23.5	15.8	11.8
CLIP [219]	ViT-B16	WIT	**48.6**	**27.0**	**16.3**	**11.1**	38.7	23.4	14.9	10.4	27.8	18.4	12.7	9.1	22.9	15.8	11.3	8.2
CLIP [119]	ViT-B16	LAION	**44.4**	**25.1**	**15.3**	**10.7**	38.5	22.4	14.0	9.9	26.5	17.4	12.2	8.6	22.5	15.5	11.1	7.8
CLIP [119]	ConvNeXt-B	LAION	36.8	22.6	14.7	10.6	47.3	27.2	17.2	11.3	**85.3**	**55.6**	**32.5**	**19.8**	77.6	46.8	29.6	20.3
CLIP [119] + AugReg	ConvNeXt-B	LAION	37.0	22.6	14.7	10.6	49.3	27.6	16.5	10.7	**84.6**	**53.8**	**31.1**	**17.8**	79.7	49.4	31.5	21.5
Text-Conditioned Image Generation Models																		
StableDiffusion [229] (t=1)	UNet	LAION	79.3	36.1	15.4	7.7	**93.4**	**60.1**	**24.3**	**11.2**	75.4	42.6	20.6	11.5	44.1	24.9	15.8	10.6
StableDiffusion [229] (t=150)	UNet	LAION	80.0	36.9	14.8	7.5	**92.5**	**59.4**	**24.0**	**11.3**	73.2	41.0	19.3	10.9	42.9	25.1	15.8	10.6
StableDiffusion [229] (t=250)	UNet	LAION	80.1	37.1	15.0	7.5	**91.0**	**57.2**	**22.8**	**10.8**	70.5	39.0	18.7	10.5	40.9	24.4	15.5	10.3

Table C.5: **Correspondence Estimation Results for SPair-71k.** We present the SPair-71k correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned based on the viewpoint variation for the image pair. The highest performing set of results for each model are highlighted and bolded.

			Block_0			Block_1			Block_2			Block_3		
Model	**Architecture**	**Dataset**	a0	a1	a2	a0	a1	a2	a0	a1	a2	a0	a1	a2
Self-Supervised Models														
MAE [103]	ViT-B16	IN-1k	7.5	4.2	3.5	8.9	5.7	4.3	**9.8**	**6.1**	**5.2**	9.4	6.3	4.8
FCMAE [313]	ConvNeXtv2-B	IN-22k	5.2	5.2	4.6	8.5	6.5	6.2	25.5	24.2	25.5	**28.5**	**26.3**	**28.5**
iBOT [352]	ViT-B16	IN-1k	7.0	4.1	4.4	10.0	5.6	4.9	13.8	7.1	5.1	**17.0**	**9.2**	**7.1**
DINO[33]	ViT-B16	IN-1k	14.7	8.0	6.2	26.1	18.4	18.2	**31.8**	**25.5**	**25.2**	29.7	24.6	25.1
DINOv2 [206]	ViT-B14	LVD	12.6	8.4	7.0	36.4	21.5	16.8	56.1	37.3	34.4	**62.7**	**52.4**	**51.8**
DINOv2+reg [59]	ViT-B14	LVD	11.8	8.3	7.2	33.6	20.3	15.9	57.8	40.3	36.4	**58.5**	**51.7**	**51.7**
Densely-Supervised Models														
SAM [138]	ViT-B16	SA-1B	9.2	5.7	4.3	15.9	10.0	8.2	28.5	18.9	13.9	**30.1**	**19.4**	**13.7**
MiDaS [223]	ViT-L16	MIX 6	15.8	10.7	8.5	27.1	23.3	23.7	**27.9**	**25.4**	**26.3**	25.7	22.5	23.8
Classification-Supervised Models														
DeiT III [287]	ViT-B16	IN-22k	21.5	12.4	9.9	**41.3**	**32.0**	**33.8**	37.9	33.7	35.4	16.2	15.4	16.5
ConvNeXT [173]	ConvNeXT-B	IN-22k	4.3	4.2	3.5	8.2	6.5	6.8	**20.4**	**17.3**	**17.0**	14.7	12.4	11.3
Language-Supervised Models														
SigLIP [339]	ViT-B16	WebLI	11.0	6.1	4.9	**12.3**	**7.9**	**7.1**	8.3	7.0	7.1	5.8	5.4	5.8
CLIP [219]	ViT-B16	WIT	**6.3**	**5.4**	**5.3**	5.4	4.7	4.0	5.2	3.9	2.8	6.2	3.4	2.2
CLIP [119]	ViT-B16	LAION	5.4	4.8	3.7	5.2	4.1	3.7	6.9	3.2	2.7	**7.8**	**3.2**	**2.3**
CLIP [119]	ConvNeXt-B	LAION	5.7	5.5	4.9	5.1	3.8	4.4	**23.9**	**19.9**	**19.5**	21.3	19.1	22.4
CLIP [119] + AugReg	ConvNeXt-B	LAION	5.3	4.7	3.4	7.0	6.0	6.3	26.9	22.7	20.9	**27.1**	**24.9**	**28.0**
Text-Conditioned Image Generation Models														
StableDiffusion [229] (t=1)	UNet	LAION	13.9	5.7	4.1	**57.5**	**34.8**	**27.7**	26.2	18.3	14.5	6.8	6.1	5.8
StableDiffusion [229] (t=150)	UNet	LAION	17.5	7.5	5.8	**62.6**	**40.8**	**32.3**	30.9	21.2	16.1	6.5	5.7	5.7
StableDiffusion [229] (t=250)	UNet	LAION	17.9	7.6	5.1	**61.4**	**40.5**	**30.7**	30.3	20.8	16.5	6.7	5.6	6.3

Bibliography

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021. 88, 95, 98, 103, 126

[2] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. SPICE: Semantic propositional image caption evaluation. In *ECCV*, 2016. 70

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *ICCV*, 2015. 68

[4] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *CVPR*, 2019. 9, 29

[5] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 46, 54, 55, 56

[6] Federica Arrigoni, Luca Magri, Beatrice Rossi, Pasqualina Fragneto, and Andrea Fusiello. Robust absolute rotation estimation via low-rank and sparse matrix decomposition. In *3DV*, 2014. 47, 50

[7] Federica Arrigoni, Beatrice Rossi, and Andrea Fusiello. Spectral synchronization of multiple views in se (3). *SIAM Journal on Imaging Sciences*, 2016. 46, 47, 50

[8] KS Arun, TS Huang, and SD Blostein. Least square fitting of two 3-d point sets. In *TPAMI*, 1987. 9, 29, 47

[9] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2019. 66, 68, 69, 75

[10] Mido Assran, Randall Balestriero, Quentin Duval, Florian Bordes, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, and Nicolas Ballas. The hidden uniform cluster prior in self-supervised learning. In *ICLR*, 2023. 3, 69, 78

[11] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *ICCV*, 2021. 93, 126, 127

[12] Hessam Bagherinezhad, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. Are elephants bigger than butterflies? reasoning about sizes of objects. In *AAAI*, 2016. 84

[13] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *CVPR*, 2020. 8, 27, 28, 29, 34

[14] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *BMVC*, 2016. 46

[15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006. 8, 46

[16] Alexander C Berg, Tamara L Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, pages 26–33. Citeseer, 2005. 95

[17] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*. International Society for Optics and Photonics, 1992. 28, 29

[18] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021. 93, 125

[19] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 94, 123

[20] Anand Bhattad, Daniel McKee, Derek Hoiem, and DA Forsyth. Stylegan knows normal, depth, albedo, and more. *arXiv preprint arXiv:2306.00987*, 2023. 88, 103

[21] Jia-Wang Bian, Huangying Zhan, Naiyan Wang, Tat-Jun Chin, Chunhua Shen, and Ian Reid. Unsupervised depth learning in challenging indoor video: Weak rectification to rescue. *arXiv*, 2020. 9

[22] Tolga Birdal, Michael Arbel, Umut Şimşekli, and Leonidas Guibas. Synchronizing probability measures on rotations via optimal transport. In *CVPR*, 2020. 47

[23] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of Annual Conference on Computational learning theory*, 1998. 69

[24] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. 81

[25] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 113

[26] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, 2019. 7, 9, 29, 47

[27] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *CVPR*, 2017. 7, 9, 29, 47

[28] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *Robotics and Automation Letters (RA-L)*, 2018. 107

[29] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. 64

[30] Luca Carlone, Roberto Tron, Kostas Daniilidis, and Frank Dellaert. Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization. In *ICRA*, 2015. 47

[31] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 28, 66, 68, 69, 75

[32] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Adv. Neural Inform. Process. Syst.*, volume 33, 2020. 66, 68, 69, 73, 75, 78, 118, 119

[33] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 91, 121, 134, 135, 136

[34] Vincent Michael Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019. 9, 10

[35] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. In *3DV*, 2017. 45, 46

[36] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. In *ICML*, 2023. 87

[37] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 73, 79

[38] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *JMLR*, 2021. 53

[39] K Chatfield, K Simonyan, A Vedaldi, and A Zisserman. Return of the devil in the details: delving deep into convolutional nets. In *BMVC*, 2014. 103

[40] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, 2020. 2, 66, 67, 68, 73, 74, 80

[41] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *Adv. Neural Inform. Process. Syst.*, volume 33, 2020. ix, 68, 70, 78, 80, 81, 115, 118, 119

[42] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 28, 30, 31, 34, 67, 68, 72, 75, 78, 118, 119

[43] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 66, 68

[44] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 78, 115, 118, 119

[45] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 1992. 28, 29

[46] Yida Chen, Fernanda Viégas, and Martin Wattenberg. Beyond surface statistics: Scene representations in a latent diffusion model. *arXiv preprint arXiv:2306.05720*, 2023. 88, 103, 131

[47] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, October 2017. ISSN 1558-2256. doi: 10.1109/jproc.2017.2675998. URL http://dx.doi.org/10.1109/JPROC.2017.2675998. 113

[48] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 68

[49] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 31, 36

[50] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 7, 15, 18, 27, 28, 29, 31, 34, 35, 36, 39, 40, 41, 47

[51] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *CVPR*, 2020. 7, 9, 12, 16, 17, 18, 20, 29, 31, 32, 36, 37, 38, 47, 49

[52] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 8

[53] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 113

[54] Adam Coates, Andrew Ng, and Honglak Lee. An Analysis of Single Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011. 113

[55] Yufeng Cui, Lichen Zhao, Feng Liang, Yangguang Li, and Jing Shao. Democratizing contrastive language-image pre-training: A CLIP benchmark of data, model, and supervision. In *ICML Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward*, 2022. 68, 73

[56] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *CVPR*, 2017. 8, 15, 28, 35, 45, 46, 53, 97, 124

[57] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM ToG*, 2017. 45, 46, 53

[58] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 1

[59] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023. 121, 134, 135, 136

[60] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *ECCV*, 2018. 8, 29, 39, 40

[61] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *CVPR*, June 2018. 29, 39, 40

[62] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. In *CVPR*, 2019. 8, 29, 47

[63] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 1, 87, 91

[64] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11162–11173, 2021. 8, 30, 68

[65] Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. RedCaps: Web-curated image-text data created by the people, for the people. In *NeurIPS Datasets and Benchmarks*, 2021. 73, 79, 87

[66] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, 2018. 7, 8, 15, 18, 46, 54, 55, 56

[67] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2018. 71

[68] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 8, 30, 68

[69] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014. 1

[70] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. In *CVPR*, 2019. 8

[71] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9588–9597, October 2021. 66, 69, 75, 86, 116

[72] Ufuk Efe, Kutalmis Gokalp Ince, and A Aydin Alatan. Effect of parameter optimization on classical and learning-based image matching methods. In *CVPR*, 2021. 55

[73] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeruIPS*, 2014. 125

[74] Mohamed El Banani and Justin Johnson. Bootstrap your own correspondences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6433–6442, 2021. 98

[75] Mohamed El Banani and Justin Johnson. Bootstrap Your Own Correspondences. In *ICCV*, 2021. 4, 26, 45, 46, 47, 49, 53, 55, 56

[76] Mohamed El Banani, Jason J. Corso, and David Fouhey. Novel object viewpoint estimation through reconstruction alignment. In *CVPR*, 2020. 9

[77] Mohamed El Banani, Luya Gao, and Justin Johnson. Unsupervisedr&r: Unsupervised point cloud registration via differentiable rendering. In *CVPR*, 2021. 4, 6, 29, 32, 33, 45, 46, 47, 49, 50, 59, 68

[78] Mohamed El Banani, Karan Desai, and Justin Johnson. Learning Visual Representations via Language-Guided Sampling. In *CVPR*, 2023. 5, 65

[79] Mohamed El Banani, Ignacio Rocco, David Novotny, Andrea Vedaldi, Natalia Neverova, Justin Johnson, and Ben Graham. Self-supervised Correspondence Estimation via Multiview Registration. In *WACV*, 2023. 4, 44

[80] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 124

[81] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004. 113

[82] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981. 7, 15, 20, 47, 49, 50

[83] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 1973. 89

[84] David Fouhey. *Factoring Scenes into 3D Structure and Style*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, June 2016. 92

[85] David F. Fouhey, Wajahat Hussain, Abhinav Gupta, and Martial Hebert. Single image 3D without a single 3D image. In *ICCV*, 2015. 127

[86] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 1, 2

[87] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 8, 30, 68

[88] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *CVPR*, 2019. 8, 29, 34, 36, 40, 50

[89] Zan Gojcic, Caifa Zhou, Jan D. Wegner, Leonidas J. Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *CVPR*, 2020. x, 7, 9, 12, 16, 17, 18, 20, 29, 31, 32, 36, 37, 38, 46, 47, 50, 110, 111

[90] Micah Goldblum, Hossein Souri, Renkun Ni, Manli Shu, Viraj Prabhu, Gowthami Somepalli, Prithvijit Chattopadhyay, Mark Ibrahim, Adrien Bardes, Judy Hoffman, Rama Chellappa, Andrew Gordon Wilson, and Tom Goldstein. Battle of the backbones: A large-scale comparison of pretrained models across computer vision tasks. In *NeurIPS Datasets and Benchmarks Track*, 2023. 87, 90, 120

[91] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. In *IJCV*, 2014. 68

[92] Walter Goodwin, Sagar Vaze, Ioannis Havoutis, and Ingmar Posner. Zero-shot category-level object pose estimation. In *ECCV*, 2022. 88, 103

[93] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975. 12, 32

[94] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019. 8, 30, 87, 90, 103

[95] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017. 68, 103

[96] Benjamin Graham. Sparse 3D convolutional neural networks. In *BMVC*. Springer, 2015. 31, 47

[97] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. In *NeurIPS*, 2020. 28, 30, 34, 68, 75

[98] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 2, 68

[99] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020. 68, 69, 86

[100] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 8

[101] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. ix, 70, 73

[102] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 68

[103] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 2, 68, 91, 120, 134, 135, 136

[104] Donald Hebb. The organization of behavior. emphnew york, 1949. 1

[105] Eric Hedlin, Gopal Sharma, Shweta Mahajan, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. Unsupervised semantic correspondence using stable diffusion. *arXiv preprint arXiv:2305.15581*, 2023. 88, 103

[106] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 113

[107] Richard Held, Yuri Ostrovsky, Beatrice de Gelder, Tapan Gandhi, Suma Ganesh, Umang Mathur, and Pawan Sinha. The newly sighted fail to match seen with felt. *Nature neuroscience*, 14(5):551–553, 2011. 107

[108] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. PointGMM: A Neural GMM Network for Point Clouds. In *CVPR*, 2020. 9, 29, 47

[109] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021. 70

[110] Donald D Hoffman, Manish Singh, and Chetan Prakash. The interface theory of perception. *Psychonomic bulletin & review*, 22:1480–1506, 2015. 2

[111] Ji Hou, Saining Xie, Benjamin Graham, Angela Dai, and Matthias Nießner. Pri3d: Can 3d priors help 2d representation learning? In *ICCV*, 2021. 68

[112] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, and Konrad Schindler Andreas Wieser. PREDATOR: Registration of 3D Point Clouds with Low Overlap. In *CVPR*, 2021. 49

[113] Xiangru Huang, Zhenxiao Liang, Chandrajit Bajaj, and Qixing Huang. Translation synchronization via truncated least squares. *NeurIPS*, 2017. 47

[114] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J Guibas, and Qixing Huang. Learning transformation synchronization. In *CVPR*, 2019. 47, 50

[115] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *CVPR*, 2020. 9, 29, 47

[116] Daniel F Huber and Martial Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 2003. 47

[117] Drew A Hudson and Christopher D Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019. 68

[118] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55. 41

[119] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773. If you use this software, please cite it as below. 122, 134, 135, 136

[120] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*, 2018. 8, 9

[121] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. *NeurIPS*, 2020. 45, 64, 68

[122] Varun Jampani, Kevis-Kokitsi Maninis, Andreas Engelhardt, Arjun Karpur, Karen Truong, Kyle Sargent, Stefan Popov, André Araujo, Ricardo Martin-Brualla, Kaushal Patel, Daniel Vlasic, Vittorio Ferrari, Ameesh Makadia, Ce Liu, Yuanzhen Li, and Howard Zhou. Navi: Category-agnostic image collections with high-quality 3d shape and pose annotations. In *NeurIPS Datasets and Benchmarks Track*, 2023. 93, 123

[123] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021. 68

[124] Andrew E Johnson. *Spin-images: a representation for 3-D surface matching*. PhD thesis, Carnegie Mellon University, 1997. 29

[125] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *TPAMI*, 1999. 7, 8, 29

[126] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2019. 72

[127] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2021. 53

[128] Justin Johnson, Lamberto Ballan, and Li Fei-Fei. Love thy neighbors: Image annotation by exploiting image metadata. In *ICCV*, 2015. 68

[129] Bela Julesz. Binocular depth perception of computer-generated patterns. *Bell System Technical Journal*, 1960. 3

[130] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 12, 32, 47, 49

[131] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6): 1–10, 2016. 9

[132] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017. 9

[133] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 68

[134] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Adv. Neural Inform. Process. Syst.*, 2020. 68, 86

[135] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *ICCV*, 2017. 27, 28, 29, 34

[136] Gyeongnyeon Kim, Wooseok Jang, Gyuseong Lee, Susung Hong, Junyoung Seo, and Seungryong Kim. Dag: Depth-aware guidance with denoising diffusion probabilistic models. *arXiv preprint arXiv:2212.08861*, 2022. 103

[137] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 16, 36, 53, 93

[138] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 91, 123, 134, 135, 136

[139] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NeurIPS*, 2015. 70

[140] Jan J Koenderink and Andrea J Van Doorn. The singularities of the visual mapping. *Biological cybernetics*, 24(1):51–59, 1976. 104

[141] Jan J Koenderink and Andrea J Van Doorn. The internal representation of solid shape with respect to vision. *Biological cybernetics*, 32(4):211–216, 1979. 104

[142] Jan J Koenderink and Andrea J Van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992. 89

[143] Jan J Koenderink, Andrea J Van Doorn, and Astrid ML Kappers. Pictorial surface attitude and local depth comparisons. *Perception & Psychophysics*, 58(2):163–173, 1996. 92

[144] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019. 73, 88, 90, 92, 103, 112

[145] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 113

[146] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 113

[147] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[148] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020. 8

[149] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *ICRA*. IEEE, 2011. 47

[150] L'ubor Ladickỳ, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 93, 124

[151] Axel Barroso Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.Net: Keypoint detection by handcrafted and learned cnn filters. In *ICCV*, 2019. 46

[152] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. In *BMVC*, 2019. 45

[153] Huu M Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. SDRSAC: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *CVPR*, 2019. 47

[154] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML - W*, 2013. 28

[155] Janghyeon Lee, Jongsuk Kim, Hyounguk Shon, Bumsoo Kim, Seung Hwan Kim, Honglak Lee, and Junmo Kim. Uniclip: Unified framework for contrastive language-image pre-training. In *NeurIPS*, 2022. 68

[156] Michelle A Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *IEEE Transactions on Robotics*, 2020. 107

[157] Karel Lenc and Andrea Vedaldi. Learning covariant feature detectors. In *ECCV*, 2016. 46

[158] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006. 109

[159] Martha Lewis, Qinan Yu, Jack Merullo, and Ellie Pavlick. Does clip bind concepts? probing compositionality in large image models. *arXiv preprint arXiv:2212.10537*, 2022. 130

[160] Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *ICCV*, 2023. 91

[161] Ang Li, Allan Jabri, Armand Joulin, and Laurens van der Maaten. Learning visual n-grams from web data. In *ICCV*, 2017. 68

[162] Aoxue Li, Tiange Luo, Zhiwu Lu, Tao Xiang, and Liwei Wang. Large-scale few-shot learning: Knowledge transfer with class hierarchy. In *CVPR*, 2019. 68

[163] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=OJLaKwiXSbx`. 87

[164] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 87

[165] Lei Li, Siyu Zhu, Hongbo Fu, Ping Tan, and Chiew-Lan Tai. End-to-end learning local multi-view descriptors for 3d point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 8, 27, 28, 29, 34

[166] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision Exists Everywhere: A Data Efficient Contrastive Language-Image Pre-training Paradigm. In *ICLR*, 2022. 68, 69, 75, 117

[167] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 45, 46, 125

[168] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In *NeurIPS*, 2022. 65, 77

[169] David C Lindberg. *Theories of Vision from al-Kindi to Kepler*. University of Chicago Press, 1976. 1, 2

[170] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 1989. 112

[171] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020. 126

[172] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023. 88, 103

[173] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 122, 134, 135, 136

[174] John Locke. *An Essay Concerning Human Understanding.* Glasgow: Printed by Robert Urie., 1759. 107

[175] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *ICLR*, 2018. 70

[176] Hugh Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 1981. 9, 29

[177] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 73, 93

[178] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 53, 73

[179] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 6, 7, 8, 9, 12, 20, 25, 32, 46, 49, 54, 127

[180] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *ICCV*, 2019. 9, 47

[181] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 64

[182] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *arXiv*, 2023. 95, 103

[183] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and egomotion from monocular video using 3d geometric constraints. In *CVPR*, 2018. 9, 10

[184] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 113

[185] Tomasz Malisiewicz and Alexei A Efros. Beyond categories: the Visual Memex model for reasoning about object relationships. In *Advances in Neural Information Processing Systems*, 2009. 107

[186] Stephane Georges Mallat. *Multiresolution representations and wavelets*. University of Pennsylvania, 1988. 1

[187] David Marr. *Vision: A computational investigation into the human representation and processing of visual information. MIT Press*. MIT Press, Cambridge, Massachusetts, 1982. 89

[188] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Royal Society of London*, 1979. 3

[189] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv*, 2020. 9

[190] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 88

[191] Iaroslav Melekhov, Zakaria Laskar, Xiaotian Li, Shuzhe Wang, and Juho Kannala. Digging into self-supervised learning of feature descriptors. In *3DV*, 2021. 46

[192] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004. 46

[193] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013. 71

[194] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 9

[195] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv prepreint arXiv:1908.10543*, 2019. x, 96, 97, 124, 129

[196] Shlok Mishra, Anshul Shah, Ankan Bansal, Abhyuday Jagannatha, Abhishek Sharma, David Jacobs, and Dilip Krishnan. Object-aware cropping for self-supervised learning. *arXiv preprint arXiv:2112.00319*, 2021. 68

[197] Hans P. Moravec. Rover visual obstacle avoidance. In *IJCAI*, 1981. 8, 29

[198] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. In *Eur. Conf. Comput. Vis.*, 2022. 67, 68, 72, 73, 78, 116

[199] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 2017. 58

[200] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31 (5):1147–1163, 2015. 47

[201] Alejandro Newell and Jia Deng. How useful is self-supervised pretraining for visual tasks? In *CVPR*, 2020. 68

[202] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019. 9

[203] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, December 2008. 113

[204] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2, 68

[205] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999. 131

[206] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, 2023. 87, 88, 91, 98, 103, 121, 125, 134, 135, 136

[207] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 50

[208] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 70

[209] Yookoon Park, Mahmoud Azab, Bo Xiong, Seungwhan Moon, Florian Metze, Gourab Kundu, and Kirmani Ahmed. Normalized contrastive learning for text-video retrieval. *EMNLP*, 2022. 68

[210] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 113

[211] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 53, 112

[212] Xiangyu Peng, Kai Wang, Zheng Zhu, Mang Wang, and Yang You. Crafting better contrastive views for siamese representation learning. In *CVPR*, pages 16031–16040, 2022. 68

[213] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 9

[214] Luigi Piccinelli, Christos Sakaridis, and Fisher Yu. idisc: Internal discretization for monocular depth estimation. In *CVPR*, 2023. 95, 127

[215] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4 (1):1–104, 2015. URL http://dx.doi.org/10.1561/2300000035. 9, 29

[216] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2022. 88, 103

[217] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023. 103

[218] Shengyi Qian, Linyi Jin, and David F. Fouhey. Associative3D: Volumetric Reconstruction from Sparse Views. In *ECCV*, 2020. 9

[219] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Int. Conf. Machine Learning*, 2021. 65, 67, 68, 73, 74, 78, 80, 81, 87, 90, 91, 112, 118, 119, 122, 134, 135, 136

[220] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. Dreambooth3d: Subject-driven text-to-3d generation. *ICCV*, 2023. 103

[221] Rene Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 7, 9, 20, 29, 32, 40, 47, 49

[222] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 91, 94, 123

[223] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 91, 94, 123, 134, 135, 136

[224] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv*, 2020. 16, 36, 53

[225] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084. ix, 69, 70, 71, 81, 114

[226] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, 2017. 46

[227] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2018. 46

[228] Barbara Roessle and Matthias Nießner. End2end multi-view feature matching using differentiable pose optimization. *arXiv preprint arXiv:2205.01694*, 2022. 47

[229] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 87, 91, 122, 134, 135, 136

[230] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 1

[231] Amir Rosenfeld and John K Tsotsos. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 9–16. IEEE, 2019. 28

[232] Karsten Roth, Oriol Vinyals, and Zeynep Akata. Integrating Language Guidance into Vision-based Deep Metric Learning. In *CVPR*, 2022. 68

[233] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011. 8

[234] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 1

[235] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 36

[236] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, 2009. 8, 29, 36, 39, 40

[237] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 87

[238] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013. 47

[239] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 2014. 29, 39, 40

[240] Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In *ECCV*, 2020. 68

[241] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 7, 9, 12, 17, 20, 46, 49, 54, 55, 56, 97, 124

[242] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *TPAMI*, 1997. 46

[243] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 7, 27, 46

[244] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos. In *CVPR*, 2017. 53

[245] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 87, 91, 122

[246] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Int. Conf. Comput. Vis.*, 2017. 89, 103

[247] Ramprasaath R Selvaraju, Karan Desai, Justin Johnson, and Nikhil Naik. Casting your model: Learning to localize improves self-supervised representations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11058–11067, 2021. 67, 68, 86

[248] Dandan Shan, Richard Ely Locke Higgins, and David Fouhey. Cohesiv: Contrastive object and hand embedding segmentation in video. In *NeurIPS*, 2021. 68

[249] Jinghuan Shang, Srijan Das, and Michael S Ryoo. Learning viewpoint-agnostic visual representations by recovering tokens in 3d space. *NeurIPS*, 2022. 68

[250] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPRW*, 2014. 1, 68, 90

[251] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual Captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2018. 73, 79

[252] Roger N Shepard and Susan Chipman. Second-order isomorphism of internal representations: Shapes of states. *Cognitive psychology*, 1(1):1–17, 1970. 2, 65, 89

[253] Roger N Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971. 89

[254] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 103

[255] Daeyun Shin, Zhile Ren, Erik B Sudderth, and Charless C Fowlkes. 3d scene reconstruction with multi-layer depth and epipolar transformers. In *ICCV*, 2019. 9

[256] Ayush Shrivastava and Andrew Owens. Contrastive random walk with layers. *In preparation*, 2024. 64

[257] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 93, 121, 124

[258] Linda B. Smith and Donald B. Katz. Chapter 12 - activity-dependent processes in perceptual and cognitive development. In Rochel Gelman and Terry Kit-Fong Au, editors, *Perceptual and Cognitive Development*, Handbook of Perception and Cognition, pages 413–445. Academic Press, San Diego, 1996. ISBN 978-0-12-279660-9. doi: https://doi.org/10.1016/B978-012279660-9/50030-0. URL https://www.sciencedirect.com/science/article/pii/B9780122796609500300. 107

[259] Linda B Smith, Swapnaa Jayaraman, Elizabeth Clerkin, and Chen Yu. The developing infant creates a curriculum for statistical learning. *Trends in Cognitive Sciences*, 2018. 107

[260] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846, 2006. 45, 46

[261] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016. 74

[262] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for language understanding. *NeurIPS*, 2020. ix, 70, 71, 81, 114

[263] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017. 32

[264] Elizabeth Spelke, Sang Ah Lee, and Véronique Izard. Beyond core knowledge: Natural geometry. *Cognitive science*, 34(5):863–884, 2010. 89, 92

[265] Charles Spence. Crossmodal correspondences: A tutorial review. *Attention, Perception, & Psychophysics*, 73:971–995, 2011. 65, 86

[266] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *ICCV*, 2017. 9

[267] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 9

[268] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 20

[269] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 122

[270] Jonathan C Stroud, Zhichao Lu, Chen Sun, Jia Deng, Rahul Sukthankar, Cordelia Schmid, and David A Ross. Learning video representations from textual web supervision. *arXiv preprint arXiv:2007.14937*, 2020. 68

[271] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *ACL*, 2019. 68

[272] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 48, 49, 55, 56, 68

[273] Igor Susmelj, Matthias Heller, Philipp Wirth, Prescott Jeremey, and Malte Ebner. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020. 116

[274] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment networks. In *ICLR*, 2018. 9

[275] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023. 88, 98, 103, 122, 128, 131

[276] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. In *ICLR*, 2019. 9

[277] TFDS. TensorFlow Datasets, a collection of ready-to-use datasets, 2023. URL `https://www.tensorflow.org/datasets`. 114

[278] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 2016. 68

[279] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *CVPR*, 2022. 88

[280] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv*, 2019. 8, 30

[281] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, 2020. 112

[282] Yonglong Tian, Olivier J Henaff, and Aäron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 68

[283] Yurun Tian, Bin Fan, and Fuchao Wu. L2-Net: Deep learning of discriminative patch descriptor in euclidean space. In *CVPR*, 2017. 46

[284] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM Workshop on 3D Object Retrieval*, 3DOR '10. Association for Computing Machinery, 2010. 29, 39, 40

[285] Philip HS Torr and David William Murray. The development and comparison of robust methods for estimating the fundamental matrix. *IJCV*, 1997. 9, 29

[286] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011. 106

[287] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit III: Revenge of the ViT. In *ECCV*, 2022. 90, 91, 121, 134, 135, 136

[288] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015. 9

[289] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *CVPR*, 2018. 8

[290] Michael Tye. *The imagery debate*. Mit Press, 2000. 2

[291] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 28

[292] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13 (04):376–380, 1991. 6, 32, 47, 49

[293] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 8, 9, 10

[294] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008. 41

[295] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based image description evaluation. In *CVPR*, 2015. 70

[296] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, 2018. 113

[297] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. In *ArXiv*, 2017. 9, 10

[298] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2, 68

[299] Matthew Walmer, Saksham Suri, Kamal Gupta, and Abhinav Shrivastava. Teaching matters: Investigating the role of supervision in vision transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023. 87, 126

[300] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023. 103

[301] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *NeurIPS*, 2020. 81, 114

[302] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 64, 68

[303] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 73, 112

[304] Yue Wang and Justin Solomon. Prnet: Self-supervised learning for partial-to-partial registration. *NeurIPS*, 2019. 9, 29

[305] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019. 8, 9, 29

[306] Ziming Wang, Xiaoliang Huo, Zhenghao Chen, Jing Zhang, Lu Sheng, and Dong Xu. Improving rgb-d point cloud registration by learning multi-scale local linear transformation. In *ECCV*, 2022. 47

[307] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 113

[308] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 120

[309] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS Workshop on ImageNet: Past, Present, and Future*, 2021. ix, 70, 78, 81, 118, 119

[310] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 10, 13, 14

[311] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *ECCV*, 2014. 45, 46

[312] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`. 120

[313] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *CVPR*, 2023. 120, 134, 135, 136

[314] Xiaoshi Wu, Hadar Averbuch-Elor, Jin Sun, and Noah Snavely. Towers of babel: Combining images, language, and 3d geometry for learning multimodal vision. In *ICCV*, 2021. 68

[315] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018. 2, 66, 68, 74

[316] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014. 124

[317] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 113

[318] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2020. 3, 67, 68, 69, 76, 86

[319] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020. 27, 28, 29, 30, 31, 34, 35

[320] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Yi Wang, and Zhangyang Wang. Neurallift-360: Lifting an in-the-wild 2d photo to a 3d object with 360deg views. In *CVPR*, 2023. 103

[321] Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying clip data. *arXiv preprint arXiv:2309.16671*, 2023. 91

[322] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 68

[323] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. ODISE: Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. In *CVPR*, 2023. 88, 91, 122

[324] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *ICCV*, 2015. 68

[325] Heng Yang, Wei Dong, Luca Carlone, and Vladlen Koltun. Self-supervised geometric perception. In *CVPR*, 2021. 46

[326] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. FILIP: Fine-grained interactive language-image pre-training. In *ICLR*, 2022. 68

[327] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *ECCV*, 2018. 8, 27, 28, 29, 34

[328] Zi Jian Yew and Gim Hee Lee. RPM-Net: Robust Point Matching using Learned Features. In *CVPR*, 2020. 9, 29, 47

[329] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016. 7, 8

[330] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 12, 46, 49

[331] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *ICCV*, 2023. 125

[332] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 9, 10

[333] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2014. 68

[334] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *ECCV*, 2020. 9, 29

[335] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 8

[336] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *CVPR*, 2019. 68

[337] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 15, 28, 35, 40, 46

[338] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022. 73

[339] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *ICCV*, 2023. 90, 134, 135, 136

[340] Guanqi Zhan, Chuanxia Zheng, Weidi Xie, and Andrew Zisserman. What does stable diffusion know about the 3d scene?, 2023. 103, 122, 126, 131

[341] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. In *NeurIPS*, 2023. 88, 95, 98, 103, 122, 124, 126, 128, 131

[342] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv*, 2020. 9

[343] Kaifeng Zhang, Yang Fu, Shubhankar Borse, Hong Cai, Fatih Porikli, and Xiaolong Wang. Self-supervised geometric correspondence for category-level 6d object pose estimation in the wild. In *ICLR*, 2023. 103

[344] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 2, 68

[345] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *ICLR*, 2019. 70

[346] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *arXiv preprint arXiv:2101.02691*, 2021. 30, 31, 34

[347] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994. 9, 28, 29

[348] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial intelligence*, 1995. 9, 29, 47

[349] Wenliang Zhao, Yongming Rao, Zuyan Liu, Benlin Liu, Jie Zhou, and Jiwen Lu. Unleashing text-to-image diffusion models for visual perception. *arXiv preprint arXiv:2303.02153*, 2023. 103, 122, 131

[350] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3D Point Capsule Networks. In *CVPR*, 2019. 29, 40

[351] Mingkai Zheng, Fei Wang, Shan You, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Weakly supervised contrastive learning. In *ICCV*, 2021. 68

[352] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *International Conference on Learning Representations (ICLR)*, 2022. 91, 121, 134, 135, 136

[353] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv*, 2018. 15, 16, 21, 35, 36, 53, 57

[354] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 8, 9, 10, 14, 46

[355] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *CVPR*, 2016. 68