Unsupervised Image Analysis by Synthesis

Tom Monnier

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Computer Science - Signal and Image Processing
from
École des Ponts ParisTech, France

Presented on December 4th, 2023 to a committee consisting of:

Matthieu Cord	Sorbonne University	President
Frédo Durand	Massachusetts Institute of Technology	Reviewer
Niloy J. Mitra	University College London	Reviewer
Katerina Fragkiadaki	Carnegie Mellon University	Examiner
Mathilde Caron	Google Research	Examiner
Mathieu Aubry	École des Ponts ParisTech	Supervisor

École des Ponts ParisTech
LIGM - IMAGINE - A3SI
6 Avenue Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée CEDEX 2
France

À mes deux grands-pères, Alain Monnier et Liem Dang.

Abstract

The goal of this thesis is to develop machine learning approaches to analyze collections of images without annotations. Specifically, given a collection of images as input, the machine should discover analysis concepts related to properties of the world, such as the object class or the underlying 3D geometry, without being taught this concept via manually annotated examples. Advances in this area hold promises for industrial applications related to 3D (*e.g.*, reconstructing a photorealistic scene with 3D actionable components for video games) where annotating examples to teach the machine is difficult, as well as more specific applications addressing a particular professional or personal need (*e.g.*, analyzing the character evolution from 12th century documents) where spending significant effort on annotating large-scale datasets is debatable. The central idea of this dissertation is to build machines that learn to analyze an image collection by synthesizing the images in the collection. We focus on three important challenges to learning image analysis by synthesis.

The first challenge is to model the variability in different images of objects reflecting the same object class. One of the most basic form of image analysis is to group images into object classes related to different concepts. A simple analysis-by-synthesis approach to this problem is the classical K-means algorithm introduced by James MacQueen in 1967, where instances of each class are approximated by the average of the class. However, this kind of framework fails when applied to images, where instances within each class demonstrate a strong variability.

The second challenge is to discover elements in the images, in such a way that elements correspond to interpretable concepts like multiple objects in the image. Discovering elements in images is typically tackled with bottom-up approaches that take advantage of low-level image cues to compute meaningful image regions. An early example is the work of Brice and Fennema in 1970, which breaks the image into atomic regions of uniform gray scale then use heuristics to group them. Yet, because these low-level image cues are generally handcrafted, such approaches typically struggle when applied to real-world images.

The third challenge is to recover 3D structure from 2D images. The perception of 3D from images can be traced back to the thesis of Lawrence Roberts on Blocks World in 1963. In this work, polyhedral 3D blocks are computed by detecting and matching the edges of an image depicting a simple scene made of textureless polyhedral shapes. Today, state-of-the-art methods for real-world scenes typically represent scenes through neural network weights, which are

difficult to interpret and manipulate. Approaches representing scenes with meshes carry a better interpretability, yet they typically compute meshes from annotations indicating the object region and do not model the scene as a composition of 3D elements. To achieve our goal and address these challenges, we present three key contributions.

The first contribution is a new conceptual approach to object class modeling. We propose to represent the class of an image, a 2D object or a 3D shape, with a prototype that is transformed using deep learning to model the different instances within the class. Specifically, we design meaningful transformations (*e.g.*, geometric deformations or color variations) and use neural networks to predict the transformations that make the prototype similar to what is observed in a given image. We demonstrate the effectiveness of this idea not only to cluster images, but also to reconstruct 3D objects from single-view images. We obtain performances on par with the best state-of-the-art methods, which all leverage handcrafted features or annotations.

The second contribution is a new way to discover elements in a collection of images. We propose to represent an image collection by a set of learnable elements composed together to synthesize the images and optimized by gradient descent. We validate this idea by discovering 2D elements related to different objects in a large image collection. Our approach has performances similar to the best concurrent methods which synthesize images with neural networks, and ours comes with better interpretability. We also demonstrate that this idea can be used to discover 3D elements related to simple primitive shapes, given as input a collection of images depicting a scene from multiple viewpoints.

The third contribution is more technical and consist in a new way to compute differentiable mesh rendering. Specifically, we formulate the differentiable rendering of a 3D mesh as the alpha compositing of the mesh faces in a decreasing depth order. Compared to prior works, this formulation is key to enable us to learn 3D meshes without requiring object region annotations. In addition, it allows us to seamlessly introduce the possibility to learn transparent meshes, which we design to model a scene as a composition of a variable number of meshes.

Keywords: Image analysis, Unsupervised learning, Analysis by synthesis, Deep learning

Résumé

Le but de cette thèse est de développer des approches d'intelligence artificielle (IA) pour analyser des collections d'images sans annotations. Concrètement, étant donné une collection d'images, l'IA doit découvrir des concepts d'analyse reliés aux propriétés du monde, telles que la classe d'un objet ou la géométrie 3D sous-jacente, sans que ce concept soit enseigné via des exemples manuellement annotés. Des avancées dans ce domaine sont prometteuses pour des applications industrielles reliées à la 3D (*e.g.*, reconstruire une scène photoréaliste avec des composantes 3D manipulables pour les jeux vidéo) où annoter des exemples pour entraîner l'IA est difficile, mais aussi pour des applications plus spécifiques répondant à un besoin professionnel ou personnel particulier (*e.g.*, analyser l'évolution des caractères dans les documents du 12ème siècle) où employer des efforts conséquents pour annoter de larges bases de données pose question. L'idée centrale de cette dissertation est de construire des IA qui apprennent l'analyse d'une collection d'images en synthétisant les images dans la collection. Nous nous concentrons sur trois problèmes importants à l'analyse d'image par synthèse.

Le premier problème est de modéliser la variabilité au sein d'images différentes des objets reflétant la même classe d'objet. Une des formes les plus basiques de l'analyse d'image est de regrouper des images dans des classes d'objet reliées à des concepts différents. Une approche simple d'analyse par synthèse pour cette tâche est l'algorithme classique K-means introduit par James MacQueen en 1967, où les instances de chaque classe sont représentées par la moyenne de la classe. Cependant, de telles approches échouent lorsqu'elles sont appliquées aux images, où les instances au sein de chaque classe possèdent une grande variabilité.

Le deuxième problème est de découvrir des éléments dans les images, d'une telle manière que les éléments correspondent à des concepts interprétables comme plusieurs objets dans l'image. Découvrir des éléments dans des images est typiquement résolu avec des approches "bottom-up" utilisant des représentations d'image bas niveau pour calculer des régions d'images pertinentes. Un exemple clé est le travail de Brice et Fennema en 1970, qui décompose une image en régions atomiques de gris uniforme et utilise des heuristiques pour les regrouper. Cependant, ces représentations d'image bas niveau sont souvent modélisées à la main, donc ces approches ont du mal à être appliquées aux images naturelles.

Le troisième problème est de retrouver la structure 3D à partir d'images 2D. La perception de la 3D à partir d'images peut être retracée à la thèse de Lawrence Roberts sur Blocks World

en 1963. Dans ce travail, des blocs 3D polyédraux sont calculés en détectant et alignant les contours d'une image illustrant une scène simple composée de formes polyédrales. Aujourd'hui, l'état de l'art représente typiquement les scènes avec les poids d'un réseau de neurones, qui sont difficilement interprétables et manipulables. Les approches représentant les scènes avec des meshes ont une meilleure interprétabilité, mais elles calculent les meshes à partir d'annotations indiquant les régions de l'objet et ne modélisent pas une scène comme une composition d'éléments 3D. Pour atteindre notre but et résoudre ces problèmes, nous présentons trois contributions clés.

La première contribution est une nouvelle approche conceptuelle à la modélisation de classe d'objets. Nous proposons de représenter la classe d'une image, d'un objet 2D ou d'une forme 3D, avec un prototype qui est transformé par apprentissage profond pour modéliser les différentes instances au sein de la classe. Plus spécifiquement, nous introduisons des transformations concrètes (*e.g.*, des déformations géométriques ou des variations de couleurs) et utilisons des réseaux de neurones pour prédire les transformations qui rendent le prototype similaire à ce qui est observé dans une image donnée. Nous démontrons l'efficacité de cette idée non seulement pour regrouper des images, mais aussi pour reconstruire des objets 3D à partir d'une seule image. Nous obtenons des performances égales aux meilleures méthodes, qui utilisent toutes des représentations d'image ad-hoc ou des annotations.

La deuxième contribution est une nouvelle manière de découvrir des éléments dans une collection d'images. Nous proposons de représenter une collection d'images par un ensemble d'éléments apprenables, composés pour synthétiser les images et optimisés par descente de gradient. Nous validons cette idée en découvrant des éléments 2D reliés à des objets différents dans une grande collection d'images. Notre approche a des performances semblables aux meilleures méthodes qui synthétisent les images par réseaux de neurones, et est plus interprétable. Nous démontrons aussi que cette idée peut être utilisée pour découvrir des éléments 3D reliés à des formes primitives simples étant donné une collection d'images illustrant une scène via différents points de vue.

La troisième contribution est plus technique et consiste en une nouvelle approche pour calculer le rendu différentiable d'un mesh. Plus spécifiquement, nous formulons le rendu différentiable d'un mesh comme l'alpha composition des faces du mesh par ordre de profondeur décroissante. Comparée aux travaux précédents, cette formulation est clé pour apprendre des meshes sans utiliser des annotations représentant les régions d'objet. En outre, cette formulation nous permet d'introduire la possibilité d'apprendre des meshes transparents, que nous utilisons pour modéliser une scène comme une composition d'un nombre variable de meshes.

Mots clés : Analyse d'image, Apprentissage non supervisé, Analyse par synthèse, Apprentissage profond

Remerciements

Merci aux membres du jury Matthieu Cord, Frédo Durand, Niloy Mitra, Katerina Fragkiadaki et Mathilde Caron d'avoir lu et commenté ce manuscrit ainsi que d'avoir participé à l'évaluation de la proposition de thèse.

Je souhaite tout d'abord remercier mon directeur de thèse, Mathieu, sans qui ce fantastique voyage dans le monde de la recherche n'aurait jamais eu lieu. Merci de m'avoir fait confiance, ainsi que d'avoir partagé et transmis ta passion pour les pixels. Merci pour ta bienveillance et ton soutien tout au long de ces quatre années, que ce soit pour ton accompagnement les soirs de rendu ou pour continuellement te soucier du bien-être de tes étudiants. Enfin, merci pour ton extrême rigueur mathématique et merci d'avoir rendu cette expérience si riche humainement et intellectuellement.

Je voudrais ensuite remercier toutes les personnes du laboratoire Imagine, avec qui j'ai partagé de très beaux moments et qui ont contribué à rendre ces années de thèse inoubliables. Merci tout d'abord aux enseignants chercheurs Pascal, Bertrand, Renaud, Chaohui, David, Gül, Vincent et Loïc ; merci d'avoir fait du laboratoire un endroit agréable et épanouissant pour les étudiants. Merci à Isabelle pour ta gentillesse et ta réactivité dans toutes les démarches administratives. Merci ensuite aux étudiants qui m'ont accueilli et aidé à faire mes premiers pas, en particulier Thibault G., Abderahmane, Pierre-Alain, Théo, Xi, Robin, Yang, Rahima, Simon, Othman, Liza, Sophie. Merci notamment à François pour m'avoir partagé et transmis ta passion pour la 3D, ainsi que pour tous ces échanges scientifiques et discussions de vie. Merci aux étudiants qui ont rejoint le voyage et m'ont accompagné, en particulier Elliot, Philippe, Victor, Romain, Mathis, Thibaut I., Yannis, Nicolas D., Georgy, Hugo, Michael, Lucas, Antoine, Monika, Nguyen, Nicolas G., Julien, Sonat, Charles et Hannah. Je souhaite aussi remercier toutes les personnes avec qui j'ai eu la chance de collaborer en dehors du laboratoire Imagine et qui m'ont beaucoup appris : Jean, Matt, Tomasz, Alyosha, Angjoo et Jake.

Enfin, j'aimerais remercier mes proches qui m'ont soutenu tout au long de ce voyage. Merci à mes parents, Catherine et Jean, mes soeurs, Kim-Lou et Lily, et mon frère Yann pour votre amour et soutien depuis toujours. Merci aussi à ma belle-famille Gabriele et Laura pour m'avoir chaleureusement accueilli pendant les longs mois de confinement. Je souhaite enfin remercier mon épouse Clara qui a été un réel pilier au cours de ce voyage. Merci d'être à mes côtés, de m'apporter ton soutien indéfectible, ta joie et ton amour.

Contents

Abstract iv

Résumé vi

Remerciements ix

1 Introduction 1

1.1 Goal . . . 1
1.2 Motivations . . . 3
1.3 Approach and context . . . 6
1.4 Challenges . . . 7
1.5 Contributions . . . 8
1.6 Thesis outline . . . 10
1.7 Publication list . . . 12

2 Background 13

2.1 Preliminary . . . 13
2.1.1 Machine learning . . . 14
2.1.2 Deep learning . . . 14
2.1.3 Feature-based vs synthesis-based . . . 17
2.1.4 General-purpose image features . . . 17
2.1.5 Mesh-based differentiable rendering . . . 20
2.2 Image clustering . . . 20
2.2.1 Feature-based methods . . . 21
2.2.2 Synthesis-based methods . . . 23
2.3 Object discovery . . . 24
2.3.1 Feature-based methods . . . 24
2.3.2 Synthesis-based methods . . . 26
2.4 Single-view reconstruction . . . 27

2.4.1 Feature-based methods 27
2.4.2 Synthesis-based methods 28
2.5 Multi-view 3D decomposition 29
2.5.1 Multi-view stereo 30
2.5.2 3D decomposition 31

3 Deep Transformation-Invariant Clustering 35
3.1 Introduction 36
3.2 Approach 37
3.2.1 DTI framework 37
3.2.2 Application to K-means and GMM 39
3.3 Learning image transformations 40
3.3.1 Architecture and transformation modules 40
3.3.2 Training 42
3.4 Experiments 43
3.4.1 Analysis and comparisons 44
3.4.2 Application to web images 47
3.5 Conclusion 47

4 Discovering Objects With Sprite Modeling 49
4.1 Introduction 49
4.2 Approach 51
4.2.1 Image formation model 51
4.2.2 Learning 53
4.3 Experiments 55
4.3.1 Multi-object synthetic benchmarks 56
4.3.2 Real image benchmarks 59
4.4 Conclusion 62

5 Single-View Reconstruction Without Supervision 65
5.1 Introduction 65
5.2 Related work 68
5.3 Approach 69
5.3.1 Structured autoencoding 69
5.3.2 Unsupervised learning with cross-instance consistency 71
5.3.3 Alternate 3D and pose learning 73
5.4 Experiments 74

5.4.1 Evaluation on the ShapeNet benchmark . . . 74
5.4.2 Results on real images . . . 75
5.4.3 Ablation study . . . 79
5.5 Conclusion . . . 79

6 Differentiable Blocks World **81**
6.1 Introduction . . . 81
6.2 Approach . . . 82
6.2.1 Parametrizing a world of blocks . . . 83
6.2.2 Differentiable rendering . . . 85
6.2.3 Optimizing a differentiable blocks world . . . 86
6.3 Experiments . . . 88
6.3.1 DTU benchmark . . . 88
6.3.2 Real-life data and applications . . . 89
6.3.3 Analysis . . . 91
6.4 Conclusion . . . 92

7 Conclusion **95**
7.1 Contributions summary . . . 95
7.2 Future works . . . 96

Appendices **99**
A Deep Transformation-Invariant Clustering . . . 99
A.1 Dataset descriptions . . . 99
A.2 Transformation invariance . . . 100
A.3 Model insights . . . 101
B Discovering Objects With Sprite Modeling . . . 105
B.1 Semantic segmentation results . . . 105
B.2 Model insights . . . 105
B.3 Training details . . . 106
B.4 Additional qualitative results . . . 110
C Single-View Reconstruction Without Supervision . . . 114
C.1 Custom differentiable rendering . . . 114
C.2 Model insights . . . 115
C.3 Quantitative evaluation . . . 118
C.4 Implementation details . . . 119
D Differentiable Blocks World . . . 124

D.1 Additional video results 124
D.2 DTU benchmark 124
D.3 Implementation details 124

Bibliography **129**

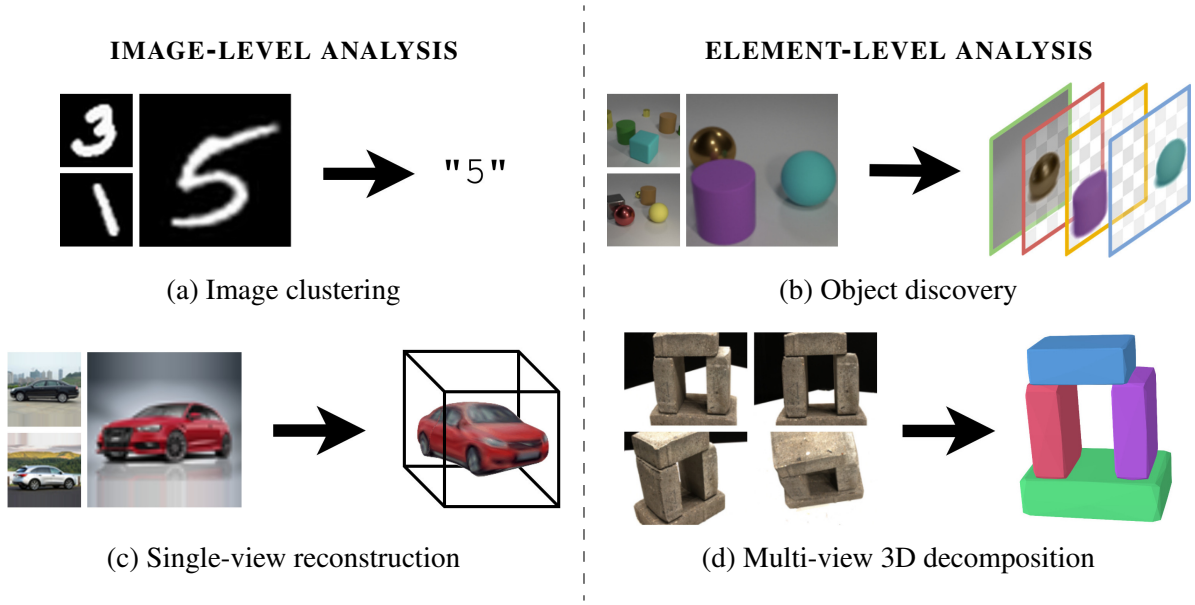
Chapter 1

Introduction

1.1 Goal

The central goal of this thesis is to develop machine learning approaches to analyze collections of images without annotations. Concretely, given a collection of images as input, the machine should discover analysis concepts related to properties of the world, such as the object classes, their location or the underlying 3D geometry, without being explicitly taught this concept via manually annotated examples. We focus on two analysis aspects, namely image-level analysis and element-level analysis, that we illustrate in Figure 1.1 and that we define next.

Image-level analysis. We tackle problems that consider a set of images as input and aim at associating to each image an image-level output. In contrast with prior works relying on strong learning assumptions like off-the-shelf features or annotations, we aim at building models that are able to learn from the raw images only. We study two computer vision problems related to image-level analysis. We first focus on the classical task of *image clustering*, whose goal is to split images into similarity groups, or equivalently to associate a group to each image. In Chapter 3, we present a clustering method computing similarities directly in the input image space and matching the performances of state-of-the-art feature-based approaches. We then focus on the task of *single-view reconstruction* (SVR), which aims at computing the 3D geometry depicted by each image in the collection. SVR is challenging and typically requires manual labels associated to each image during training, like the ground-truth 3D model, the viewpoint of the camera that took the picture or the image region representing the object. In Chapter 5, we design an SVR system that does not require any annotations; it takes as input a set of images depicting different instances of an object class and outputs the 3D reconstruction for each image. We obtain performances that are on par with state-of-the-art methods that are manually supervised.



(a) Image clustering (b) Object discovery

(c) Single-view reconstruction (d) Multi-view 3D decomposition

Figure 1.1: **Goal.** We aim at developing unsupervised machine learning approaches to analyze image collections without annotations. We focus on two image analysis aspects, namely image-level analysis and element-level analysis, that we study through four computer vision problems. *Image-level analysis* aims at computing for each image a single image-level output, like **(a)** a group for image clustering or **(c)** a 3D reconstruction for single-view reconstruction. *Element-level analysis* instead aims at finding elements in each image and computing for each element an element-level output, like **(b)** an object mask for object discovery or **(d)** a primitive 3D shape for multi-view 3D decomposition.

Element-level analysis. We tackle problems that consider a set of images as input and aim at finding elements in each image and associating to each element an element-level output. Compared to previous works modeling images implicitly with neural networks, we aim at representing an image collection with an explicit set of learnable elements that are interpretable. We study two computer vision problems related to element-level analysis. First, we tackle the task of *object discovery*, whose goal is to locate and identify the recurrent objects in the image collection. In Chapter 4, we introduce a new image modeling consisting in the explicit alpha compositing of learnable sprites, that matches the state-of-the-art performances of implicit image models. Then, we focus on the task which aims at decomposing a scene represented by a collection of multiple view images into a set of 3D geometric primitives. This task has rarely been studied and we refer to it as *multi-view 3D decomposition*. In Chapter 6, we present an approach that computes a primitive-based 3D reconstruction by optimizing learnable and textured superquadric meshes through image rendering. We report performances that are much better than state-of-the-art methods which all fit 3D primitives in a pre-computed 3D point cloud of the scene.

1.2 Motivations

Compared to machines that are explicitly taught how to analyze images via manually annotated examples, analyzing image collections without any annotations is motivated by several theoretical advantages as well as a wide range of applications.

Advantages. Analyzing images without annotations is related to machine learning problems that are *unsupervised*, in contrast with the *supervised* problems, where machines can learn via the supervision of annotated examples. Such an unsupervised setting is motivated by four theoretical benefits.

Flexibility. Supervised approaches often require experts to annotate a considerable amount of data to teach a machine to predict the desired output. Depending on the problem, these annotations are typically the result of a costly and tedious process that can be difficult to scale. On the contrary, methods learned without manual supervision can be seamlessly applied to new data, without spending any effort on building the annotations. For example, our clustering approach developed in Chapter 3 for images was adapted to 3D and audio data by Loiseau et al. [2021, 2022a].

Generalization. In some cases, annotations can be extremely challenging to acquire. For example, defining the 3D reconstructions for a large collection of real-world images is difficult; the most closely related datasets are either synthetically generated using simulation engines (*e.g.*, Chang et al. [2015]; Roberts et al. [2021]) or limited in diversity and size (*e.g.*, Reizenstein et al. [2021]; Collins et al. [2022]). Consequently, approaches learned on such targeted data typically generalizes poorly to more complex data. This is not the case for unsupervised methods that can learn from the vast amount of available data. For example, the unsupervised single-view reconstruction approach we present in Chapter 5 is not limited to the 3D reconstructions of synthetic car images like the supervised approach from Niemeyer et al. [2020]; it can successfully be applied to real-world car images.

Robustness. The annotations necessary to learn supervised machines are typically acquired by a manual or automatic process that makes approximations and errors when building the ground-truth annotations. This introduces noise in the dataset and potentially harms the performances of the learned system. Conversely, unsupervised approaches do not suffer from noisy ground truth. For example, in Chapter 6, we accurately find 3D scene decomposition by fitting textured primitives through rendering in the multiple views of real-world scenes like DTU [Jensen et al., 2014]. Prior state-of-the-art methods

(a) Augmented reality

(b) Virtual reality

(c) Historical data analysis

(d) Earth imagery analysis

Figure 1.2: **Applications.** Examples of applications where unsupervised image analysis hold high promises includes: **(a)** augmented reality (© Andrey Popov/Adobe Stock), **(b)** virtual reality [Kerbl et al., 2023], **(c)** historical data analysis [Siglidis et al., 2023], and **(d)** Earth imagery analysis [Vincent et al., 2023]

like Ramamonjisoa et al. [2022]; Liu et al. [2022] instead fall short for this real-world scenario, because primitives are computed from the 3D ground-truth point cloud which is extremely noisy and incomplete.

Less human bias. Another theoretical benefit that is important to emphasize, is the fact that unsupervised methods are less prone to human bias than supervised ones. Indeed, supervised methods are learning from annotations which are typically made by a group of individuals with different environments, cultural backgrounds and experiences. During the annotation process, these individuals may unconsciously introduce their preferences or prejudices that will influence the output of the resulting machine. These biases can lead to systems that are inaccurate or unfair, and thus raise ethical concerns. Although human bias can still be introduced during the data collection, unsupervised approaches are not influenced by manual and bias-prone annotations.

Applications. Compared to supervised approaches to image analysis, discovering analysis concepts from image collections holds bigger promises for applications where annotations are scarce or impossible to acquire. Figure 1.2 shows four examples of applications.

Augmented reality. Augmented reality corresponds to the addition of artificial effects or artificial 3D assets to our visual world captured by various sensors as illustrated in Figure 1.2a. Today, the creation of artificial 3D assets is a manual task requiring significant efforts from a graphic designer, who typically takes inspiration from a drawing or an image. One way to facilitate this process is to build a system that automatically generates a 3D asset from a given image, as formulated by the single-view reconstruction problem. Collecting annotations to solve this task in a supervised manner is challenging, as it requires to create a realistic 3D asset that accurately corresponds to each image. Consequently, supervised methods like Choy et al. [2016]; Groueix et al. [2018]; Wang et al. [2018] are typically limited to synthetic annotated datasets and struggle to be applied to real-world images. Learning to solve this task without annotations is thus especially appealing. In Chapter 5, we present a completely unsupervised approach that learns to reconstruct 3D objects from real-world images.

Virtual reality. The goal of virtual reality is to create a fully immersive and virtual world that individuals can interact with through their avatars. An important aspect of this application is the reconstruction of 3D scenes that are as realistic as possible. However, common libraries of 3D assets like Sketchfab or TurboSquid are typically costly to collect, orders of magnitudes smaller than standard image databases, and corresponding 3D models typically lack realism. Besides, creating a realistic 3D scene from scratch requires a tremendous amount of work from a graphic designer. In contrast, recent neural advances in 3D reconstruction from multi-view images like Mildenhall et al. [2020]; Müller et al. [2022]; Kerbl et al. [2023] enable to reconstruct a 3D scene with outstanding realism in a few minutes (Figure 1.2b). One shortcoming of such approaches is that they do not model objects or elements, making it difficult to understand and manipulate the resulting 3D scene. In Chapter 6, we solve this shortcoming by presenting an approach which outputs a 3D reconstruction made of simple geometric primitives. In particular, this primitive-based 3D reconstruction is actionable: it allows us to perform element-based physical simulations and scene editing without any effort.

Historical data analysis. The main difficulty with historical data compared to everyday-life data like natural images, is the need of qualified experts to annotate and analyze them. As a result, datasets with annotations are hard to gather and when they exist, they are typically of relatively small scale. This hinders the use of supervised machine learning methods to facilitate the analysis process. For example, analyzing the evolution of characters from historical handwritten charters remains a difficult yet important problem for paleography. To tackle this specific task, the work of Siglidis et al. [2023] on text

line analysis (Figure 1.2c) is building on top of the unsupervised object discovery model introduced in Chapter 4.

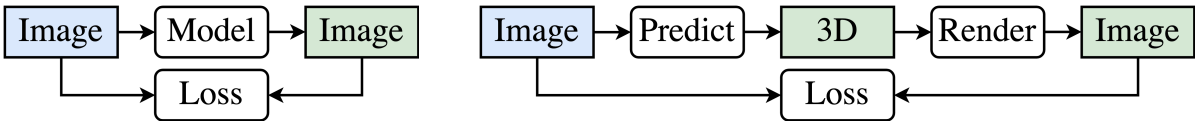
Earth imagery analysis. Analyzing images of Earth is at the core of many critical applications like climate change monitoring, agricultural-related predictions or even urbanization studies. Earth imagery typically corresponds to large-scale datasets of very high-resolution images that evolve through time. Hence, the process of collecting annotations in this scenario is even more tedious and costly than natural images, and having methods that learn without supervision becomes even more appealing. For example, Vincent et al. [2023] analyze agricultural images without annotations (Figure 1.2d) by leveraging the clustering framework described in Chapter 3.

1.3 Approach and context

Analyzing image collections without annotations is a broad and difficult problem. Early methods typically focus on building intermediate image representations, or *features*, that are used to run unsupervised analysis models. Such a featured-based approach is a two-stage process that often requires to carefully tune the representation depending on the analysis task. In addition, because features cannot be easily associated to meaningful concepts of our world, the results output by these methods are difficult to interpret.

In this dissertation, the central idea is to build machines that learn to analyze an image collection by synthesizing the images in the collection. More concretely, given a collection of images and an analysis task, we first design a differentiable image formation model that explicitly exhibits the desired analysis concept. Then, we leverage deep learning softwares and their optimization tools to update our model parameters by minimizing a loss between images and their reconstructions through gradient descent. Figure 1.3 illustrates the framework with the general setup and an applied example. This idea is related to the paradigm of *analysis-by-synthesis*, where an analysis task is performed by synthesizing the data. In machine learning terms, it is a generative approach to solving a problem, as opposed to a discriminative one.

Learning image analysis by synthesis has several advantages. First, such an approach is unsupervised and data-driven: models are automatically learned from the data, thus removing the need for annotations and handcrafted and data-specific priors. Second, because the analysis concept is explicitly modeled within the image formation process, it leads to results that are explainable. In contrast with deep black-box systems, this is an important aspect with two major benefits: (i) it helps decision-making because the end-user understands why the system made this prediction, and (ii) it favors further improvements by experts because the system is intuitive. Finally, recent approaches showcase that unsupervised deep learning is powerful and



(a) General setup (b) Example for a single-view reconstruction model

Figure 1.3: **Approach.** We propose to learn analysis models by synthesizing images, a framework known as *analysis-by-synthesis*. **(a)** Specifically, for each analysis problem, we design a model using a deep learning software like PyTorch and update our model by minimizing a loss between images (input, in blue) and their reconstructions (output, in green) through gradient descent. **(b)** We illustrate the framework with the example of an abstract model designed for single-view 3D reconstruction.

effective, in the field of computer vision [Mildenhall et al., 2020; Ramesh et al., 2021] but also in other domains like natural language processing [Radford et al., 2018, 2019].

1.4 Challenges

Learning analysis models by synthesis is difficult because it requires the design of a learnable image generation model that explicitly exhibits the desired analysis concept, *e.g.* the 3D geometry depicted by the image. This is typically not the case for standard generative models like autoencoders [Kramer, 1991; Hinton et al., 2006], variational autoencoders [Kingma and Welling, 2014], generative adversarial networks [Goodfellow et al., 2014] or diffusion models [Ho et al., 2020] which generate images implicitly through neural networks. We identify the following three independent challenges to learning image analysis by synthesis.

Modeling objects from the same class. One of the most basic form of image analysis is to group images or objects into classes related to different concepts, like a cat or dog concept. Learning this task through generation not only requires modeling different object classes during the generation process, but also addressing the variability of objects belonging to the same class. A simple approach in this direction is the classical K-means algorithm introduced by MacQueen [1967], where each class is represented by a different prototype and each sample is assigned to the closest prototype. However, such a framework does not model object variations within a class, and thus typically fails in the case of images, where objects exhibit a strong variability.

Discovering elements. Finding elements in an image that correspond to interpretable concepts like multiple objects is a difficult problem that can be traced back to the work of Brice and Fennema [1970]. Beyond the association of a single output to a single image, discovering elements by synthesis requires to explicitly model different elements during the image generation, which comes with three main hurdles. First, the number of elements to discover is

not known *a priori*, and it typically differs from one image to another. Hence, this involves modeling a variable number of elements which is a hard task as it involves optimizing over a discrete variable. Some recent works tackle the problem using reinforcement learning [Tulsiani et al., 2017a] or probabilistic approximations [Paschalidou et al., 2019]. Second, modeling elements increases ambiguity as it multiplies the number of possibilities to synthesize the exact same image. For example, the image of an 8 digit can either be associated to a single 8 digit or to two 0 digits. Third, composing elements during image generation naturally introduces occlusions between the elements which are difficult to handle.

Learning 3D from 2D. The perception of 3D from 2D images is a long standing problem that can be dated back to the early Blocks World model of Roberts [1963]. It is a challenging and ill-posed task because the objective is to recover the 3D information that has been lost while generating the image by 3D-to-2D projection. The 3D perception from images is mainly tackled in two settings, namely *multi-view stereo* which aims at reconstructing the 3D geometry of a scene represented by multiple images taken from different viewpoints, and *single-view reconstruction* which aims at reconstructing the 3D geometry of a scene depicted by a single image. In both cases, recent advances [Mescheder et al., 2019; Niemeyer et al., 2020; Mildenhall et al., 2020] enable the 3D reconstruction of objects and scenes that are realistic and accurate. However, they are typically represented with weights of neural networks, thus making it difficult to interpret, manipulate and integrate the results into standard computer graphics pipelines for further applications. A mesh is a much more appealing 3D representation from that perspective, yet most advanced mesh-based works [Liu et al., 2019; Goel et al., 2020, 2022] compute meshes from object mask annotations and they do not model the scene as a composition of 3D elements.

1.5 Contributions

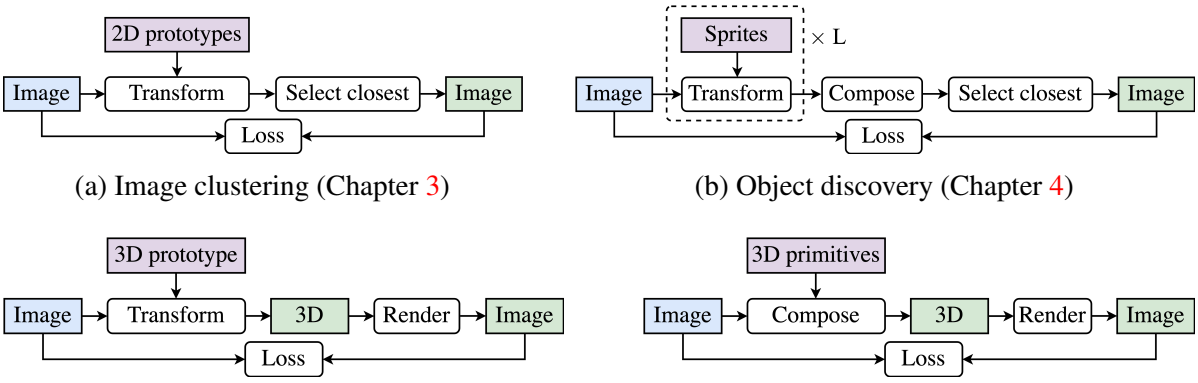
To tackle these challenges, we present the following three contributions.

Object class modeling with prototypes and deep transformations. We introduce a new conceptual approach to object class modeling. We propose to represent the class of an image, a 2D object or a 3D shape, with a prototype that is transformed using deep learning to model the different instances within the class. Specifically, we design differentiable parametric functions corresponding to meaningful transformations (*e.g.*, geometric deformations, color variations or morphological changes) and use neural networks to predict the transformation parameters that make the prototype look similar to what is observed in a given image. We simultaneously learn

the prototype and the deep predictors by gradient descent. We demonstrate the effectiveness of this idea to cluster images in Chapter 3 using prototypical images and 2D transformations as well as to reconstruct 3D objects from single-view images in Chapter 5 using prototypical 3D shapes and deep 3D deformations. In both cases, we obtain performances on par with the best state-of-the-art methods which leverage handcrafted features or annotations. In particular, this allows us to introduce a state-of-the-art image clustering method that performs clustering in pixel space.

Discovery by composition of learnable elements. We propose a new way to discover elements in images by representing an image collection with a set of learnable elements composed together to synthesize the images and updated by gradient descent. We first showcase in Chapter 4 how this idea can be used to discover 2D elements, in the form of learnable sprites, from a large collection of images depicting recurrent objects. Then, we further demonstrate this idea effectiveness in Chapter 6 by discovering 3D elements, in the form of learnable superquadric meshes, from a collection of images depicting a scene from multiple viewpoints. To solve the problem of modeling variable number of elements, we propose two different solutions in Chapter 4 and Chapter 6, respectively leveraging a greedy algorithm computing the best solution among all possibilities and soft transparency variables that allow us to visually remove elements by making them transparent. In both Chapter 4 and Chapter 6, we propose to solve the challenges posed by (i) the image generation ambiguity with regularization terms encouraging the use of a minimal number of elements, and (ii) occlusions with soft variables and smoothness priors.

Advances in mesh-based differentiable rendering. This contribution is more technical and consist in a new formulation to compute differentiable mesh rendering. Specifically, building upon the mesh-based differentiable rendering of Liu et al. [2019], we formulate the differentiable rendering of a 3D mesh as the alpha compositing of the mesh faces in a decreasing depth order. This is notably in line with the layered image model we propose in Chapter 4 as well as the formulation of NeRF [Mildenhall et al., 2020] which can be interpreted as alpha compositing points along the rays. This formulation is developed in Chapter 5 to learn unsupervised single-view reconstruction and used in Chapter 6 to find 3D primitives in a multi-view stereo setting. In addition, such a formulation allows us to seamlessly introduce the possibility to learn transparent meshes in Chapter 6, which we design to model a scene as a composition of a variable number of meshes.



(a) Image clustering (Chapter 3)

(b) Object discovery (Chapter 4)

(c) Single-view reconstruction (Chapter 5)

(d) Multi-view 3D decomposition (Chapter 6)

Figure 1.4: **Models and thesis outline.** We leverage our three contributions to build analysis-by-synthesis models to tackle the four computer vision problems we study in this thesis. Each model corresponds to a specific chapter in the manuscript and it is represented with a high-level diagram.

1.6 Thesis outline

Our three contributions are leveraged to build analysis-by-synthesis models to tackle the four computer vision problems we study in this thesis. Figure 1.4 illustrates our models with high-level analysis-by-synthesis diagrams. Each model is related to a specific chapter that we present next. This thesis is organized as follows.

Chapter 2: Background. In this chapter, we start by briefly introducing preliminary content related to general concepts in machine learning, deep learning and computer vision that are used throughout this manuscript. Then, we review the literature for each image analysis problem we study in this thesis, namely image clustering, object discovery, single-view reconstruction and multi-view 3D decomposition. For each problem, we strive to separate analysis methods based on features from analysis methods learned by image synthesis.

Chapter 3: Deep Transformation-Invariant Clustering. This chapter introduces the foundations of the first contribution of this thesis: a new object class modeling based on prototypes and deep transformations evaluated in the context of image clustering. We first describe how to represent an image collection by learning prototypical images and deep image transformations. We then present the family of transformations we consider and how we parametrize them. Finally, we empirically compare our approach to state-of-the-art methods on standard clustering benchmarks that are relatively simple (*e.g.*, MNIST [LeCun et al., 1998], Fashion-MNIST [Xiao et al., 2017], SVHN [Netzer et al., 2011]), and showcase its applicability to real photograph collections downloaded from the web like MegaDepth [Li and Snavely, 2018]. In contrast to

concurrent approaches, our method performs clustering in pixel space and exhibits a strong interpretability in the results.

Chapter 4: Discovering Objects With Sprite Modeling. We present a method to discover 2D recurrent objects in image collections. We model images as an alpha compositing of several layers made of transformed prototypical images with transparency called *sprites*. More specifically, we jointly learn in an unsupervised manner a small dictionary of sprites and parametric functions predicting their transformations to explain the images. We experimentally demonstrate that our method is on par with the state of the art on common multi-object discovery datasets (*e.g.*, Tetrominoes [Greff et al., 2019] and CLEVR [Johnson et al., 2017]). We also qualitatively show that our model can be applied to challenging sets of internet images like Instagram hashtag collections to discriminate foreground from background.

Chapter 5: Single-View Reconstruction Without Supervision. This chapter describes a method that learns to perform single-view 3D reconstruction from a raw collection of single-view images. Compared to prior works, this SVR approach *does not* rely on 3D shapes, multiple views of the same instance, camera viewpoints, keypoints or object masks during learning; it instead learns directly from raw images, without relying on any form of supervision. We first introduce the complete image rendering process: starting from a prototypical ellipsoidal 3D mesh, we deform and texture it using neural networks and render on top of a background image in a differentiable manner. We then present how to learn such a system and avoid bad local minima. Finally, we experimentally compare our approach to more supervised state-of-the-art methods on the standard ShapeNet benchmark [Chang et al., 2015], and demonstrate its advantages by applying it to real image collections like CompCars [Yang et al., 2015].

Chapter 6: Differentiable Blocks World. We develop the idea of discovering elements in multi-view images by modeling them as the projection of simple 3D geometric primitives. More specifically, given a set of calibrated multi-view images, we optimize a small set of textured superquadric meshes such that they are photo-consistent across the different views. We first describe how to parametrize a scene with primitive meshes, then present how to optimize our model parameters through differentiable rendering. We quantitatively and qualitatively evaluate our approach on DTU [Jensen et al., 2014] and demonstrate its superiority over state-of-the-art methods. Lastly, we showcase its advantages on real-life captures (*e.g.*, BlendedMVS [Yao et al., 2020]), including straightforward physics-based simulations and amodal scene completion.

Chapter 7: Conclusion. This chapter summarizes the contributions presented in this thesis, and proposes potential avenues for future research.

1.7 Publication list

Four papers are presented in the manuscript [Monnier et al., 2020, 2021, 2022, 2023]:

- Monnier, T., Groueix, T., and Aubry, M. [2020]. Deep Transformation-Invariant Clustering. In ***NeurIPS*** **(oral presentation)**.
- Monnier, T., Vincent, E., Ponce, J., and Aubry, M. [2021]. Unsupervised Layered Image Decomposition into Object Prototypes. In ***ICCV***.
- Monnier, T., Fisher, M., Efros, A. A., and Aubry, M. [2022]. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency. In ***ECCV***.
- Monnier, T., Austin, J., Kanazawa, A., Efros, A. A., and Aubry, M. [2023]. Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives. In ***NeurIPS***.

We open-sourced the code corresponding to all four papers[1], created webpages[2] for each project with additional visualizations of the results of the thesis. The webpages received overall around 5k unique visitors and the various project on GitHub received a total of 500 stars.

During my PhD, I was fortunate to participate in the following other publications [Monnier and Aubry, 2020; Loiseau et al., 2021; Wysoczańska et al., 2022; Siglidis et al., 2023; Guédon et al., 2023], which are not discussed in this manuscript:

With Mathieu Aubry

- Monnier, T., and Aubry, M. [2020]. docExtractor: An off-the-shelf historical document element extraction. In ***ICFHR***.
- Loiseau, R., Monnier, T., Aubry, M., and Landrieu, L. [2021]. Representing Shape Collections with Alignment-Aware Linear Models. In ***3DV***.
- Siglidis, I., Gonthier, N., Gaubil, J., Monnier, T., and Aubry, M. [2023]. The Learnable Typewriter: A Generative Approach to Text Line Analysis. ***arXiv:2302.01660 [cs.CV]***.

Without Mathieu Aubry

- Wysoczańska, M., Monnier, T., Trzciński, T., and Picard, D. [2022]. Towards Unsupervised Visual Reasoning: Do Off-The-Shelf Features Know How to Reason? In ***NeurIPS Workshops***.
- Guédon, A., Monnier, T., Monasse, P., and Lepetit, V. [2023] MACARONS: Mapping And Coverage Anticipation with RGB Online Self-Supervision. In ***CVPR***.

[1] https://github.com/monniert
[2] https://www.tmonnier.com

Chapter 2

Background

Approaches to unsupervised image analysis problems related to this thesis can be broadly split into two groups. The first group of methods traditionally leverages image representations, or *features*, to run unsupervised analysis models. Such representations typically correspond to general-purpose features or more recently, task-specific features that are learned. The second group of methods does not rely on intermediate image representations; it instead proposes to directly learn analysis models by synthesizing images. We respectively refer to these groups as *feature-based methods* and *synthesis-based methods*. In this chapter, we first briefly present preliminary content such as the machine learning context, foundational concepts in deep learning that are used throughout the manuscript and standard approaches for computing general-purpose features. Then, we review the literature related to each image analysis problem we tackle in this thesis, namely image clustering, object discovery, single-view reconstruction and multi-view 3D decomposition. For each problem, we strive to separate analysis methods based on features from analysis methods learned by image synthesis.

2.1 Preliminary

All the approaches presented in this thesis correspond to unsupervised machine learning methods, which are learned by gradient descent using the convenient automatic differentiation (or *auto-differentiation*) of recent deep learning softwares. In this section, we first introduce some machine learning context, describe important deep learning advances for computer vision and present a general formulation of feature-based and synthesis-based approaches. Then, we describe key approaches to compute general-purpose image features. Finally, we provide a brief history of advances to compute 3D meshes through differentiable rendering, which we build upon to propose our new formulation used in Chapters 5 and 6. We refer the reader

to Hastie et al. [2001]; Goodfellow et al. [2016]; Szeliski [2010] for an exhaustive review of machine learning, deep learning and standard image features respectively.

2.1.1 Machine learning

A machine learning problem is typically formulated as follows: given a *dataset* (*e.g.*, an image collection), the goal is to predict an *outcome measurement*, also called *label* (*e.g.*, a cat or dog image classification), based on the samples in the dataset. These predictions are performed by a model that is directly learned from the data. Machine learning models are typically split in two categories, depending on their learning settings.

Supervised learning. In this setting, we assume the access to a *training set* of data, in which the outcome measurement for a set of samples is known. This annotated data is used to learn a prediction model, which will be able to predict the outcome for new unseen samples that are typically referred to as a *test set*. A good model is one that accurately predicts such an outcome. In this case, models are called *supervised* because they are explicitly taught the desired outcome via labeled examples. Classical examples of supervised prediction models are linear least square models, nearest-neighbors methods associating the labels from neighboring training samples, or support vector machines finding separating hyperplanes for classification.

Unsupervised learning. In this setting, the outcome measurement is not known for any sample in the dataset. The task is rather to discover patterns in the data or try to describe how the data are organized or clustered. Compared to supervised problems where the objective is known, the objective in unsupervised problems is typically more difficult to formulate. Classical examples of unsupervised prediction models are K-means and gaussian mixture models for clustering, or principal components analysis and non-negative matrix factorization for dimensionality reduction.

2.1.2 Deep learning

Deep learning is branch of machine learning where the prediction model is a *neural network*. A neural network architecture corresponds to a family of parametric functions, which can be seen as a sequence of differentiable functions (called *layers*). The parameters of the neural network (also called *weights*) are optimized to minimize an energy function (called a *loss function*). Because the optimization problem is in general non-convex, a global optimization is impossible and it is typically solved with local optimization methods. The most common

optimization framework for learning neural networks is to compute gradients using backpropagation [Ivakhnenko and Lapa, 1966; Rumelhart et al., 1985; Lecun, 1988] and update the parameters by gradient descent methods. Today, recent deep learning softwares like PyTorch or TensorFlow integrate an automatic differentiation tool computing gradients automatically for any neural networks built of traditional layers, making it relatively easy to learn deep neural networks, or even any differentiable model written using the built-in functions. We next briefly describe few key architectures of neural networks related to computer vision, as well as important advances that dramatically improved their learning.

Classical architectures. Today, there exists a large collection of neural network models that are applied to tackle computer vision tasks. We here describe some important architectures that are used throughout this manuscript.

Multi-layer perceptron. A multi-layer perceptron (or MLP) is one of the earliest and most basic form of neural networks. It corresponds to a feed-forward network made of a sequence of fully-connected linear layers (also known as *perceptrons*), each followed by a non-linear differentiable functions like a rectified linear unit activation (ReLU) function [Fukushima, 1969]. MLPs build upon the perceptron model which is first introduced in the early work of Rosenblatt [1958], and MLPs are still widely used as a key component of modern deep learning approaches.

VGG. Introduced by Simonyan and Zisserman [2015], the family of VGG networks follows the idea of AlexNet [Krizhevsky et al., 2012] to use a deep and wide convolutional neural network (CNN) architecture, but proposes a simpler architecture with wider receptive fields. One of the crucial downsides of VGG networks is that they correspond to huge networks which take a lot of time to train. This is in part related to the problem of vanishing gradients, where gradients become very small for deep layers due to the successive application of the chain rule.

ResNet. This family of networks is presented in the work of He et al. [2016] to alleviate the problem of vanishing gradients in deep architectures. Specifically, residual connections are introduced, a concept allowing the signal to flow from one layer to another using an identity function. In particular, it enables to effectively train much deeper architectures which, in turn, significantly outperform existing architectures by the time of the publication on the ImageNet classification competition. Today, ResNet is still widely used as one of the go-to backbone architectures to tackle computer vision tasks.

Learning advances. A lot of advances have been proposed to effectively train deep neural networks. We here describe some important deep learning advances that are used throughout this manuscript.

Optimizers. Effectively optimizing the weights of a neural network given a large dataset and an objective function is a difficult task. The classical solution is to perform gradient descent iterations on small random chunks of data called *batches*, which is much less expensive than performing gradient descent iterations on the full dataset. In each gradient descent iteration, the loss function is computed on a given batch, gradients are computed by backpropagation and the weights are updated in the opposite gradient direction multiplied by an hyper-parameter called *learning rate*. This process is known as batch stochastic gradient descent (SGD). A lot of efforts have been dedicated to build optimizers that accelerate the training of deep neural networks and improve their performances. This notably includes introducing a momentum term to smooth the parameter updates (SGD with momentum [Nesterov, 1983]) and adapting the learning rate for each parameter using either an accumulation of all past gradients (AdaGrad [Duchi et al., 2011]) or an exponentially moving average of the square gradients (RMSProp [Tieleman and Hinton, 2012]). Adam [Kingma and Ba, 2015] can be seen as combining both concepts of momentum and RMSProp by smoothing the parameter updates using a moving average of the gradient's first and second moments. This optimizer is still widely used today, and it is the one we use to optimize all the models proposed in this manuscript.

Normalization layers. Normalizing the inputs of a machine learning model is a classical technique to attenuate biases in the input and often improves the model performance. Because a neural network is a sequence of layers that can be seen as micro models, the idea to normalize the input of several of its layers has been studied to improve the neural network performance. More specifically, the work of Ioffe and Szegedy [2015] introduces batch normalization, a layer that standardizes the input of the following layer across a single batch. This technique has proven to dramatically improve the learning of deep neural networks, by accelerating the training and allowing better convergence. Other works like LayerNorm [Ba et al., 2016], InstanceNorm [Ulyanov et al., 2016], BatchRenorm [Ioffe, 2017] or GroupNorm [Wu and He, 2019] further introduce normalization layers that are designed to improve deep neural networks training.

Miscellaneous. A large amount of other techniques have been developed to improve the convergence of neural networks. In particular, this includes techniques like data augmentation, dropout [Srivastava et al., 2014], sophisticated scheduling of the learning rate [Smith, 2017] and weight decay [Loshchilov and Hutter, 2019].

2.1.3 Feature-based vs synthesis-based

We describe a general formulation for feature-based and synthesis-based approaches to unsupervised image analysis, and emphasize their differences.

Feature-based approach. An unsupervised analysis approach based on features typically consist in two stages. In the first stage, a method is used to perform the extraction of meaningful features (also known as *descriptors* or *embeddings*) representing the images. Each image can be represented by a set of features corresponding to different locations of interest in the image (or *keypoints*), or a single image-level feature. In the second stage, an unsupervised model takes as input the set of features and outputs the desired analysis outcome (*e.g.*, grouping the images by similarity). Such a two-stage process requires to carefully select and tune the representation depending on the analysis task. In addition, although recent approaches typically leverage deep models that are learned separately in both stages, they do not fully benefit from deep end-to-end learning advances. Note that feature-based methods typically corresponds to a *bottom-up* approach: it starts by creating low-level features from the pixels, then it manipulates the extracted features to build the desired outcome.

Synthesis-based approach. Analyzing an image collection by synthesis is instead *top-down*: it aims at learning an analysis model by synthesizing the images in the collection, without relying on a feature-based multi-stage procedure. Such approaches have been empowered by recent deep gradient-based softwares which provide efficient tools like automatic differentiation and advanced optimizers to seamlessly design and learn a desired synthesis model. Specifically, the general structure for recent synthesis-based approaches is as follows: a differentiable parametric model generating images is implemented and its parameters are learned through gradient descent by minimizing a *rendering loss* (or *reconstruction loss*) between the sample images and the predicted renderings.

2.1.4 General-purpose image features

Computing a feature that represents well an image or a local patch is at the very root of many image analysis problems, like stereo matching, motion estimation or object tracking to name a few. Designing good image representations is challenging because there is no clear answer to what is the ideal image representation: should it contain information about the objects in the image (*e.g.*, their type, scale, texture), or should it model the depth of different image regions, or something else? We first review a few examples of handcrafted features. Then, we describe

image features that are learned, with a focus on features output by deep neural networks, which we refer to as *deep features*.

Handcrafted features. Traditional image features are manually computed using heuristics which are for example based on normalized pixel intensities [Brown et al., 2005], Gaussian filtering [Freeman and Adelson, 1991] or image gradients [Freeman and Roth, 1995; Oliva and Torralba, 2001; Lowe, 2004]. We describe a few examples of key handcrafted features that are based on gradient:

HOG. The histogram of oriented gradients (HOG) computes a feature based on occurrences of gradient orientation in localized portions of an image. It is first described in the patent application of McConnell [1982] without specifically using the HOG term, and its usage becomes widespread by Freeman and Roth [1995] and more significantly by Dalal and Triggs [2005], who successfully demonstrate its effectiveness for pedestrian detection. To represent an entire image, HOG features are typically computed in overlapping image regions using a sliding window and concatenated, thus resulting in a dense representation.

SIFT. Scale invariant feature transform (SIFT) is introduced by Lowe [2004] and includes both a keypoint detector and a feature descriptor. Compared to the dense HOG representation, an entire image is typically represented by a set of SIFT features computed at the detected keypoints. SIFT features are formed by computing the gradient at each pixel in a 16×16 neighborhood of a desired image location. Gradient magnitudes are downweighted by a Gaussian function to reduce the gradient influence far from the center. 8-bin histograms of gradient orientations are built in each 4×4 quadrant, resulting in a feature vector of 128 non-negative values. SURF features [Bay et al., 2006] are a variant of SIFT where the feature extraction is significantly faster.

GIST. Introduced by Oliva and Torralba [2001], GIST corresponds to a compact representation of an image based on the spatial envelope of its gradient information. Compared to a SIFT representation based on local features computed at different keypoints, GIST aims at computing a representation of the image that is global.

Deep features. A recent trend instead proposes to compute image features using deep learning. The main benefit of deep features is that neural networks can automatically learn a good representation to solve a desired task, while obtaining this level of representation by manually designing hundreds of heuristics with millions of parameters looks unfeasible. Methods computing deep features can be split into three groups, depending on how the neural networks are learned.

Learning with a supervised task. One of the most common solution to obtain deep features is to train a neural network through a supervised task (*e.g.*, the pioneering AlexNet [Krizhevsky et al., 2012] for image classification on ImageNet benchmark [Deng et al., 2009]) and use this pre-trained network as a feature extractor for any downstream application. This strategy, also known as *transfer learning*, has quickly proven to be surprisingly very effective [Donahue et al., 2014; Zeiler and Fergus, 2014; Oquab et al., 2014; Razavian et al., 2014] and relies on the intuition that solving the supervised task requires to learn image representations that are sufficiently good to be transferred to other data distributions and other tasks. Note that in this case, training the network requires annotated examples.

Learning by image generation. Prior to the widespread solution of transferring supervised knowledge, several works focus on computing deep representations without any annotations, by learning a deep model that best synthesizes the images in the dataset. A classical example in this direction is the autoencoder model [Hinton and Salakhutdinov, 2006; Ranzato et al., 2007; Vincent et al., 2008], which is composed of an encoder mapping the image to an intermediate representation (also known as the *bottleneck*) and a decoder mapping this representation back to the image space. Other generative approaches like variational autoencoders (VAE) [Kingma and Welling, 2014] or generative adversarial networks (GAN) [Goodfellow et al., 2014] are studied for learning good image representations without supervision [Radford et al., 2016; Dumoulin et al., 2017; Donahue et al., 2017]. However, the resulting performances are typically moderate.

Learning with a self-supervised task. The importance of obtaining good features combined with the effectiveness of deep features sparked a real interest to shift the focus from designing task-specific methods to learning general-purpose representations, or *representation learning*, which became a new field in itself. Recent unsupervised approaches for this problem typically leverage *self-supervision*, by manipulating the data to design a supervised pretext task that will be used to train the network. A representative example is the work of Gidaris et al. [2018] which randomly rotates the input image by a multiple of 90° and learns to predict the image rotation. The task is thus framed as a supervised classification problem into four categories, without any manually annotated examples. Common self-supervised techniques include relative patch position [Doersch et al., 2015], image colorization [Zhang et al., 2016] or image inpainting [Pathak et al., 2016]. Today, the most successful approaches [Chen et al., 2020; Grill et al., 2020; Gidaris et al., 2021; Caron et al., 2021] are derived from the central idea of Dosovitskiy et al. [2014] which proposes to learn representations that are invariant to common transformations of the

data. This idea leverages the intuition that a good representation should not vary when the input image is altered in a way that the semantic content is preserved.

We refer the reader to Caron [2021] for an exhaustive review of representation learning literature. Overall, these methods are designed to output general-purpose features, yet they do not solve an analysis task *per se*. One would have to apply an unsupervised algorithm on top of the extracted features in order to compute the desired image analysis outcome (*e.g.*, grouping the images by similarity).

2.1.5 Mesh-based differentiable rendering

Recent 3D reconstruction approaches representing the scene with meshes leverage advances in mesh-based differentiable rendering. One of the first attempts to render meshes in a differentiable manner is the work of Loper and Black [2014]. In this work, they introduce a differentiable renderer built upon the auto-differentiation framework Chumpy, by approximating derivatives with local filters that are applied differently whether a pixel contains occlusion boundaries. Later, Kato et al. [2018] propose an alternative formulation based on manual derivative approximations that are more suitable to learning neural networks. Another set of methods proposes to instead approximate the rendering function with a differentiable function that allows backpropagation. This idea is first presented by Liu et al. [2019] introducing the classical SoftRasterizer, which is later computationally optimized by Ravi et al. [2020]. Other works like [Chen et al., 2019b; Laine et al., 2020] build upon the SoftRasterizer idea and proposes alternatives. We refer the reader to [Kato et al., 2020] for a comprehensive study.

In Chapter 5 and Chapter 6, we also build upon the idea of SoftRasterizer, but modify the rendering formulation to enable the learning of 3D meshes from raw photometric comparisons, without any silhouette information.

2.2 Image clustering

The goal of image clustering is to group images into object classes (or categories) associated to different concepts without having access to any object class annotations to train a recognition system. This is related to the widespread supervised task of image classification (or category recognition as it used to be called), which assumes the access to a set of annotated examples to train the system.

2.2.1 Feature-based methods

We first review classical methods that leverage handcrafted features, then we discuss more recent approaches based on deep features.

Classical methods. Prior to the groundbreaking AlexNet [Krizhevsky et al., 2012], category recognition is still considered a largely unsolved problem even with training images, and thus most classical methods tackle category recognition by assuming the access to a training set. Such methods typically extract handcrafted features and learn a classification algorithm in a supervised setting. Although these methods can theoretically be learned without annotations by replacing the supervised classification algorithm with an unsupervised one, only few works study this direction. We describe two classical approaches to category recognition, namely bag of words and part-based models.

Bag of words. The *bag of words* approach (also known as *bag of features*, *bag of keypoints*) represents an image as an unordered collection of features and is considered as one of the simplest methods for category recognition. Specifically, it consists in computing the distribution of quantized features (also called *visual words*) at detected location of keypoints in the query image, and comparing this distribution to those found in the training images. This idea is first explored by Sivic and Zisserman [2003] for matching objects in videos and the term *bag of keypoints* is introduced by Csurka et al. [2004], which compute quantized SIFT features using K-means algorithm and employs both a naive Bayes classifier and support vector machines for classification. Further approaches in this direction [Grauman and Darrell, 2005; Lazebnik et al., 2006; Zhang et al., 2007] typically investigate other means to detect keypoints, compute features, or perform classification. One of the first attempt to apply this idea to unlabeled images is the work of Grauman and Darrell [2006] which uses a spectral clustering technique to recover the image grouping in SIFT feature space. Another noticeable unsupervised idea to this problem is the methods of Sivic et al. [2005, 2008] employing text analysis models to discover *topics* from pre-computed visual words and assimilating the discovered topics to different object categories.

Part-based models. Recognizing an object by finding its constituent parts and measuring their geometric relationships can be traced back to the early work of Fischler and Elschlager [1973] introducing *pictorial structures*, a model where geometric relationships between parts are represented by spring-like connections. Building upon this idea, further works investigate other topologies for the geometric relationships between parts such as a tree structure [Felzenszwalb and Huttenlocher, 2005; Fergus et al., 2005], a directed

acyclic graph [Bouchard and Triggs, 2005; Carneiro and Lowe, 2006] or a constellation model [Burl et al., 1998; Fergus et al., 2007]. In particular, the constellation model is at the root of key approaches designed for unsupervised category recognition like Weber et al. [2000a,b]; Fergus et al. [2003]. Despite their elegant interpretability, part-based models are difficult to fit to real-world images, which typically contain a preponderance of clutter.

Deep methods. Most recent feature-based approaches to image clustering employs deep learning. This kind of methods typically aims at learning without supervision deep features that will specifically provide good clustering performances after the application of a standard unsupervised clustering algorithm (*e.g.*, K-means [MacQueen, 1967]). These methods can be grouped depending on the framework employed to learn the clustering features.

Autoencoder. One of the most natural solution to learn deep features suited for clustering is to adapt the classical deep autoencoder model [Hinton and Salakhutdinov, 2006; Ranzato et al., 2007; Vincent et al., 2008] such that the space of the bottleneck representation is (as much as possible) discriminative and well-separated. Works in this direction include Xie et al. [2016]; Dizaji et al. [2017]; Jiang et al. [2017]. A noticeable work related to this thesis is Kosiorek et al. [2019] which leverages the idea of capsules [Hinton et al., 2011] to learn equivariant image features, in a similar fashion of equivariant models [Lenc and Vedaldi, 2015; Tai et al., 2019].

GAN. Several works investigate the idea to adapt the unsupervised learning framework of GAN to learn meaningful representations that are suited for clustering. One of the first attempt is InfoGAN [Chen et al., 2016] which augments the traditional input noise vector with latent variables drawn from structured distributions (*e.g.*, a uniform categorical distribution) and learns disentangled representations by maximizing the mutual information between these latent variables and the observations. Other works in this direction include [Zhou et al., 2018; Mukherjee et al., 2019].

Contrastive learning. Another line of works proposes to directly inject the desired objective of learning discriminative features into the loss function to minimize during learning. This is generally done using the paradigm of *contrastive learning*, which aims at encouraging the model to bring features of positive pairs (similar images) closer together, and to push features of negative pairs (dissimilar images) farther apart. Works in this direction typically use the triplet loss [Yang et al., 2016] originally introduced by Schroff et al. [2015a], pseudo classification labels [Chang et al., 2017], a loss based on siamese networks [Shaham et al., 2018] firstly introduced by [Chopra et al., 2005], or strategies

maximizing representation similarities between a sample and random transformations of this sample [Hu et al., 2017; Häusser et al., 2018; Ji et al., 2019], in a flavor similar to the seminal idea of Dosovitskiy et al. [2014].

2.2.2 Synthesis-based methods

Clustering images by synthesis requires to model object categories in the image synthesis process. One of the most naive approaches is the classical K-means algorithm, originally introduced as an iterative process by MacQueen [1967] and later adapted to a gradient-based optimization by Bottou and Bengio [1995]. In K-means, representatives (or *prototypes*) for each category are computed such that they best synthesize the data, and each sample is assigned to the group of the closest prototype. The notion of closeness involves a distance function (traditionally, the Euclidean distance) defined in the input space (for images, this corresponds to the pixel space). Despite its known popularity, K-means dramatically fails when directly applied to pixels of images. One major reason for such a failure is that computing distances in pixel space is not robust to common image transformations. For example, the Euclidean distance between an image and an augmented version translated by a few pixels would likely not be zero for non-trivial images, which is problematic.

To tackle this issue, a group of approaches aims at aligning images in pixel space using a relevant family of transformations (such as translations, rotations, or affine transformations) to make pixel distances at least robust to such transformations. Frey and Jojic [1999, 2002, 2003] are one the first to investigate this direction by formulating their seminal work about *transformation-invariant clustering*. More formally, they propose to model image translations with pixel permutations represented by a discrete latent variable that is incorporated within an Expectation Maximization (EM) [Dempster et al., 1977] procedure for a mixture of Gaussians. Their approach is however limited to a finite set of discrete transformations. *Congealing* generalized the idea to continuous parametric transformations, and in particular affine transformations, initially by using entropy minimization [Miller et al., 2000; Learned-Miller, 2005]. A later version using least square costs [Cox et al., 2008, 2009] demonstrated the relation of this approach to the classical Lukas-Kanade image alignment algorithm [Lucas et al., 1981]. In its standard version, congealing tackles the joint alignment of images belonging to the same object category, but the idea was extended to clustering [Liu et al., 2009; Mattar et al., 2012; Li et al., 2016], for example using a Bayesian model [Mattar et al., 2012], or in a spectral clustering framework [Li et al., 2016]. These works typically formulate difficult joint optimization problems and solve them by alternating between clustering and transformation optimization for each sample. They are thus limited to relatively small datasets and, to the best of our knowledge, were never compared to modern deep approaches on large benchmarks.

Deep learning is recently used to scale the idea of congealing for global alignment of a single class of images [Annunziata et al., 2019] or time series [Weber et al., 2019]. Both works build on the idea developed by Jaderberg et al. [2015] introducing spatial transformer networks (STN), where a spatial transformation is modeled as the differentiable sampling of a source input using a deformed sampling grid.

In Chapter 3, we take direct inspiration from this line of works and introduce a framework called *deep transformation-invariant clustering*. Specifically, we propose a simple modification to prototype-based algorithms like K-means which amounts to simultaneously learning prototypes and image-to-prototype alignments using deep learning. We also build upon spatial transformer networks [Jaderberg et al., 2015], but go beyond single-class alignment to jointly perform clustering. Additionally, we extend the idea from traditional geometric transformations to more generic transformations, like color and morphological variations. We believe our work is the first to use deep learning to effectively perform clustering in pixel space by explicitly aligning images. In particular, it allows us to turn a failing K-means algorithm into a state-of-the art clustering method.

2.3 Object discovery

Object discovery corresponds to the task of locating and identifying objects in a collection of images without any supervision, and thus relates to the supervised problems of object detection, semantic segmentation and instance segmentation. When the goal is to precisely recover the spatial extent of the object as a pixel-accurate image region (also called *segmentation*, *mask* or *silhouette*), the problem can be referred to as *cosegmentation* and it typically assumes a simpler setting where images correspond to a single object of the same category.

2.3.1 Feature-based methods

We first describe classical methods leveraging handcrafted features, then we review more recent methods based on deep features.

Classical methods. An early approach that separate and identify meaningful regions in images is the work of Brice and Fennema [1970] which breaks the image into atomic regions of uniform gray scale then use heuristics to group them. More recent works in this direction take advantages of handcrafted features, instead of low-level image cues, to compute meaningful regions. Below, we present two ideas particularly developed in the literature.

Topic discovery. A first group of methods takes inspiration from text analysis approaches designed to discover topics in a corpus using the bag-of-words document representation. Specifically, this group of methods computes bag-of-words image representations, apply topic discovery approaches to the resulting visual words and treat the discovered topics as object categories. One of the first attempt that seeded this line of works is Sivic et al. [2005] which successfully discovers objects in images, by identifying their category and their approximate location. The follow-up work of Russell et al. [2006] extends the approach to more precisely identify the object location, by predicting the pixel-accurate object segmentation. Further works in this direction include Cao and Fei-Fei [2007]; Sivic et al. [2008].

Feature matching. A second group of methods aims at matching features across images to identify recurrent objects in the image collection. One of the first attempts is the work of Grauman and Darrell [2006] which represents images as bags of SIFT features, computes pairwise image similarities by partially matching features, and uses spectral clustering to compute the image groupings. Within an image group, the approximate object location is recovered by looking at the matched features. Some works [Rother et al., 2006; Joulin et al., 2010; Vicente et al., 2011; Rubio et al., 2012; Joulin et al., 2012; Rubinstein et al., 2013] specifically focus on refining the object location by predicting its spatial extent, typically assuming the traditional cosegmentation setting of a single object from the same category. Other works [Faktor and Irani, 2012; Cho et al., 2015; Vo et al., 2019] instead focus on scaling object discovery to a more general scenario of real-world image collections representing several object classes. In particular, Cho et al. [2015] represent part-based region proposals with HOG features and match them across images using a probabilistic Hough transform.

Deep methods. Most recent approaches use deep features to tackle object discovery, rather than using generic handcrafted representations. A first line of works aims at incorporating CNN-based features to improve cosegmentation performances for single object category. An early example is the work of Quan et al. [2016] which uses both SIFT descriptors and deep features extracted from an off-the-shelf CNN trained for ImageNet classification. Other examples [Li et al., 2018, 2019a] try to incorporate the unsupervised learning of the CNN extractor to obtain deep features more suited for cosegmentation. Another line of works [Shen et al., 2019; Vo et al., 2020; Siméoni et al., 2021] instead focuses on leveraging deep features to scale object discovery to more complex real-world setups, where multiple objects from different classes are represented within the same image.

2.3.2 Synthesis-based methods

Discovering objects by synthesis requires to design an image model that explicitly integrates multiple elements during the image formation. A classical image model is to build an image by alpha compositing several images with transparency (or *layers*), each representing different region of the final image. We first review works related to layered image modeling, which are typically limited to simpler settings like targeting foreground-background decomposition or assuming the access to image sequences. We then discuss recent deep learning based works focusing on decomposing images into an unknown number of objects, an unsupervised task often referred to as *object-based image decomposition* or *scene decomposition*.

Layered image modeling. The idea of building images by compositing successive layers can already be found in the early work of Matheron [1968] introducing the *dead leaves model*. In this work, an image is assembled as a set of templates partially occluding one another and laid down in layers. Originally meant for material statistics analysis, this work is extended by Lee et al. [2001] to a scale-invariant representation of natural images. Jojic and Frey [2001] propose to decompose video sequences into layers undergoing spatial modifications (called *flexible sprites*) and demonstrate applications to video editing. Leveraging this idea, Winn and Jojic [2005] introduce LOCUS, a method designed for learning object models from unlabeled images and evaluated for foreground segmentation. Recently, approaches like Yang et al. [2017]; Lin et al. [2018]; Singh et al. [2019]; Chen et al. [2019a]; Arandjelović and Zisserman [2019] use GANs to learn layered image compositions, yet they are limited to a foreground-background decomposition. A noticeable work related to ours is Reddy et al. [2020], which proposes to optimize a single image decomposition into a set of pattern elements that are assumed to be known.

Scene decomposition. A recent trend of works proposes to leverage deep learning to learn image decomposition into objects in an unsupervised setting. A first line of works tackles the problem from a spatial mixture model perspective where latent variables encoding pixel assignment to groups are estimated. The seminal work from Greff et al. [2016a,b, 2017] introduces spatial mixtures, models complex pixel dependencies with neural networks and uses an iterative refinement to estimate the mixture parameters. MONet [Burgess et al., 2019] jointly learns a recurrent segmentation network and a VAE to predict component mask and appearance. IODINE [Greff et al., 2019] instead uses iterative variational inference to refine object representations jointly decoded with a spatial broadcast decoder [Watters et al., 2019] as mixture assignments and components. Combining ideas from MONet and IODINE, GENESIS [Engelcke et al., 2020] predicts, with an autoregressive prior, object

mask representations used to autoencode masked regions of the input. Leveraging an iterative attention mechanism [Vaswani et al., 2017], Slot Attention [Locatello et al., 2020] produces object-centric representations decoded in a fashion similar to IODINE. Other related methods are [van Steenkiste et al., 2018; von Kügelgen et al., 2020; Yang et al., 2020; Smith et al., 2023]. Another set of approaches builds upon the work of Eslami et al. [2016] who propose a VAE-based model using spatial attention [Jaderberg et al., 2015] to iteratively specify regions to reconstruct. This notably includes SQAIR [Kosiorek et al., 2018], SPAIR [Crawford and Pineau, 2019] and the more recent SPACE [Lin et al., 2020b]. To the best of our knowledge, none of these methods explicitly model object categories, nor demonstrate applicability to real images.

In Chapter 4, we present a scene decomposition approach which is orthogonal to this line of works generating layers implicitly with neural networks. It instead takes inspiration from traditional layered image models like Matheron [1968]; Jojic and Frey [2001] and represents layers as the explicit transformation of learnable prototypical images with transparency called *sprites*. Specifically, we jointly learn a small dictionary of sprites and neural networks predicting their transformations to explain the images, in a fashion similar to the idea developed in Chapter 3. In contrast to prior works, our sprite modeling allows us to not only identify different object types, each represented by a different sprite, but also to demonstrate results on Internet images. Note that this idea is also explored by MarioNette [Smirnov et al., 2021], a concurrent approach modeling sprites with latent features, and further works like Reddy et al. [2022] build upon such sprite-based image modeling.

2.4 Single-view reconstruction

Single-view reconstruction (SVR) is a long standing computer vision problem that aims at computing the 3D model of a scene depicted by a single image. We first review feature-based approaches, then we discuss analysis-by-synthesis methods, which typically leverage 3D reconstruction models based on deep learning.

2.4.1 Feature-based methods

Classical approaches to SVR are typically bottom-up; they compute low-level image features that are used to predict a 3D model explaining such features. An early example is the very first computer vision thesis of Roberts [1963] proposing *Blocks World*. In this work, polyhedral 3D blocks are computed by detecting and matching the edges of an image depicting a simple scene

made of textureless polyhedral shapes. Criminisi et al. [2000] propose to detect vanishing points and use some additional user input to infer the 3D geometry. Hoiem et al. [2005] instead present a fully automatic method for creating a 3D model from a single outdoor photograph. The authors start by computing superpixels that are grouped into regions labeled as ground, vertical or sky regions. Then, region boundaries are used to infer 3D lines along which the image is folded into a "pop-up". The work of Delage et al. [2006] also assumes a 3D model made of a flat floor and vertical walls but proposes a system designed for indoor images. Other works goes beyond the assumption of a "floor-wall" 3D geometry. Saxena et al. [2009] represent 3D scenes with many small planar surfaces recovered from superpixels, while Gupta et al. [2010]; Lee et al. [2010] recover 3D reconstruction by computing 3D cuboids that best match surface orientation predictions output by the off-the-shelf system of Hoiem et al. [2005, 2007].

2.4.2 Synthesis-based methods

Recent SVR works typically leverage deep learning to learn the priors necessary to hallucinate the 3D parts hidden in the single-view image, as illustrated by the 3D supervised approaches of Choy et al. [2016]; Wang et al. [2018]; Groueix et al. [2018]; Mescheder et al. [2019]; Xu et al. [2019]. A first group of deep synthesis-based methods proposes to learn deep SVR models through multi-view supervision, a learning setting that does not correspond to most image collections. Another group of methods instead focuses on removing the need for supervision, to learn 3D reconstructions from raw image collections. We first review approaches related to these two groups, then we describe in more details works involving the differentiable rendering of meshes.

Multi-view supervision. Approaches based on multi-view supervision generally work as follows: given multiple views of the same object during training, one view is used to predict a 3D object and the other views are used to compare the 3D object renderings and update the model parameters. This type of approaches also assumes the access for each image to the object silhouette, *i.e.* an image-sized binary mask representing the object spatial extent. The first methods using multiple views and silhouettes initially require camera poses and are developed for diverse 3D shape representations: Yan et al. [2016]; Tulsiani et al. [2017b] opt for voxels, Kato et al. [2018]; Liu et al. [2019]; Chen et al. [2019b] learn meshes, and [Niemeyer et al., 2020; Sitzmann et al., 2019, 2021] leverage implicit neural representations. Works like Insafutdinov and Dosovitskiy [2018]; Tulsiani et al. [2018] then introduce techniques to remove the assumption of known camera poses. Except for Zhang et al. [2021b] who leverage GAN-generated images from the model of Karras et al. [2019], these works are typically limited to synthetic image collections.

Towards unsupervision. There is a clear trend to remove supervision from deep SVR pipeline to learn 3D from raw 2D images. This is very challenging and works in this direction typically focus on learning 3D from images of a single object category. Early works [Vicente et al., 2014; Kar et al., 2015; Tulsiani et al., 2017b] estimate camera poses with keypoints and minimize the silhouette reprojection error. The ability to predict textures is first incorporated by CMR [Kanazawa et al., 2018] which, in addition to keypoints and silhouettes, uses symmetry priors. Recent works [Lin et al., 2020a; Duggal and Pathak, 2022] replace the mesh representation of CMR with implicit functions that do not require symmetry priors, yet the predicted texture quality is strongly deteriorated. Henderson et al. [2020] improve upon CMR and develop a framework for images with camera annotations that does not rely on silhouettes. Two works managed to further avoid the need for camera estimates but at the cost of additional hypothesis: Henderson and Ferrari [2019] show results with textureless synthetic objects, Wu et al. [2020] model 2.5D objects like faces with limited background and viewpoint variation. Finally, recent works only require object silhouettes but they also make additional assumptions: Goel et al. [2020]; Tulsiani et al. [2020] use known template shapes, Li et al. [2020] assume access to an off-the-shelf system predicting part semantics, and Wu et al. [2021] target solids of revolution. Other related works [Gadelha et al., 2017; Kato and Harada, 2019; Henzler et al., 2019; Pavllo et al., 2020; Ye et al., 2021; Hu et al., 2021] leverage in addition generative adversarial techniques to improve the learning process.

In Chapter 5, we present an unsupervised SVR approach that *does not* use camera estimates, keypoints, silhouettes, nor strong dataset-specific assumptions, which correspond to a form of manual supervision restricting the application to specific image collections. In addition, we demonstrate results for both diverse shapes and real images. To the best of our knowledge, we present the first generic SVR system learned from raw image collections.

2.5 Multi-view 3D decomposition

Decomposing a 3D scene represented by multiple view images is a task we study in Chapter 6 and that has rarely been formalized. Specifically, given a set of images representing multiple views of a scene taken from known camera calibrations and poses, the goal is to decompose the scene into 3D geometric primitive shapes (like cuboids or cylinders) in such a that they reconstruct well the 3D scene geometry. This type of input is often shorten as *calibrated* or *posed multi-view images* and cameras are typically automatically computed from the multi-view images by a structure-from-motion algorithm. Such a multi-view 3D decomposition task can be seen as a primitive-based approach to the classical multi-view stereo problem, and it is

also related to prior approaches computing 3D decompositions from 3D point clouds or image representations like depth maps. We first review approaches to multi-view stereo and we then discuss works related to the general 3D decomposition problem.

2.5.1 Multi-view stereo

Multi-view stereo (MVS) refers to a 3D reconstruction problem where the input is a set of images representing multiple views of a scene, along with their camera calibration and pose. We first review approaches based on features, then discuss synthesis-based approaches.

Feature-based methods. We briefly present recent MVS works that perform 3D reconstruction using features. We refer the reader to Hartley and Zisserman [2003]; Furukawa and Hernández [2015] for an exhaustive review of classical MVS methods.

Feature-based MVS approaches typically rely on several processing steps to extract the final geometry from the images. Most recent methods [Zheng et al., 2014; Galliani et al., 2015; Yao et al., 2018, 2019; Zhang, 2020; Gu et al., 2020; Sinha et al., 2020], including the popular COLMAP developed by Schönberger et al. [2016], first estimate depth maps for each image through feature matching [Zheng et al., 2014; Schönberger et al., 2016] or neural network predictions [Yao et al., 2018, 2019; Zhang, 2020; Gu et al., 2020; Sinha et al., 2020]), then apply a depth fusion step to generate a textured point cloud. Finally, a mesh can be obtained with a meshing algorithm like Kazhdan and Hoppe [2013]; Labatut et al. [2007]. Other feature-based approaches directly rely on point clouds [Furukawa and Ponce, 2007; Labatut et al., 2007] or voxel grids [Ji et al., 2017; Murez et al., 2020]. Note that works like Ji et al. [2017]; Murez et al. [2020] leverage neural networks to directly regress the geometry, but they rely on a training phase requiring 3D supervision before being applied to unknown sets of multi-view images. Computing 3D reconstruction through multiple steps involves careful tuning of each stage, thus increasing the pipeline complexity. An important recent effort to MVS instead aims at directly learning 3D reconstruction by synthesizing the multi-view images, which we discuss next.

Synthesis-based methods. MVS can also be formulated as a global optimization problem, where a 3D scene representation is optimized by rendering the scene and maximizing the photoconsistency across the different views. An early attempt is the voxel coloring algorithm of Seitz and Dyer [1997] labeling voxels that are sufficiently photoconsistent as parts of the object. Kutulakos and Seitz [1999] generalize voxel coloring to space carving, where the 3D voxel grid is carved away along different camera axes by using silhouettes. Other early works like Faugeras and Keriven [1998]; Pons et al. [2007]; Kolev et al. [2009]; Cremers and Kolev [2011] investigate continuous scene representations to improve the reconstruction quality.

Reconstructing a 3D scene by synthesizing images has recently gain attention through advances in differentiable rendering, whose development has been eased thanks to deep gradient-based softwares. Specifically, the general approach to recent 3D reconstruction by synthesis is to design a 3D scene model along with a differentiable rendering process, and to learn the 3D model by rendering it from different views and maximizing the photometric consistency across the views through gradient descent. A first group of methods uses neural networks to implicitly represent the 3D scene, in the form of occupancy fields [Niemeyer et al., 2020], signed distance functions [Yariv et al., 2020] or radiance fields, as introduced in NeRF [Mildenhall et al., 2020]. Several works incorporate surface constraints in neural volumetric rendering to further improve the scene geometry [Oechsle et al., 2021; Yariv et al., 2021; Wang et al., 2021; Darmon et al., 2022; Fu et al., 2022], with a quality approaching that of traditional MVS methods. Another group of methods [Gao et al., 2020; Zhang et al., 2021a; Goel et al., 2022; Munkberg et al., 2022] instead propose to leverage recent advances in mesh-based differentiable rendering [Liu et al., 2019; Ravi et al., 2020] to explicitly optimize textured meshes. Compared to implicit 3D representations, meshes are highly interpretable and are straightforward to use in computer graphic pipelines [Munkberg et al., 2022]. However, all the above approaches represent the scene as a single mesh, making it ill-suited for manipulation and editing.

In Chapter 6, we introduce an approach to MVS that reconstruct a scene using textured primitive meshes. Such a primitive-based representation is interpretable and actionable; it allows us to perform physics-based simulations and scene editing in a seamless fashion. The concurrent work PartNeRF [Tertikas et al., 2023] proposes to introduce parts in NeRFs. However, only synthetic scenes with a single object are studied and the discovered parts mostly correspond to regions in the 3D space rather than interpretable geometric primitives.

2.5.2 3D decomposition

The goal of understanding a scene by decomposing it into a set of geometric primitives can be traced back to the very fist computer vision thesis by Roberts [1963] on Blocks World where polyhedral 3D blocks are used as primitives. In the 1970s, Binford [1971] proposes the use of Generalized Cylinders as general primitives, later refined by Biederman [1987] into the recognition-by-components theory. But applying these ideas to real-world image data has proved rather difficult, and most approaches in this direction typically propose to directly fit primitives to 3D representations or convenient image features. We next discuss such approaches.

3D fitting. A large family of methods does not consider images at all, instead focusing on finding primitives in 3D data. Building upon the classical idea of RANSAC [Fischler and Bolles, 1981], works like [Bolles and Fischler, 1981; Chaperon and Goulette, 2001; Schnabel et al., 2007, 2009; Li et al., 2011; Ramamonjisoa et al., 2022] accurately extract various primitive shapes (*e.g.*, planes, spheres and cylinders for Schnabel et al. [2007, 2009]; Li et al. [2011]) from a point cloud. In particular, MonteBoxFinder [Ramamonjisoa et al., 2022] is a recent RANSAC-based system that robustly extracts cuboids from noisy point clouds by selecting the best proposals through Monte Carlo Tree Search. To avoid the need for RANSAC hyperparameter tuning while retaining robustness, Liu et al. [2022] introduce a probabilistic framework dubbed EMS that recovers superquadrics [Barr, 1981]. Other methods instead leverage neural learning advances to robustly predict primitive decomposition from a collection of shapes (*e.g.*, ShapeNet [Chang et al., 2015]), in the form of cuboids [Tulsiani et al., 2017a], superquadrics [Paschalidou et al., 2019, 2020; Wu et al., 2022], shapes from a small dictionary [Li et al., 2019b; Le et al., 2021], learnable prototypical shapes [Deprelle et al., 2019; Loiseau et al., 2023] or parts generated by neural networks [Paschalidou et al., 2021; Prabhudesai et al., 2023]. However, they are typically limited to shapes of known categories and require perfect 3D data. More generally, the decomposition results of all 3D-based methods highly depend on the quality of the 3D input, which is always noisy and incomplete for real scenes. For a survey of classical 3D decomposition methods into geometric primitives, we refer the reader to Kaiser et al. [2019].

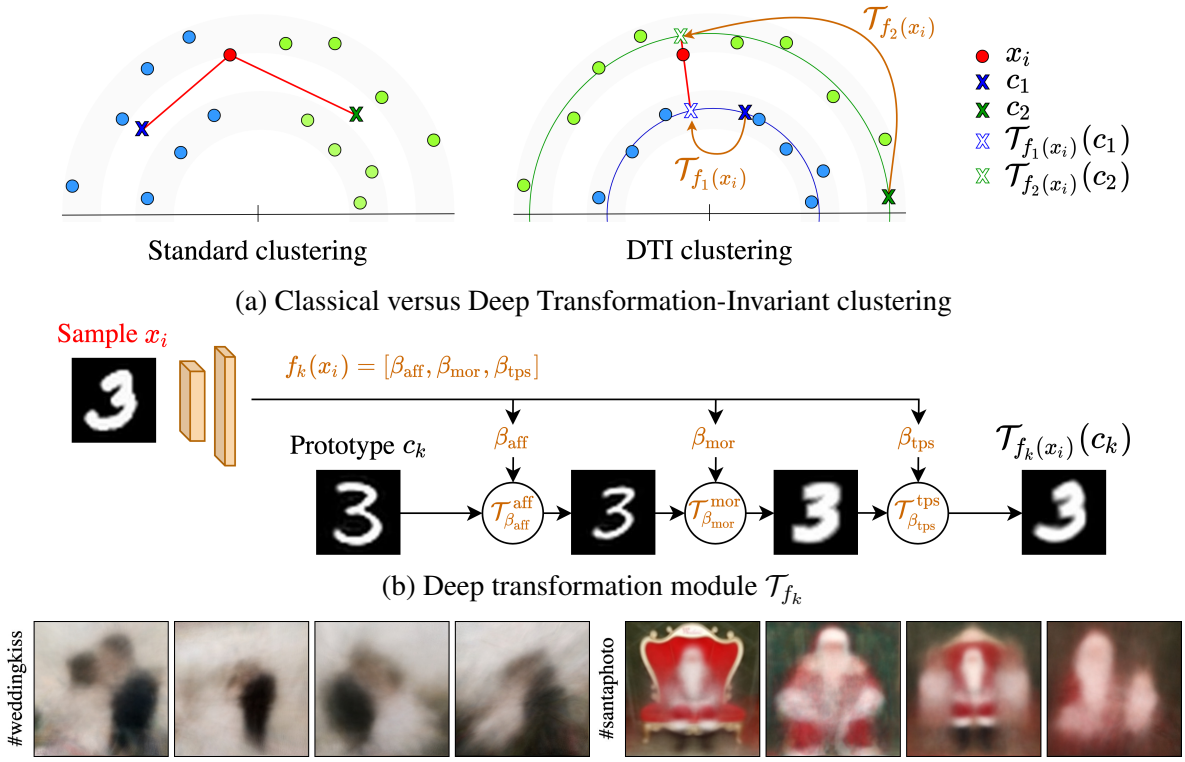
Image features fitting. More recently, there has been a renewed effort to fit 3D primitives to various image representations, such as depth maps, segmentation predictions or low-level image features. Depth-based approaches [Jiang and Xiao, 2013; Fouhey et al., 2013; Lin et al., 2013; Geiger and Wang, 2015; Kluger et al., 2021] naturally associate a 3D point cloud to each image which is then used for primitive fitting. However, the resulting point cloud is highly incomplete, ambiguous and sometimes inaccurately predicted, thus limiting the decomposition quality. Building upon the single-image scene layout estimation from Hoiem et al. [2005, 2007], works like Gupta et al. [2010]; Lee et al. [2010] compute cuboids that best match the predicted surface orientations. Finally, Façade, the classic image-based rendering work developed by Debevec et al. [1996], leverages user annotations across multiple images with known camera viewpoints to render a scene with textured 3D primitives.

In Chapter 6, we present a method that *does not* rely on 3D, depth, segmentation, low-level features, or user annotations to compute the 3D decomposition. Instead, taking inspiration from Façade [Debevec et al., 1996] and recent multi-view modeling advances [Mildenhall

et al., 2020], we only require calibrated views of the scene and we directly optimize textured primitives through photometric consistency in an end-to-end fashion. That is, we solve the 3D decomposition and multi-view stereo problems simultaneously.

Chapter 3

Deep Transformation-Invariant Clustering



(a) Classical versus Deep Transformation-Invariant clustering

(b) Deep transformation module $\mathcal{T}_{f_k}$

(c) Prototypes learned from unfiltered Instagram images associated to different hashtags

Figure 3.1: **Deep Transformation-Invariant Clustering.** **(a)** Given a sample x_i and prototypes c_1 and c_2, standard clustering such as K-means assigns the sample to the closest prototype. Our DTI clustering first aligns prototypes to the sample using a family of parametric transformations - here rotations - then picks the prototype whose alignment yields the smallest distance. **(b)** We predict alignment with deep learning. Given an image x_i, each parameter predictor f_k predicts parameters for a sequence of transformations - here affine $\mathcal{T}^{\text{aff}}_{\beta_{\text{aff}}}$, morphological $\mathcal{T}^{\text{mor}}_{\beta_{\text{mor}}}$, and thin plate spline $\mathcal{T}^{\text{tps}}_{\beta_{\text{tps}}}$ - to align prototype c_k to x_i. **(c)** We apply our method with 40 clusters on large Instagram image sets (15k each).

3.1 Introduction

Gathering collections of images on a topic of interest is getting easier every day: simple tools can aggregate data from social media, web search, or specialized websites and filter it using hashtags, GPS coordinates, or semantic labels. However, identifying visual trends in such image collections remains difficult and usually involves manually organizing images or designing an ad hoc algorithm. Our goal in this chapter is to design a clustering method which can be applied to such image collections, output a visual representation for each cluster and show how it relates to every associated image.

Directly comparing image pixels to decide if they belong to the same cluster leads to poor results because they are strongly impacted by factors irrelevant to clustering, such as exact viewpoint or lighting. Approaches to obtain clusters invariant to these transformations can be broadly classified into two groups. A first set of methods extracts invariant features and performs clustering in feature space. The features can be manually designed, but most state-of-the-art methods learn them directly from data. This is challenging because images are high-dimensional and learning relevant invariances thus requires huge amounts of data. For this reason, while recent approaches perform well on simple datasets like MNIST, they still struggle with real images. Another limitation of these approaches is that learned features are hard to interpret and visualize, making clustering results difficult to analyze. A second set of approaches, following the seminal work of Frey and Jojic [1999, 2002, 2003] on transformation-invariant clustering, uses explicit transformation models to align images before comparing them. These approaches have several potential advantages: (i) they enable direct control of the invariances to consider; (ii) because they do not need to discover invariances, they are potentially less data-hungry; (iii) since images are explicitly aligned, clustering process and results can easily be visualized. However, transformation-invariant approaches require solving a difficult joint optimization problem. In practice, they are thus often limited to small datasets and simple transformations, such as affine transformations, and to the best of our knowledge they have never been evaluated on large standard image clustering datasets.

In this chapter, we propose a deep transformation-invariant (DTI) framework that enables to perform transformation-invariant clustering at scale and uses complex transformations. Our main insight is to jointly learn deep alignment and clustering parameters with a single loss, relying on the gradient-based adaptations of K-means [MacQueen, 1967] and GMM optimization [Dempster et al., 1977]. Not only is predicting transformations more computationally efficient than optimizing them, but it enables us to use complex color, thin plate spline and morphological transformations without any specific regularization. Because it is pixel-based, our deep transformation-invariant clustering is also easy to interpret: cluster centers and image alignments can be visualized to understand assignments. Despite its apparent simplicity, we

demonstrate that our DTI clustering framework leads to results on par with the most recent feature learning approaches on standard benchmarks. We also show it is capable of discovering meaningful modes in real photograph collections, which we see as an important step to bridge the gap between theoretically well-grounded clustering approaches and semi-automatic tools relying on hand-designed features for exploring image collections, such as AverageExplorer [Zhu et al., 2014] or ShadowDraw [Lee et al., 2011].

We first present our DTI framework in Section 3.2 (Figure 3.1a).Then, Section 3.3 introduces our deep transformation modules and architecture (Figure 3.1b) and discuss training details. Finally, Section 3.4 presents and analyzes our results (Figure 3.1c).

Contributions. In this chapter we present:

- a deep transformation-invariant clustering approach that jointly learns to cluster and align images,
- a deep image transformation module to learn spatial alignment, color modifications and for the first time morphological transformations,
- an experimental evaluation showing that our approach is competitive on standard image clustering benchmarks, improving over state-of-the-art on Fashion-MNIST and SVHN, and provides highly interpretable qualitative results even on challenging web image collections.

Code, data, models as well as more visual results are available on our project webpage: www.tmonnier.com/DTIClustering.

3.2 Approach

In this section, we first discuss a generic formulation of our deep transformation-invariant clustering approach, then derive two algorithms based on K-means [MacQueen, 1967] and Gaussian mixture model [Dempster et al., 1977].

Notations. In all the rest of the paper, we use the notation $a_{1:n}$ to refer to the set $\{a_1, \ldots, a_n\}$.

3.2.1 DTI framework

Contrary to most recent image clustering methods which rely on feature learning, we propose to perform clustering in pixel space by making the clustering invariant to a family of transformations. We consider N image samples $x_{1:N}$ and aim at grouping them in K clusters using a *prototype method*. More specifically, each cluster k is defined by a prototype c_k, which can

also be seen as an image, and prototypes are optimized to minimize a loss $\mathcal{L}$ which typically evaluates how well they represent the samples. We further assume that $\mathcal{L}$ can be written as a sum of a loss l computed over each sample:

$$\mathcal{L}(c_{1:K}) = \sum_{i=1}^{N} l(x_i,\ \{c_1, \dots, c_K\}). \tag{3.1}$$

Once the problem is solved, each sample x_i will be associated to the closest prototype.

Our key assumption is that in addition to the data, we have access to a group of parametric transformations $\{\mathcal{T}_\beta,\ \beta \in B\}$ to which we want to make the clustering invariant. For example, one can consider $\beta \in \mathbb{R}^6$ and $\mathcal{T}_\beta$ the 2D affine transformation parametrized by β. Other transformations are discussed in Section 3.3.1. Instead of finding clusters by minimizing the loss of Equation (3.1), one can minimize the following transformation-invariant loss:

$$\mathcal{L}_{\text{TI}}(c_{1:K}) = \sum_{i=1}^{N} \min_{\beta_{1:K}}\ l(x_i,\ \{\mathcal{T}_{\beta_1}(c_1), \dots, \mathcal{T}_{\beta_K}(c_K)\}). \tag{3.2}$$

In this equation, the minimum over $\beta_{1:K}$ is taken for each sample independently. This loss is invariant to transformations of the prototypes (see proof in Appendix A.2). Also note there is not a single optimum since the loss is the same if any prototype c_k is replaced by $\mathcal{T}_\beta(c_k)$ for any $\beta \in B$. If necessary, for example for visualization purposes, this ambiguity can easily be resolved by adding a small regularization on the transformations. The optimization problem associated to $\mathcal{L}_{\text{TI}}$ is of course difficult. A natural approach, which we use as baseline (noted TI), is to alternatively minimize over transformations and clustering parameters. We show that performing such optimization using a gradient descent can already lead to improved results over standard clustering but is computationally expensive.

We experimentally show it is faster and actually better to instead learn K (deep) predictors $f_{1:K}$ for each prototype, which aim at associating to each sample x_i the transformation parameters $f_{1:K}(x_i)$ minimizing the loss, i.e. to minimize the following loss:

$$\mathcal{L}_{\text{DTI}}(c_{1:K}, f_{1:K}) = \sum_{i=1}^{N} l(x_i,\ \{\mathcal{T}_{f_1(x_i)}(c_1), \dots, \mathcal{T}_{f_K(x_i)}(c_K)\}), \tag{3.3}$$

where predictors $f_{1:K}$ are now shared for all samples. We found that using deep parameters predictors not only enables more efficient training but also leads to better clustering results especially with more complex transformations. Indeed, the structure and optimization of the predictors naturally regularize the parameters for each sample, without requiring any specific

Algorithm 1: Deep Transformation-Invariant Gaussian Mixture Model

Input: data $\mathbf{X}$, number of clusters K, transformation $\mathcal{T}$
Output: cluster assignations, Gaussian parameters $\mu_{1:K}, \Sigma_{1:K}$, deep predictors $f_{1:K}$
Initialization: $\mu_{1:K}$ with random samples, $\Sigma_{1:K} = 0.5$, $\eta_{1:K} = 1$ and $\forall k, \forall x, \mathcal{T}_{f_k(x)} = \mathrm{Id}$

1 **while** *not converged* **do**
2 i. sample a batch of data points $x_{1:N}$
3 ii. compute mixing probabilities: $\pi_{1:K} = \mathrm{softmax}(\eta_{1:K})$
4 iii. compute per-sample Gaussian transformed parameters:
$$\forall k,\ \forall i,\quad \tilde{\mu}_{ki} = \mathcal{T}_{f_k(x_i)}(\mu_k) \ \text{ and } \ \tilde{\Sigma}_{ki} = \mathcal{T}^*_{f_k(x_i)}(\Sigma_k) + \mathrm{diag}(\sigma^2_{\min})$$
5 iv. compute responsibilities: $\forall k,\ \forall i,\ \gamma_{ki} = \frac{\pi_k G(x_i\,;\tilde{\mu}_{ki},\tilde{\Sigma}_{ki})}{\sum_j \pi_j G(x_i\,;\tilde{\mu}_{ji},\tilde{\Sigma}_{ji})}$ (*E-step*)
6 v. minimize expected negative log-likelihood w.r.t to $\{\mu_{1:K}, \Sigma_{1:K}, \eta_{1:K}, f_{1:K}\}$:
$$\mathbb{E}[\mathcal{L}_{\text{DTI GMM}}] = -\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ki}\Big(\log\Big(G(x_i\,;\tilde{\mu}_{ki},\tilde{\Sigma}_{ki})\Big) + \log(\pi_k)\Big) \qquad (\textit{M-step})$$
7 **end**

regularization loss, especially in the case of high numbers N of samples and transformation parameters.

In the next section we present concrete losses and algorithms. We then describe differentiable modules for relevant transformations and discuss parameter predictor architecture as well as training in Section 3.3.

3.2.2 Application to K-means and GMM

K-means. The goal of K-means algorithm [MacQueen, 1967] is to find a set of prototypes $c_{1:K}$ such that the average Euclidean distance between each sample and the closest prototype is minimized. Following the reasoning of Section 3.2.1, the loss optimized in K-means can be transformed into a transformation-invariant loss:

$$\mathcal{L}_{\text{DTI K-means}}(c_{1:K}, f_{1:K}) = \sum_{i=1}^{N} \min_k \|x_i - \mathcal{T}_{f_k(x_i)}(c_k)\|^2. \tag{3.4}$$

Following batch gradient-based trainings [Bottou and Bengio, 1995] of K-means, we can then simply jointly minimize $\mathcal{L}_{\text{DTI K-means}}$ over prototypes $c_{1:K}$ and deep transformation parameter predictors $f_{1:K}$ using a batch gradient descent algorithm. In practice, we initialize prototypes $c_{1:K}$ with random samples and predictors $f_{1:K}$ such that $\forall k, \forall x, \mathcal{T}_{f_k(x)} = \mathrm{Id}$.

Gaussian mixture model. We now consider that data are observations of a mixture of K multivariate normal random variables $X_{1:K}$, i.e. $X = \sum_k \delta_{k,\Delta} X_k$ where δ is the Kronecker function and $\Delta \in \{1, \dots, K\}$ is a random variable defined by $P(\Delta = k) = \pi_k$, with $\forall k,\ \pi_k > 0$ and $\sum_k \pi_k = 1$. We write μ_k and Σ_k the mean and covariance of X_k and $G(\,\cdot\,; \mu_k, \Sigma_k)$ associated probability density function. The transformation-invariant negative log-likelihood can then be written:

$$\mathcal{L}_{\text{DTI GMM}}(\mu_{1:K}, \Sigma_{1:K}, \pi_{1:K}, f_{1:K}) = -\sum_{i=1}^{N} \log \Big(\sum_{k=1}^{K} \pi_k G\Big(x_i\,; \mathcal{T}_{f_k(x_i)}(\mu_k), \mathcal{T}^*_{f_k(x_i)}(\Sigma_k)\Big)\Big), \quad (3.5)$$

where $\mathcal{T}^*$ is slightly modified version of $\mathcal{T}$. Indeed, $\mathcal{T}$ may include transformations that one can apply to the covariance, such as spatial transformations, and other that would not make sense, such as additive color transformations. We jointly minimize $\mathcal{L}_{\text{DTI GMM}}$ over Gaussian parameters, mixing probabilities, and deep transformation parameters $f_{1:K}$ using a batch gradient-based EM procedure similar to [Hosseini and Sra, 2015; Greff et al., 2017; Gepperth and Pfülb, 2019] and detailed in Algorithm 1. In practice, we assume that pixels are independent resulting in diagonal covariance matrices.

In such gradient-based procedures, two constraints have to be enforced, namely the positivity and normalization of mixing probabilities π_k and the non-negativeness of the diagonal covariance terms. For the mixing probabilities constraints, we adopt the approach used in [Hosseini and Sra, 2015] and [Gepperth and Pfülb, 2019] which optimize mixing parameters η_k used to compute the probabilities π_k using a softmax instead of directly optimizing π_k, which we write $\pi_{1:K} = \mathrm{softmax}(\eta_{1:K})$. For the variance non-negativeness, we introduce a fixed minimal variance value σ^2_{min} which is added to the variances when evaluating the probability density function. This approach is different from the one in [Gepperth and Pfülb, 2019] which instead use clipping, because we found training with clipped values was harder. In practice, we take $\sigma_{\text{min}} = 0.25$.

3.3 Learning image transformations

3.3.1 Architecture and transformation modules

We consider a set of prototypes $c_{1:K}$ we would like to transform to match a given sample x. To do so, we propose to learn for each prototype c_k, a separate deep predictor which predicts transformation parameters β. We propose to model the family of transformations $\mathcal{T}_\beta$ as a sequence of M parametric transformations such that, writing $\beta = (\beta^1, \dots, \beta^M)$, $\mathcal{T}_\beta = \mathcal{T}^M_{\beta^M} \circ \dots \circ \mathcal{T}^1_{\beta^1}$. In the following, we describe the architecture of transformation parameter

predictors $f_{1:K}$, as well as each family of parametric transformation modules we use. Figure 3.1b shows our learned transformation process on a MNIST example.

Parameters prediction network. For all experiments, we use the same parameter predictor network architecture composed of a shared ResNet [He et al., 2016] backbone truncated after the global average pooling, followed by $K \times M$ Multi-Layer Perceptrons (MLPs), one for each prototype and each transformation module. For the ResNet backbone, we use ResNet-20 for images smaller than 64×64 and ResNet-18 otherwise. Each MLP has the same architecture, with two hidden layers of 128 units.

Spatial transformer module. To model spatial transformations of the prototypes, we follow the spatial transformers developed by Jaderberg et al. [2015]. The key idea is to model spatial transformations as a differentiable image sampling of the input using a deformed sampling grid. We use affine $\mathcal{T}_\beta^{\text{aff}}$, projective $\mathcal{T}_\beta^{\text{proj}}$ and thin plate spline $\mathcal{T}_\beta^{\text{tps}}$ [Bookstein, 1989] transformations which respectively correspond to 6, 8 and 16 (a 4x4 grid of control points) parameters.

Color transformation module. We model color transformation with a channel-wise diagonal affine transformation on the full image, which we write $\mathcal{T}_\beta^{\text{col}}$. It has 2 parameters for greyscale images and 6 parameters for colored images. We first used a full affine transformation with 12 parameters, however the network was able to hide several patterns in the different color channels of a single prototype (Appendix A.3). Note that a similar transformation was theoretically introduced in capsules [Kosiorek et al., 2019], but with the different goal of obtaining a color-invariant feature representation. Deep feature-based approaches often handle color images with a pre-processing step such as Sobel filtering [Caron et al., 2018; Ji et al., 2019; Kosiorek et al., 2019]. We believe the way we align colors of the prototypes to obtain color invariance in pixel space is novel, and it enables us to directly work with colored images without using any pre-processing or specific invariant features.

Morphological transformation module. We introduce a new transformation module to learn morphological operations [Haralick et al., 1987] such as dilation and erosion. We consider a greyscale image $x \in \mathbb{R}^D$ of size $U \times V = D$, we write $x[u, v]$ the value of the pixel (u, v) for $u \in \{1, \dots, U\}$ and $v \in \{1, \dots, V\}$. Given a 2D region A, the dilation of x by A, $\mathcal{D}_A(x) \in \mathbb{R}^D$, is defined by $\mathcal{D}_A(x)[u, v] = \max_{(u',v') \in A} \; x[u + u', v + v']$ and its erosion by A, $\mathcal{E}_A(x) \in \mathbb{R}^D$, is defined by $\mathcal{E}_A(x)[u, v] = \min_{(u',v') \in A} \; x[u + u', v + v']$. Directly learning the region A which parametrizes these transformations is challenging, we thus propose to learn

parameters (α, a) for the following soft version of these transformations:

$$\mathcal{T}^{\text{mor}}_{(\alpha,a)}(x)[u,v] = \frac{\sum_{(u',v')\in W} \; x[u+u',v+v'] \cdot a[u+u',v+v'] \cdot e^{\alpha x[u+u',v+v']}}{\sum_{(u',v')\in W} \; a[u+u',v+v'] \cdot e^{\alpha x[u+u',v+v']}}, \tag{3.6}$$

where W is a fixed set of 2D positions, α is a softmax (positive values) or softmin (negative values) parameter and a is a set of parameters with values between 0 and 1 defined for every position $(u', v') \in W$. Parameters a can be interpreted as an image, or as a soft version of the region A used for morphological operations. Note that if $a[u', v'] = \mathbf{1}_{\{(u',v')\in A\}}$, when $\alpha \to +\infty$ (resp. $-\infty$), it successfully emulates $\mathcal{D}_A$ (resp. $\mathcal{E}_A$). In practice, we use a grid of integer positions around the origin of size 7×7 for W. Note that since morphological transformations do not form a group, transformation-invariant denomination is slightly abusive.

3.3.2 Training

We found that two key elements were critical to obtain good results: empty cluster reassignment and curriculum learning. We then discuss further implementation details and computational cost.

Empty cluster reassignment. Similar to [Caron et al., 2018], we adopt an empty cluster reassignment strategy during our clustering optimization. We reinitialize both prototype and deep predictor of "tiny" clusters using the parameters of the largest cluster with a small added noise. In practice, the size of balanced clusters being N/K, we define "tiny" as less than 20% of N/K.

Curriculum learning. Learning to predict transformations is a hard task, especially when the number of parameters is high. To ease learning, we thus adopt a curriculum learning strategy by gradually adding more complex transformation modules to the training. Given a target sequence of transformations to learn, we first train our model without any transformation - or equivalently with an identity module - then iteratively add subsequent modules once convergence has been reached. We found this is especially important when modeling local deformations with complex transformations with many parameters, such as TPS and morphological transformations. Intuitively, prototypes should first be coarsely aligned before attempting to refine the alignment with more complex transformations.

Implementation details. Both clustering parameters and parameter prediction networks are learned jointly and end-to-end using Adam optimizer [Kingma and Ba, 2015] with a 10^{-6}

Method	Runs	Eval	MNIST ACC	MNIST NMI	MNIST-test ACC	MNIST-test NMI	USPS ACC	USPS NMI	F-MNIST ACC	F-MNIST NMI	FRGC ACC	FRGC NMI	SVHN ACC
Clustering on a learned feature													
DEC [Xie et al., 2016]	9	max	86.3	83.4	85.6	83.0	76.2	76.7	51.8	54.6	37.8	50.5	-
InfoGAN [Chen et al., 2016]	5	max	89.0	86.0	-	-	-	-	61.0	59.0	-	-	-
VaDE [Jiang et al., 2017]	10	max	94.5	87.6	-	-	56.6	51.2	57.8	63.0	-	-	-
ClusterGAN [Mukherjee et al., 2019]	5	max	95.0	89.0	-	-	-	-	63.0	64.0	-	-	-
JULE [Yang et al., 2016]	3	avg	96.4	91.3	96.1	91.5	95.0	91.3	56.3	60.8	46.1	57.4	-
DEPICT [Dizaji et al., 2017]	5	avg	96.5	91.7	96.3	91.5	**96.4**	**92.7**	39.2	39.2	**47.0**	**61.0**	-
DSCDAN [Yang et al., 2019]	10	avg	**97.8**	**94.1**	**98.0**	**94.6**	86.9	85.7	**66.2**	**64.5**	-	-	-
Clustering on a learned feature with data augmentation and/or ad hoc data representation													
SpectralNet [Shaham et al., 2018]	5	avg	97.1§	**92.4**§	-	-	-	-	-	-	-	-	-
IMSAT [Hu et al., 2017]	12	avg	98.4$^{\triangledown}$	-	-	-	-	-	-	-	-	-	**57.3**$^{\triangledown\dagger}$
ADC [Häusser et al., 2018]	20	avg	98.7$^{\triangledown}$	-	-	-	-	-	-	-	**43.7**$^{\triangledown}$	-	38.6$^{\triangledown}$
SCAE [Kosiorek et al., 2019]	5	avg	98.7$^{\triangledown}$	-	-	-	-	-	-	-	-	-	55.3‡
IIC [Ji et al., 2019]	5	avg	98.4$^{\triangledown}$	-	-	-	-	-	-	-	-	-	-
	5	minLoss	**99.2**$^{\triangledown}$	-	-	-	-	-	-	-	-	-	-
Clustering on pixel values													
K-means [MacQueen, 1967]	10	avg	54.8	50.2	55.9	51.2	65.3	61.2	54.1	51.4	22.7	26.5	12.2
GMM [Dempster et al., 1977]	10	avg	54.2	51.7	55.6	54.7	66.0	60.9	49.7	51.2	24.2	27.9	11.6
DTI K-means	10	avg	**97.3**	**94.0**	96.6	94.6	86.4	88.2	61.2	63.7	39.6	48.7	36.4 / 44.5*
	10	minLoss	97.2	93.8	**98.0**	**95.3**	**89.8**	**89.5**	57.4	64.1	41.1	49.7	39.6 / 62.6*
DTI GMM	10	avg	95.9	93.2	97.8	94.7	84.5	87.2	59.6	62.2	40.1	48.9	36.7 / 57.4*
	10	minLoss	97.1	93.7	**98.0**	95.1	87.3	89.0	**68.2**	**66.3**	**41.6**	**51.1**	39.5 / **63.3***

Table 3.1: **Comparisons.** We report ACC and NMI in % on standard clustering benchmarks. Symbols mark methods that use data augmentation ($\triangledown$) and manually selected features as input ($\S$ for pretrained features from best VaDE run, $\dagger$ for GIST features, $\ddagger$ for Sobel filters) and are thus not directly comparable. For SVHN, we also report our results with our Gaussian weighted loss ($\star$). Eval column refers to the aggregate used: best run (*max*), average (*avg*) or run with minimal loss (*minLoss*).

weight decay on the neural network parameters. We sequentially add transformation modules at a constant learning rate of 0.001 then divide the learning rate by 10 after convergence - corresponding to different numbers of epochs depending on the dataset characteristics - and train for a few more epochs with the smaller learning rate. We use a batch size of 64 for real photograph collections and 128 otherwise.

Computational cost. Training DTI K-means or DTI GMM on MNIST takes approximately 50 minutes on a single Nvidia GeForce RTX 2080 Ti GPU and full dataset inference takes 30 seconds. We found it to be much faster than directly optimizing transformation parameters (TI clustering) for which convergence took more than 10 hours of training.

3.4 Experiments

In this section, we first analyze our approach and compare it to state-of-the-art, then showcase its interest for image collection analysis and visualization.

(a) Prototypes learned for different datasets

(b) Transformations predicted for all prototypes for 4 MNIST images

Figure 3.2: **Qualitative results. (a)** compares prototypes learned from GMM and our DTI GMM, **(b)** shows transformed prototypes given query samples from MNIST and highlight the closest prototype.

3.4.1 Analysis and comparisons

Similar to previous work on image clustering, we evaluate our approach with global classification accuracy (ACC), where a cluster-to-class mapping is computed using the Hungarian algorithm [Kuhn and Yaw, 1955], and Normalized Mutual Information (NMI). Datasets and corresponding transformation modules we used are described in Appendix A.1.

Comparison on standard benchmarks. In Table 3.1, we report our results on standard image clustering benchmarks, i.e. digit datasets (MNIST [LeCun et al., 1998], USPS [Hastie et al., 2001]), a clothing dataset (Fashion-MNIST [Xiao et al., 2017]) and a face dataset (FRGC [Phillips et al., 2005]). We also report results for SVHN [Netzer et al., 2011] where concurrent methods use pre-processing to remove color bias. In the table, we separate representation-based from pixel-based methods and mark results using data augmentation or manually selected features as input. Note that our results depend on initialization, we provide detailed statistics in Appendix A.3.

Our DTI clustering is fully unsupervised and does not require any data augmentation, ad hoc features, nor any hyper-parameter while performing clustering directly in pixel space. We report average performances and performances of the minimal loss run which we found to correlate well with high performances (Appendix A.3). Because this non-trivial criterion allows to automatically select a run in a fully unsupervised way, we argue it can be compared to average results from competing methods which don't provide such criterion.

First, DTI clustering achieves competitive results on all datasets, in particular improving state-of-the-art by a significant margin on SVHN and Fashion-MNIST. For SVHN, we first found that the prototypes quality was harmed by digits on the side of the image. To pay more

Method	Eval	MNIST-1k	MNIST-color	affNIST-test
[Jiang et al., 2017]	avg	49.6 (5.6)	11.9 (1.2)	Div.
[Hu et al., 2017]	avg	67.9 (2.3)	10.6 (0.1)	18.2 (2.6)
[Ji et al., 2019]	avg	63.4 (0.4)	10.6 (0.0)	57.6 (0.0)
	minLoss	63.2	10.6	57.6
DTI K-means	avg	79.8 (6.9)	96.7 (0.1)	95.5 (3.3)
	minLoss	**90.5**	**96.8**	**97.0**
DTI GMM	avg	80.8 (7.2)	96.0 (0.2)	93.3 (5.9)
	minLoss	87.1	95.8	**97.0**

(a) **Augmented and specific datasets.** Clustering accuracy (%) with standard deviation for methods applied on raw images (no pre-processing). We used 10 runs for our method and 5 for the baselines.

Method	Avg	MinLoss
DTI clustering (aff-morpho-tps)	**97.3**	**97.2**
ordering: aff-tps-morpho	95.5	96.9
ordering: morpho-aff-tps	27.5	97.0
w/o morphological	94.8	95.8
w/o thin plate spline	90.0	82.5
w/o affine	85.1	96.8
affine only	90.1	90.5
w/o empty cluster reassignment	80.9	78.6
w/o curriculum learning	83.9	78.9
TI clustering (aff-morpho-tps, 1 run)	26.3	26.3
TI clustering (affine only)	73.0	73.1

(b) **Ablation study on MNIST.** Clustering accuracy (%) over 10 runs.

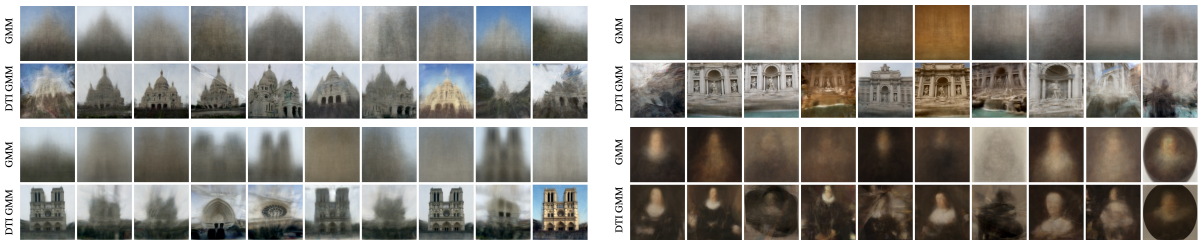
Table 3.2: **Additional quantitative results.**

attention to the center digit, we weighted the clustering loss by a Gaussian weight ($\sigma = 7$). It led to better prototypes and allowed us to improve over all concurrent methods by a large margin. Compared to representation-based methods, our pixel-based clustering is highly interpretable. Figure 3.2a shows standard GMM prototypes and our prototypes learned with DTI GMM which appear to be much sharper than standard ones. This directly stems from the quality of the learned transformations, visualized in Figure 3.2b. Our transformation modules can successfully align the prototype, adapt the thickness and apply local elastic deformations. More alignment results are available on our project webpage.

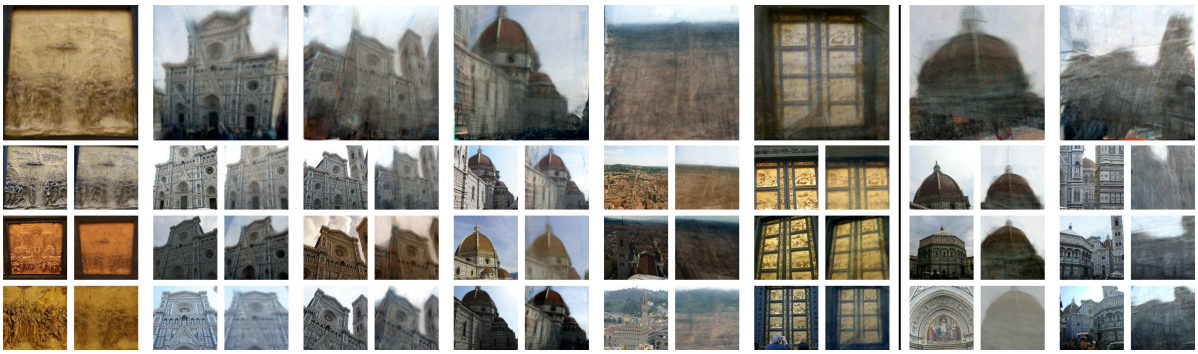
Augmented and specific datasets. DTI clustering also works on small, colored and misaligned datasets. In Table 3.2a, we highlight these strengths on specifics datasets generated from MNIST: MNIST-1k is a 1000 images subset, MNIST-color is obtained by randomly selecting a color for the foreground and background and affNIST-test[1] is the result of random affine transformations. We used an online implementation[2] for VaDE [Jiang et al., 2017] and official ones for IMSAT [Hu et al., 2017] and IIC [Ji et al., 2019] to obtain baselines. Our results show that the performances of DTI clustering is barely affected by spatial and color transformations, while baseline performances drop on affNIST-test and are almost chance on MNIST-color. Figure 3.2a shows the quality and interpretability of our cluster centers on affNIST-test and MNIST-color. DTI clustering also seems more data-efficient than the baselines we tested.

Ablation on MNIST. In Table 3.2b, we conduct an ablation study on MNIST of our full model trained following Section 3.3.2 with affine, morphological and TPS transformations.

[1] https://www.cs.toronto.edu/~tijmen/affNIST/
[2] https://github.com/GuHongyang/VaDE-pytorch



(a) Full sets of prototypes discovered with GMM and DTI GMM



(b) Examples of cluster centers and aligned images with DTI GMM (20 clusters)

Figure 3.3: **Qualitative results on real photographs.** **(a)** Clustering results from photographs of different locations in Li and Snavely [2018] (1,089 Sacre Coeur top-left, 1,688 Trevi fountain top-right, 2,625 Notre-Dame bottom-left) and 980 Baroque portraits from Karayev et al. [2014] (bottom-right). **(b)** Clustering results from 1,892 Florence cathedral images from Li and Snavely [2018]. Top row shows learned prototypes while the three bottom rows show examples of images from each cluster and aligned prototypes. These clusters contain respectively 44, 154, 134, 64, 71, 133, 85 and 64 images. The left six examples are successful clusters while the two right clusters are relative failure cases.

We first explore the effect of transformation modules. Their order is not crucial, as shown by similar minLoss performances, but can greatly affect the stability of the training, as can be seen in the average results. Each module contributes to the final performance, affine transformations being the most important. We then validate our training strategy showing that both empty cluster reassignment and curriculum learning for the different modules are necessary. Finally, we directly optimize the loss of Equation (3.2) (TI clustering) by optimizing the transformation parameters for each sample at each iteration of the batch clustering algorithm, without using our parameter predictors. With rich transformations which have many parameters, such as TPS and morphological ones, this approach fails completely. Using only affine transformations, we obtain results clearly superior to standard clustering, but worse than ours.

3.4.2 Application to web images

One of the main interest of our DTI clustering is that it allows to discover trends in real image collections. All images are resized and center cropped to 128×128. The selection of the number of clusters is a difficult problem and is discussed in Appendix A.3.

In Figure 3.1c, we show examples of prototypes discovered in very large unfiltered sets (15k each) of Instagram images associated to different hashtags[3] using DTI GMM applied with 40 clusters. While many images are noise and are associated to prototypes which are not easily interpretable, we show prototypes where iconic photos and poses can be clearly identified. To the best of our knowledge, we believe we are the first to demonstrate this type of results from raw social network image collections. Comparable results in AverageExplorer [Zhu et al., 2014], e.g. on Santa images, could be obtained using ad hoc features and user interactions, while our results are produced fully automatically.

Figure 3.3 shows qualitative clustering results on MegaDepth [Li and Snavely, 2018] and WikiPaintings [Karayev et al., 2014]. Similar to our results on image clustering benchmarks, our learned prototypes are more relevant and accurate than the ones obtained from standard clustering. Note that some of our prototypes are very sharp: they typically correspond to sets of photographs between which we can accurately model deformations, e.g. scenes that are mostly planar, with little perspective effects. On the contrary, more unique photographs and photographs with strong 3D effects that we cannot model will be associated to less interpretable and blurrier prototypes, such as the ones in the last two columns of Figure 3.3b. In Figure 3.3b, in addition to the prototypes discovered, we show examples of images contained in each cluster as well as the aligned prototype. Even for such complex images, the simple combination of our color and spatial modules manages to model real image transformations like illumination variations and viewpoint changes. More web image clustering results are shown on our project webpage.

3.5 Conclusion

We have introduced an efficient deep transformation-invariant clustering approach in raw input space. Our key insight is the online optimization of a single clustering objective over clustering parameters and deep image transformation modules. We demonstrate competitive results on standard image clustering benchmarks, including improvements over state-of-the-art on SVHN and Fashion-MNIST. We also demonstrate promising results for real photograph collection clustering and visualization. Finally, note that our DTI clustering framework is not specific

[3] https://github.com/arc298/instagram-scraper was used to scrape photographs

to images and can be extended to other types of data as long as appropriate transformation modules are designed beforehand.

Chapter 4

Discovering Objects With Sprite Modeling

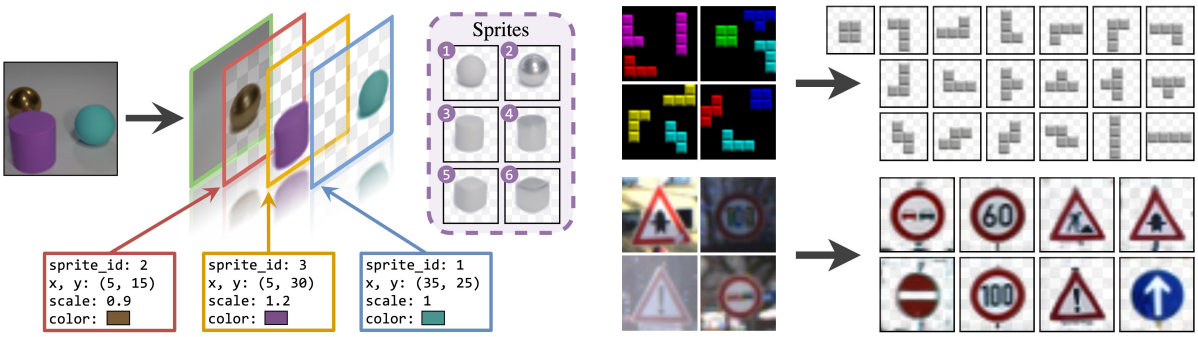


Figure 4.1: **Discovering Objects with Sprite Modeling.** Our approach learns without supervision to decompose images into layers modeled as transformed instances of prototypical objects called *sprites*. We show an example of decomposition on CLEVR [Johnson et al., 2017] (left) and examples of discovered sprites for Tetrominoes [Greff et al., 2019] and GTSRB [Stallkamp et al., 2012] (right). Transparency is visualized using light gray checkerboards.

4.1 Introduction

The aim of this chapter is to learn without any supervision a layered decomposition of images, where each layer is a transformed instance of a prototypical object. Such an interpretable and layered model of images could be beneficial for a plethora of applications like object discovery [Eslami et al., 2016; Burgess et al., 2019], image edition [Wu et al., 2017b; Greff et al., 2019], future frame prediction [Wu et al., 2017a], object pose estimation [Romaszko et al., 2017] or environment abstraction [Anand et al., 2019; Lin et al., 2020b]. Recent works in this direction [Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020] typically learn layered image decompositions by generating layers with autoencoder networks. In contrast, we explicitly model them as transformations of a set of prototypical images with transparency,

which we refer to as *sprites*. These sprites are mapped onto their instances through geometric and colorimetric transformations resulting in what we call *object layers*. An image is then assembled from ordered object layers so that each layer occludes previous ones in regions where they overlap.

Our composition model is reminiscent of the classic computer graphics sprite model, popular in console and arcade games from the 1980s. While classical sprites were simply placed at different positions and composited with a background, we revisit the notion in a spirit similar to Jojic and Frey's work on video modeling [Jojic and Frey, 2001] by using the term in a more generic sense: our sprites can undergo rich geometric transformations and color changes.

We jointly learn in an unsupervised manner both the sprites and parametric functions predicting their transformations to explain images. This is related to the recent deep transformation-invariant (DTI) method designed for clustering by Monnier et al. [2020]. Unlike this work, however, we handle images that involve a variable number of objects with limited spatial supports, explained by different transformations and potentially occluding each other. This makes the problem very challenging because objects cannot be treated independently and the possible number of image compositions is exponential in the number of layers.

We experimentally demonstrate in Section 4.3.1 that our method is on par with the state of the art on the synthetic datasets commonly used for image decomposition evaluation [Greff et al., 2019]. Because our approach explicitly models image compositions and object transformations, it also enables us to perform simple and controlled image manipulations on these datasets. More importantly, we demonstrate that our model can be applied to real images (Section 4.3.2), where it successfully identifies objects and their spatial extent. For example, we report an absolute 5% increase upon the state of the art on the popular SVHN benchmark [Netzer et al., 2011] and good cosegmentation results on the Weizmann Horse database [Borenstein and Ullman, 2004]. We also qualitatively show that our model successfully discriminates foreground from background on challenging sets of social network images.

Contributions. To summarize, we present:

- an unsupervised learning approach that explains images as layered compositions of transformed sprites with a background model;
- strong results on standard synthetic multi-object benchmarks using the usual instance segmentation evaluation, and an additional evaluation on *semantic* segmentation, which to the best of our knowledge has never been reported by competing methods; and
- results on real images for clustering and cosegmentation, which we believe has never been demonstrated by earlier unsupervised image decomposition models.

Code and data are available at our project webpage: www.tmonnier.com/DTI-Sprites.

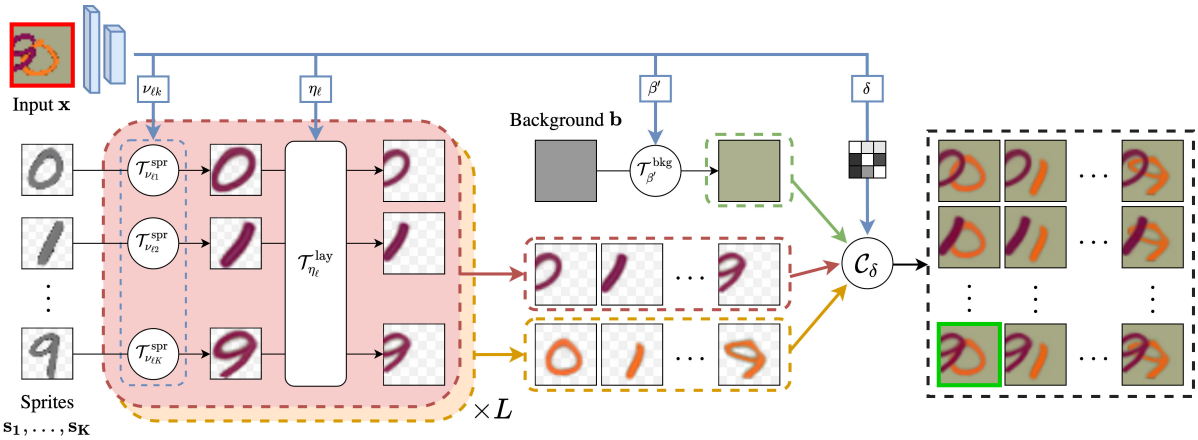


Figure 4.2: **Overview.** Given an input image (highlighted in red) we predict for each layer the transformations to apply to the sprites that best reconstruct the input. Transformed sprites and background can be composed into many possible reconstructions given a predicted occlusion matrix δ. We introduce a greedy algorithm to select the best reconstruction (highlighted in green).

4.2 Approach

In this section, we first present our image formation model (Section 4.2.1), then describe our unsupervised learning strategy (Section 4.2.2). Given N colored images $\mathbf{x}_{1:N}$ of size $H \times W$, we want to learn their decomposition into L object layers defined by the instantiations of K sprites. Figure 4.2 shows an overview of our approach.

Notations. We write $a_{1:n}$ the ordered set $\{a_1, \ldots, a_n\}$, $\odot$ pixel-wise multiplication and use bold notations for images. Given N colored images $\mathbf{x}_{1:N}$ of size $H \times W$, we want to learn their decomposition into L object layers defined by the instantiations of K sprites.

4.2.1 Image formation model

Layered composition process. Motivated by early works on layered image models [Matheron, 1968; Jojic and Frey, 2001], we propose to decompose an image into L object layers $\mathbf{o}_{1:L}$ which are overlaid on top of each other. Each object layer $\mathbf{o}_\ell$ is a four-channel image of size $H \times W$, three channels correspond to a colored RGB appearance image $\mathbf{o}_\ell^c$, and the last one $\mathbf{o}_\ell^\alpha$ is a transparency or alpha channel over $\mathbf{o}_\ell^c$. Given layers $\mathbf{o}_{1:L}$, we define our image formation process as a recursive composition:

$$\forall \ell > 0,\ \mathbf{c}_\ell = \mathbf{o}_\ell^\alpha \odot \mathbf{o}_\ell^c + (\mathbf{1} - \mathbf{o}_\ell^\alpha) \odot \mathbf{c}_{\ell-1}, \tag{4.1}$$

where $\mathbf{c}_0 = \mathbf{0}$, and the final result of the composition is $\mathbf{c}_L$. Note that this process explicitly models occlusion: the first layer corresponds to the farthest object from the camera, and layer L is the closest, occluding all the others. In particular, we model background by using a first layer with $\mathbf{o}_1^\alpha = \mathbf{1}$.

Unfolding the recursive process in Equation (4.1), the layered composition process can be rewritten in the compact form:

$$\mathcal{C}_\delta(\mathbf{o}_1, \dots, \mathbf{o}_L) = \sum_{\ell=1}^{L} \left(\prod_{j=1}^{L} (\mathbf{1} - \delta_{j\ell} \mathbf{o}_j^\alpha) \right) \odot \mathbf{o}_\ell^\alpha \odot \mathbf{o}_\ell^c, \tag{4.2}$$

where $\delta_{j\ell} = \mathbb{1}_{[j>\ell]}$ is the indicator function of $j > \ell$. δ is a $L \times L$ binary matrix we call *occlusion matrix*: for given indices j and ℓ, $\delta_{j\ell} = 1$ if layer j occludes layer ℓ, and $\delta_{j\ell} = 0$ otherwise. This gives Equation (4.2) a clear interpretation: each layer appearance $\mathbf{o}_\ell^c$ is masked by its own transparency channel $\mathbf{o}_\ell^\alpha$ and other layers j occluding it, *i.e.* for which $\delta_{j\ell} = 1$. Note that we explicitly introduce the dependency on δ in the composition process $\mathcal{C}_\delta$ because we will later predict it, which intuitively corresponds to a layer reordering.

Sprite modeling. We model each layer as an explicit transformation of one of K learnable sprites $\mathbf{s}_{1:K}$, which can be seen as prototypes representing the object categories. Each sprite $\mathbf{s}_k$ is a learnable four-channel image of arbitrary size, an RGB appearance image $\mathbf{s}_k^c$ and a transparency channel $\mathbf{s}_k^\alpha$. To handle variable number of objects, we model object absence with an empty sprite $\mathbf{s}_0 = \mathbf{0}$ added to the K sprite candidates and penalize the use of non-empty sprites during learning (see Section 4.2.2). Such modeling assumes we know an upper bound of the maximal number of objects, which is rather standard in such a setting [Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020].

Inspired by the recent deep transformation-invariant (DTI) framework designed for clustering [Monnier et al., 2020], we assume that we have access to a family of differentiable transformations $\mathcal{T}_\beta$ parametrized by β - *e.g.* an affine transformation with β in $\mathbb{R}^6$ implemented with a spatial transformer [Jaderberg et al., 2015] - and we model each layer as the result of the transformation $\mathcal{T}_\beta$ applied to one of the K sprites. We define two sets of transformations for a given layer ℓ: (i) $\mathcal{T}_{\eta_\ell}^{\text{lay}}$ the transformations parametrized by η_ℓ and shared for all sprites in that layer, and (ii) $\mathcal{T}_{\nu_{\ell k}}^{\text{spr}}$ the transformations specific to each sprite and parametrized by $\nu_{\ell k}$. More formally, for given layer ℓ and sprite k we write:

$$\mathcal{T}_{\beta_{\ell k}}(\mathbf{s}_k) = \mathcal{T}_{\eta_\ell}^{\text{lay}} \circ \mathcal{T}_{\nu_{\ell k}}^{\text{spr}}(\mathbf{s}_k), \tag{4.3}$$

where $\beta_{\ell k} = (\eta_\ell, \nu_{\ell k})$ and $\mathcal{T}_{(\eta_\ell, \nu_{\ell k})} = \mathcal{T}_{\eta_\ell}^{\text{lay}} \circ \mathcal{T}_{\nu_{\ell k}}^{\text{spr}}$.

Although it could be included in $\mathcal{T}^{\text{spr}}_{\nu_{\ell k}}$, we separate $\mathcal{T}^{\text{lay}}_{\eta_\ell}$ to constrain transformations and avoid bad local minima. For example, we use it to model a coarse spatial positioning so that all sprites in a layer attend to the same object in the image. On the contrary, we use $\mathcal{T}^{\text{spr}}_{\nu_{\ell k}}$ to model sprite specific deformations, such as local elastic deformations.

When modeling background, we consider a distinct set of K' background prototypes $\mathbf{b}_{1:K'}$, without transparency, and different families of transformations $\mathcal{T}^{\text{bkg}}_{\beta'}$. For simplicity, we write the equations for the case without background and omit sprite-specific transformations in the rest of the chapter, writing $\mathcal{T}_{\beta_\ell}(\mathbf{s}_k)$ instead of $\mathcal{T}_{\beta_{\ell k}}(\mathbf{s}_k)$.

To summarize, our image formation model is defined by the occlusion matrix δ, the per-layer sprite selection $(k_1, \dots, k_L)$, the corresponding transformation parameters $(\beta_1, \dots, \beta_L)$, and outputs an image $\mathbf{x}$ such that:

$$\mathbf{x} = \mathcal{C}_\delta\Big(\mathcal{T}_{\beta_1}(\mathbf{s}_{k_1}), \dots, \mathcal{T}_{\beta_L}(\mathbf{s}_{k_L})\Big). \tag{4.4}$$

We illustrate our image formation model in Figure 4.1 and provide a detailed example in Figure 4.2.

4.2.2 Learning

We learn our image model without any supervision by minimizing the objective function:

$$\mathcal{L}(\mathbf{s}_{1:K}, \phi_{1:L}, \psi) = \sum_{i=1}^{N} \min_{k_1,\dots,k_L} \left(\lambda \sum_{j=1}^{L} \mathbb{1}_{[k_j \neq 0]} + \left\| \mathbf{x}_i - \mathcal{C}_{\psi(\mathbf{x}_i)}\Big(\mathcal{T}_{\phi_1(\mathbf{x}_i)}(\mathbf{s}_{k_1}), \dots, \mathcal{T}_{\phi_L(\mathbf{x}_i)}(\mathbf{s}_{k_L})\Big) \right\|_2^2 \right), \tag{4.5}$$

where $\mathbf{s}_{1:K}$ are the sprites, $\phi_{1:L}$ and ψ are neural networks predicting the transformation parameters and occlusion matrix for a given image $\mathbf{x}_i$, λ is a scalar hyper-parameter and $\mathbb{1}_{[k_j \neq 0]}$ is the indicator function of $k_j \neq 0$. The first sum is over all images in the database, the minimum corresponds to the selection of the sprite used for each layer and the second sum counts the number of non-empty sprites. If $\lambda > 0$, this loss encourages reconstructions using the minimal number of non-empty sprites. In practice, we use $\lambda = 10^{-4}$.

Note the similarity between our loss and the gradient-based adaptation [Bottou and Bengio, 1995] of the K-means algorithm [MacQueen, 1967] where the squared Euclidean distance to the closest prototype is minimized, as well as with its transformation-invariant version [Monnier et al., 2020] including neural networks modeling transformations. In addition to the layered composition model described in the previous section, the main two differences with our model are the joint optimization over L sprite selections and the occlusion modeling that we discuss next.

Algorithm 2: Greedy sprite selection.

Input: image $\mathbf{x}$, occlusion δ, $(K+1) \times L$ object layers candidates $\mathcal{T}_{\phi_\ell(\mathbf{x})}(\mathbf{s}_k)$, steps T
Output: sprite indices $(k_1, \dots, k_L)$
Initialization: $\forall \ell \in \{1, \dots, L\},\ k_\ell \leftarrow 0, \mathbf{o}_\ell \leftarrow \mathbf{0}$

```
for t = 1, ..., T do                                   # iterations
    for ℓ = 1, ..., L do                               # loop on layers
        k_ℓ ← min_k [ λ 1_[k≠0] +
                ‖x − C_δ(o_{1:ℓ−1}, T_{φ_ℓ(x)}(s_k), o_{ℓ+1:L})‖_2^2 ]
        o_ℓ ← T_{φ_ℓ(x)}(s_{k_ℓ})
    end
end
return k_1, ..., k_L
```

Sprites selection. Because the minimum in Equation (4.5) is taken over the $(K+1)^L$ possible selections leading to as many reconstructions, an exhaustive search over all combinations quickly becomes impossible when dealing with many objects and layers. Thus, we propose an iterative greedy algorithm to estimate the minimum, described in Algorithm 2 and used when $L > 2$. While the solution it provides is of course not guaranteed to be optimal, we found it performs well in practice. At each iteration, we proceed layer by layer and iteratively select for each layer the sprite k_ℓ minimizing the loss, keeping all other object layers fixed. This reduces the number of reconstructions to perform to $T \times (K+1) \times L$. In practice, we have observed that convergence is reached after 1 iteration for Tetrominoes and 2-3 iterations for Multi-dSprites and CLEVR6, so we have respectively used $T = 1$ and $T = 3$ in these experiments. We experimentally show in our ablation study presented in Section 4.3.1 that this greedy approach yields performances comparable to an exhaustive search when modeling small numbers of layers and sprites.

Occlusion modeling. Occlusion is modeled explicitly in our composition process defined in Equation (4.2) since $\mathbf{o}_1, \dots, \mathbf{o}_L$ are ranked by depth. However, we experimentally observed that layers learn to specialize to different regions in the image. This seems to correspond to a local minimum of the loss function, and the model does not manage to reorder the layers to predict the correct occlusion. Therefore, we relax the model and predict an occlusion matrix $\delta = \psi(\mathbf{x}) \in [0,1]^{L \times L}$ instead of keeping it fixed. More precisely, for each image $\mathbf{x}$ we predict $\frac{1}{2}L(L-1)$ values using a neural network followed by a sigmoid function. These values are then reshaped to a lower triangular $L \times L$ matrix with zero diagonal, and the upper part is computed by symmetry such that: $\forall i < j,\ \delta_{ij} = 1 - \delta_{ji}$. While such predicted occlusion matrix

is not binary and does not directly translate into a layer reordering, it still allows us to compute a composite image using Equation (4.2) and the masks associated to each object. Note that such a matrix could model more complex occlusion relationships such as non-transitive ones. At inference, we simply replace δ_{ij} by $\delta_{ij} > 0.5$ to obtain binary occlusion relationships. We also tried computing the closest matrix corresponding to a true layer reordering and obtained similar results. Note that when we use a background model, its occlusion relationships are fixed, *i.e.* $\forall j > 1, \delta_{j1} = 1$.

Training details. Two elements of our training strategy are crucial to the success of learning. First, following Monnier et al. [2020] we adopt a curriculum learning of the transformations, starting by the simplest ones. Second, inspired by Tieleman [2014] and Kosiorek et al. [2019], we inject uniform noise in the masks in such a way that masks are encouraged to be binary (see Appendix B.3 for details). This allows us to resolve the ambiguity that would otherwise exist between the color and alpha channels and obtain clear masks. We provide additional details about networks' architecture, computational cost, transformations used and implementation in Appendix B.3.

4.3 Experiments

Assessing the quality of an object-based image decomposition model is ambiguous and difficult, and downstream applications on synthetic multi-object benchmarks such as Kabra et al. [2019] are typically used as evaluations. Thus, recent approaches (*e.g.*, Burgess et al. [2019]; Greff et al. [2019]; Engelcke et al. [2020]; Locatello et al. [2020]) first evaluate their ability to infer spatial arrangements of objects through quantitative performances for object instance discovery. The knowledge of the learned concept of object is then evaluated qualitatively through convincing object-centric image manipulation [Burgess et al., 2019; Greff et al., 2019], occluded region reconstructions [Burgess et al., 2019; Locatello et al., 2020] or realistic generative sampling [Engelcke et al., 2020]. None of these approaches explicitly model categories for objects and, to the best of our knowledge, their applicability is limited to synthetic imagery only.

In this section, we first evaluate and analyse our model on the standard multi-object synthetic benchmarks (Section 4.3.1). Then, we demonstrate that our approach can be applied to real images (Section 4.3.2). We use the 2-layer version of our model to perform clustering (Section 4.3.2), cosegmentation (Section 4.3.2), as well as qualitative object discovery from unfiltered web image collections (Section 4.3.2).

4.3.1 Multi-object synthetic benchmarks

Datasets and evaluation. Tetrominoes [Greff et al., 2019] is a 60k dataset generated by placing three Tetrominoes without overlap in a 35×35 image. There is a total of 19 different Tetrominoes (counting discrete rotations). Multi-dSprites [Kabra et al., 2019] contains 60k images of size 64×64 with 2 to 5 objects sampled from a set of 3 different shapes: ellipse, heart, square. CLEVR6 [Johnson et al., 2017; Greff et al., 2019] contains 34,963 synthetically generated images of size 128×128. Each image is composed of a variable number of objects (from 3 to 6), each sampled from a set of 6 categories - 3 different shapes (sphere, cylinder, cube) and 2 materials (rubber or metal) - and randomly rendered. We thus train our method using one sprite per object category and as many layers as the maximum number of objects per image, with a background layer when necessary. Following standard practices [Greff et al., 2019; Locatello et al., 2020], we evaluate object instance segmentation on 320 images by averaging over all images the Adjusted Ranked Index (ARI) computed using ground-truth foreground pixels only (ARI-FG in our tables). Note that because background pixels are filtered, ARI-FG strongly favors methods like Greff et al. [2019]; Locatello et al. [2020] which oversegment objects or do not discriminate foreground from background. To limit the penalization of our model which explicitly models background, we reassign predicted background pixels to the closest object layers before computing this metric. However, we argue that foreground/background separation is crucial for any downstream applications and also advocate the use of a true ARI metric computed on all pixels (including background) which we include in our results. In addition, we think that the knowledge of object category should be evaluated and include quantitative results for unsupervised semantic segmentation in Appendix B.1.

Results. Our results are compared quantitatively to state-of-the-art approaches in Table 4.1. On Multi-dSprites, an outlier run out of 5 was automatically filtered based on its high reconstruction loss compared to the others. Our method obtains results on par with the best competing methods across all benchmarks. While our approach is more successful on benchmarks depicting 2D scenes, it still provides good results on CLEVR6 where images include 3D effects. We provide our results using the real ARI metric which we believe to be more interesting as it is not biased towards oversegmenting methods. While this measure is not reported by competing methods, a CLEVR6 decomposition example shown in official IODINE implementation[1] gives a perfect 100% ARI-FG score but reaches 20% in terms of ARI.

Compared to all competing methods, our approach explicitly models categories for objects. In particular, it is able to learn prototypical images that can be associated to each object category.

[1] https://github.com/deepmind/deepmind-research/blob/master/iodine

Method	Metric	Tetrominoes	Multi-dSprites	CLEVR6
MONet [Burgess et al., 2019]	ARI-FG	-	90.4 ± 0.8	96.2 ± 0.6
IODINE [Greff et al., 2019]	ARI-FG	99.2 ± 0.4	76.7 ± 5.6	$\mathbf{98.8 \pm 0.0}$
Slot Att. [Locatello et al., 2020]	ARI-FG	$99.5^{\triangle} \pm 0.2$	91.3 ± 0.3	$\mathbf{98.8 \pm 0.3}$
Ours	ARI-FG	$\mathbf{99.6 \pm 0.2}$	$\mathbf{92.5^{\triangle} \pm 0.3}$	97.2 ± 0.2
Ours	ARI	99.8 ± 0.1	$95.1^{\triangle} \pm 0.1$	90.7 ± 0.1

Table 4.1: **Multi-object instance discovery.** Following standard practices, we report ARI-FG (ARI on foreground pixels only) averaged over 5 runs. We also report our results with the real ARI, a metric we advocate for future comparisons. We mark results ($\triangle$) where one outlier run is filtered out.

Dataset	Model	ARI-FG	ARI
Multi-dSprites2	Full	$\mathbf{95.5 \pm 2.1}$	95.2 ± 1.9
	w/o greedy algo.	94.4 ± 2.7	$\mathbf{95.9 \pm 0.3}$
Multi-dSprites	Full	$\mathbf{91.5 \pm 2.2}$	$\mathbf{95.0 \pm 0.3}$
	w/o occ. prediction	85.7 ± 2.2	94.2 ± 0.2
Tetrominoes	Full	$\mathbf{99.6 \pm 0.2}$	$\mathbf{99.8 \pm 0.1}$
	w/o shared transfo.	95.3 ± 3.7	82.6 ± 12.2

Table 4.2: **Ablation study.** Results are averaged over 5 runs.

The sprites discovered from CLEVR6 and Tetrominoes are shown in Figure 4.1. Note how learned sprites on Tetrominoes are sharp and how we can identify material in CLEVR6 by learning two different sprites for each shape.

In Figure 4.3, we show some qualitative results obtained on the three benchmarks. Given sample images, we show from left to right the final reconstruction, semantic segmentation (evaluated quantitatively in Appendix B.1) where each color corresponds to a different sprite, instance segmentation, and the first four layers of the image decomposition. Note how we manage to successfully predict occlusions, model variable number of objects, separate the different instances, as well as identify the object categories and their spatial extents. More random decomposition results are shown in Appendix B.4 and on our webpage.

Compared to other approaches which typically need a form of supervision to interpret learned representations as object visual variations, our method has the advantage to give a direct access to the object instance parameters, enabling us to directly manipulate them in images. In Figure 4.4, we show different object-centric image manipulations such as objects swapping as well as color, position and scale variations. Note that we are also able to render out of distribution instances, like the pink sphere or the gigantic cylinder.

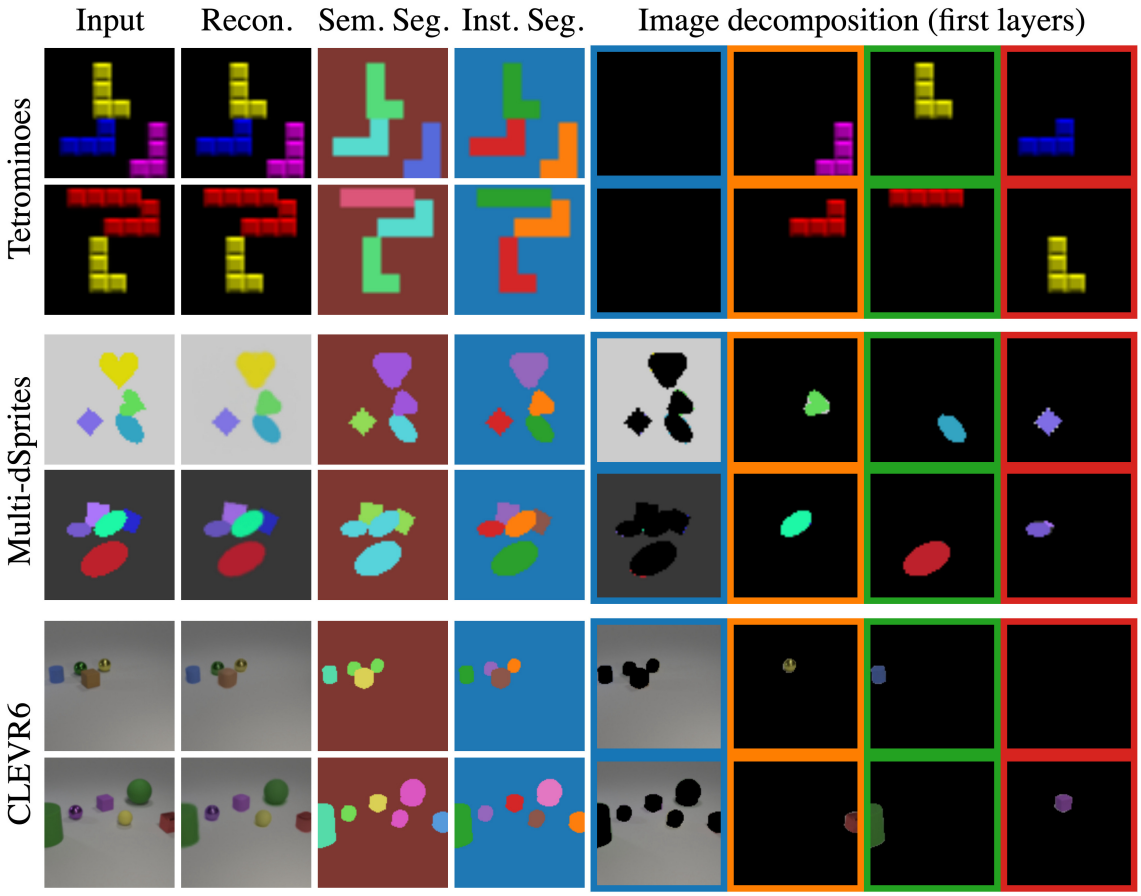


Figure 4.3: **Multi-object discovery.** From left to right, we show inputs, reconstructions, semantic (each color corresponds to a different sprite) and instance segmentations, and first decomposition layers colored w.r.t. their instance mask.

Ablation study. We analyze the main components of our model in Table 4.2. For computational reasons, we evaluate our greedy algorithm on Multi-dSprites2 - the subset of Multi-dSprites containing only 2 objects - and show comparable performances to an exhaustive search over all combinations. Occlusion prediction is evaluated on Multi-dSprites which contains many occlusions. Because our model with fixed occlusion does not manage to reorder the layers, performances are significantly better when occlusion is learned. Finally, we compare results obtained on Tetrominoes when modeling sprite-specific transformations only, without shared ones, and show a clear gap between the two settings. We provide analyses on the effects of K and λ in Appendix B.2.

Limitations. Our optimization model can be stuck in local minima. A typical failure mode on Multi-dSprites can be seen in the reconstructions in Figure 4.3 where a triangular shape is learned instead of the heart. This sprite can be aligned to a target heart shape using three

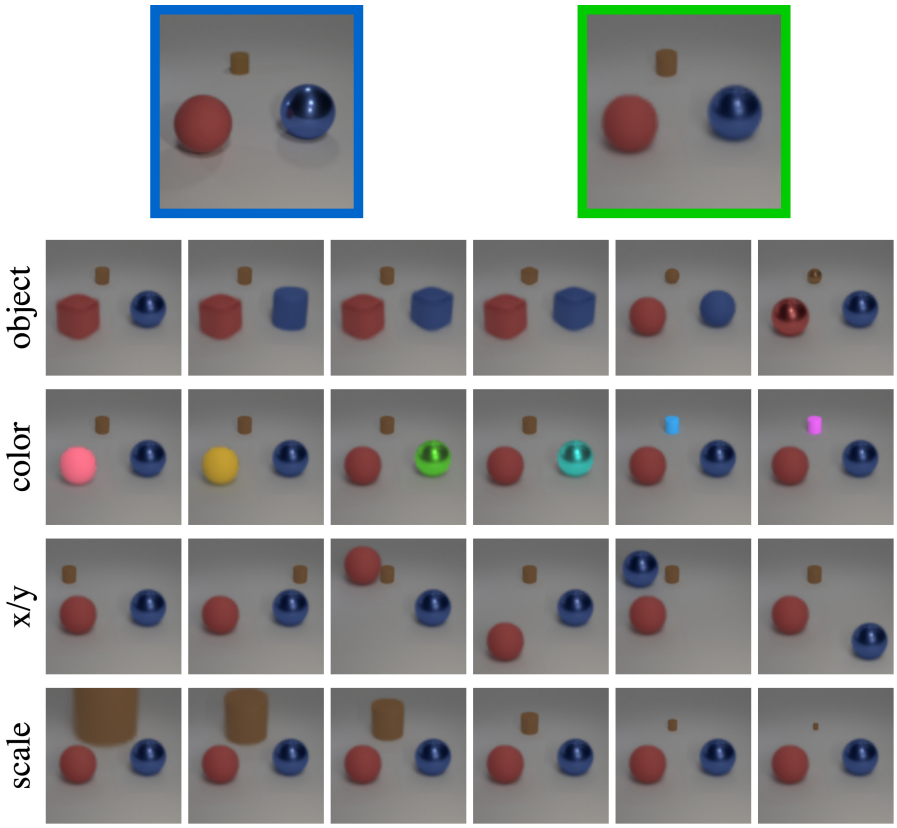


Figure 4.4: **Object-centric image manipulation**. Given a query image (top left) from CLEVR6 [Johnson et al., 2017], we show the closest reconstruction (top right) and several image manipulations (next four rows). From top to bottom, we respectively use different sprites, change the objects color, vary their positions and modify the scale.

different equivalent rotations, and our model does not manage to converge to a consistent one. This problem could be overcome by either modeling more sprites, manually computing reconstructions with different discrete rotations, or guiding transformation predictions with supervised sprite transformations.

4.3.2 Real image benchmarks

Clustering

Datasets. We evaluate our model on two real image clustering datasets using 2 layers, one for the background and one for the foreground object. SVHN [Netzer et al., 2011] is a standard clustering dataset composed of digits extracted from house numbers cropped from Google Street View images. Following standard practices [Hu et al., 2017; Kosiorek et al., 2019; Monnier et al., 2020], we evaluate on the labeled subset (99,289 images), but also use 530k unlabeled extra samples for training. We also report results on traffic sign images using a

Method	Runs	GTSRB-8	SVHN
Clustering on learned features			
[Häusser et al., 2018]	20	-	$38.6^{\triangledown}$
[Kosiorek et al., 2019]	5	-	$55.3^{\ddagger}$
[Hu et al., 2017]	12	$26.9^{\triangledown\star}$	$57.3^{\triangledown\dagger}$
[Van Gansbeke et al., 2020]	5	$\mathbf{90.4}^{\triangledown\star}$	$54.2^{\triangledown\star}$
Clustering on pixel values			
[Monnier et al., 2020]	10	$54.3^{\star}$	57.4
Ours	10	89.4	**63.1**

(a) Clustering results

Method	Accuracy (%)
[Rubio et al., 2012]	74.9
[Joulin et al., 2010]	80.1
[Lattari et al., 2015]	84.6
[Chang et al., 2011]	86.4
[Yu et al., 2014]	87.6
Ours	**87.9**

(b) Cosegmentation results

Table 4.3: **Additional comparisons. (a)** We report average clustering accuracy. We mark methods we ran ourselves with official implementations ($\star$), use data augmentation ($\triangledown$) or ad-hoc representations ($\dagger$ for GIST, $\ddagger$ for Sobel filters). **(b)** We report segmentation accuracy on the Weizmann Horse database.

balanced subset of the GTSRB dataset [Stallkamp et al., 2012] which we call GTSRB-8. We selected classes with 1000 to 1500 instances in the training split, yielding 8 classes and 10,650 images which we resize to 28×28.

Results. We compare our model to state-of-the-art methods in Table 4.3a using global clustering accuracy, where the cluster-to-class mapping is computed using the Hungarian algorithm [Kuhn and Yaw, 1955]. We train our 2-layer model with as many sprites as classes and a single background prototype. On both benchmarks, our approach provides competitive results. In particular, we improve state of the art on the standard SVHN benchmark by an absolute 5% increase.

Similar to DTI-Clustering, our method performs clustering in pixel-space exclusively and has the advantage of providing interpretable results. Figure 4.5 shows learned sprites on the GTSRB-8 and SVHN datasets and compares them to prototypes learned with DTI-Clustering. Note in particular the sharpness of the discovered GTSRB-8 sprites.

Cosegmentation

Dataset. We use the Weizmann Horse database [Borenstein and Ullman, 2004] to evaluate quantitatively the quality of our masks. It is composed of 327 side-view horse images resized to 128×128. Although relatively simple compared to more recent cosegmentation datasets, it presents significant challenges compared to previous synthetic benchmarks because of the diversity of both horses and backgrounds. The dataset was mainly used by classical (non-deep) methods which were trained and evaluated on 30 images for computational reasons while we train and evaluate on the full set.

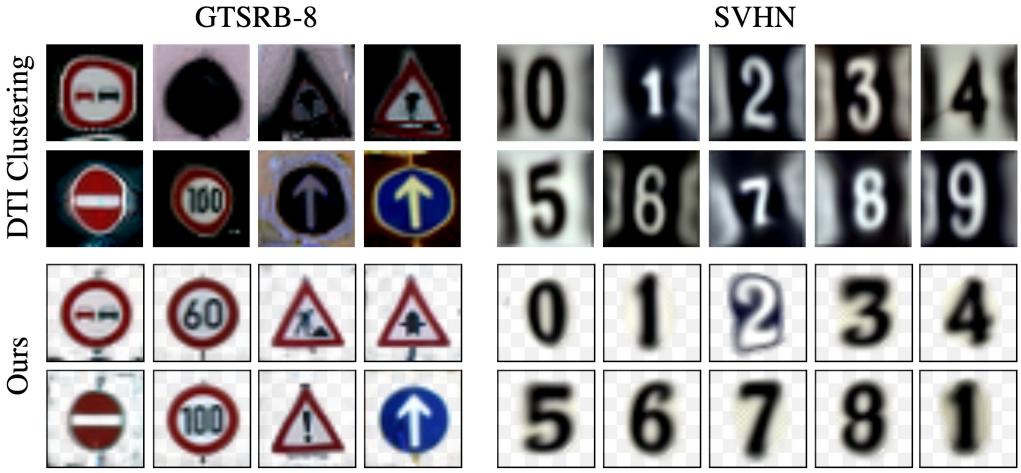


Figure 4.5: **Qualitative clustering results.** We compare prototypes learned using DTI-Clustering and our discovered sprites on GTSRB-8 (left) and SVHN (right).

Results. We compare our 2-layer approach with a single sprite to classical cosegmentation methods in Table 4.3b and report segmentation accuracy - mean % of pixels correctly classified as foreground or background - averaged over 5 runs. Our results compare favorably to these classical approaches. Although more recent approaches could outperform our method on this dataset, we argue that obtaining performances on par with such competing methods is already a strong result for our layered image decomposition model.

We present in Figure 4.6 some visual results of our approach. First, the discovered sprite clearly depicts a horse shape and its masks is sharp and accurate. Learning such an interpretable sprite from this real images collection is already interesting and validates that our sprite-based modeling generalizes to real images. Second, although the transformations modeled are quite simple (a combination of color and spatial transformations), we demonstrate good reconstructions and decompositions, yielding accurate foreground extractions.

Unfiltered web image collections

We demonstrate our approach's robustness by visualizing sprites discovered from web image collections. We use the same Instagram collections introduced in Monnier et al. [2020], where each collection is associated to a specific hashtag and contains around 15k images resized and center cropped to 128×128. We apply our model with 40 sprites and a background.

Figure 4.7 shows the 8 best qualitative sprites discovered from Instagram collections associated to #santaphoto and #weddingkiss. Even in this case where images are mostly noise, our approach manages to discover meaningful sprites and segmentations with clear visual variations. For example, we can distinguish standing santas from seating ones, as well as the

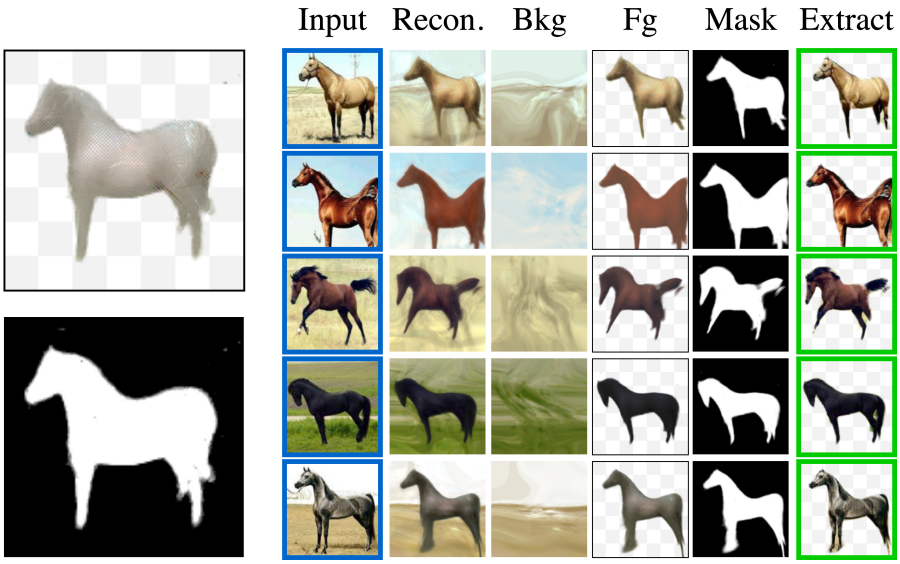


Figure 4.6: **Qualitative cosegmentation results.** Sprite and mask (left) learned from Weizmann Horse [Borenstein and Ullman, 2004] and some result examples (right) giving for each input, its reconstruction, the layered composition and extracted foreground.

ones alone or surrounded by children. We additionally show examples of reconstructions and image compositions for some of the 8 sprites shown for #santaphoto.

4.4 Conclusion

We have introduced a new unsupervised model which jointly learns sprites, transformations and occlusions to decompose images into object layers. Beyond standard multi-object synthetic benchmarks, we have demonstrated that our model leads to actual improvements for real image clustering with a 5% boost over the state of the art on SVHN and can provide good segmentation results. We even show it is robust enough to provide meaningful results on unfiltered web image collections. Although our object modeling involves unique prototypical images and small sets of transformations limiting their instances diversity, we argue that accounting for such diversity while maintaining a category-based decomposition model is extremely challenging, and our approach is the first to explore this direction as far as we know.

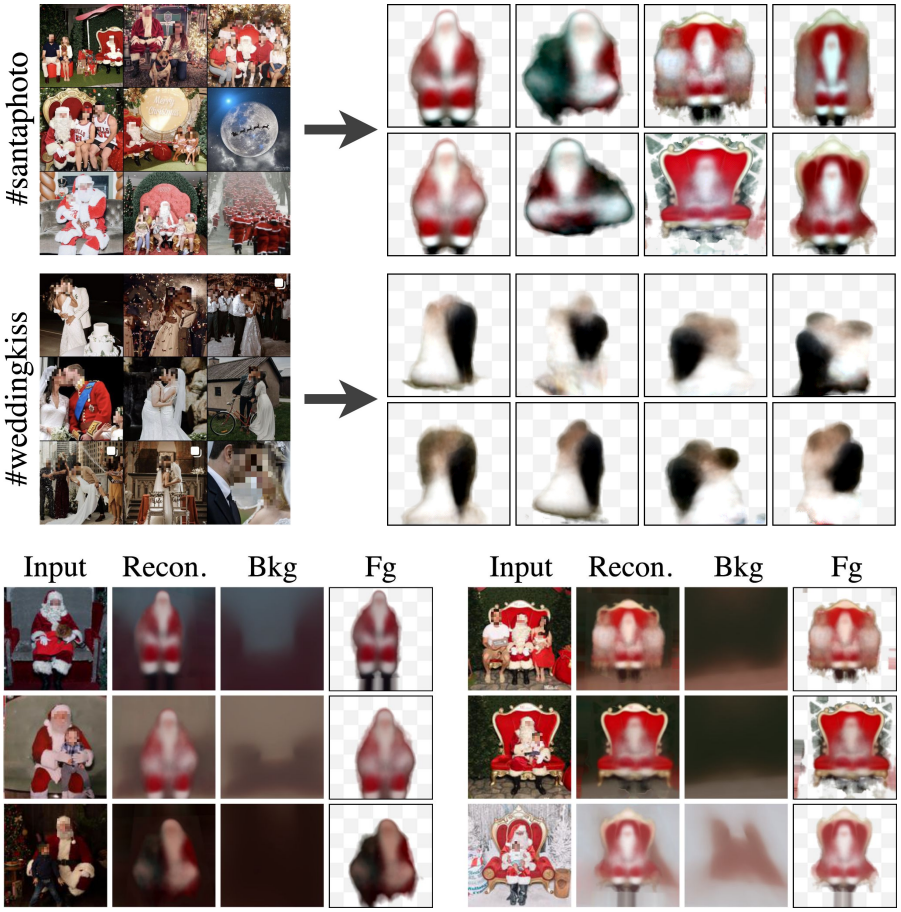


Figure 4.7: **Web image results.** We show the 8 best qualitative sprites among 40 discovered from Instagram collections (top) as well as decomposition results for samples represented by one of the sprites shown for #santaphoto (bottom).

Chapter 5

Single-View Reconstruction Without Supervision

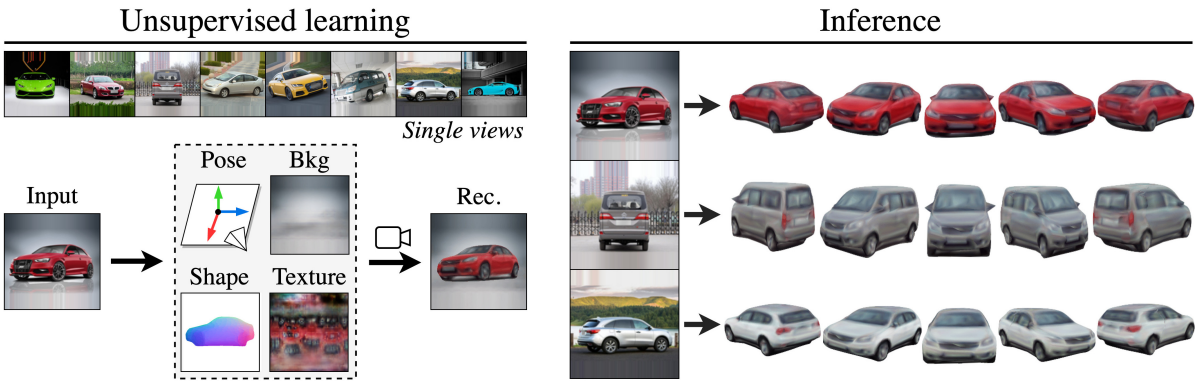


Figure 5.1: **Single-View Reconstruction Without Supervision. (left)** Given a collection of single-view images from an object category, we learn without additional supervision an autoencoder that explicitly generates shape, texture, pose and background. **(right)** At inference time, our approach reconstructs high-quality textured meshes from raw single-view images.

5.1 Introduction

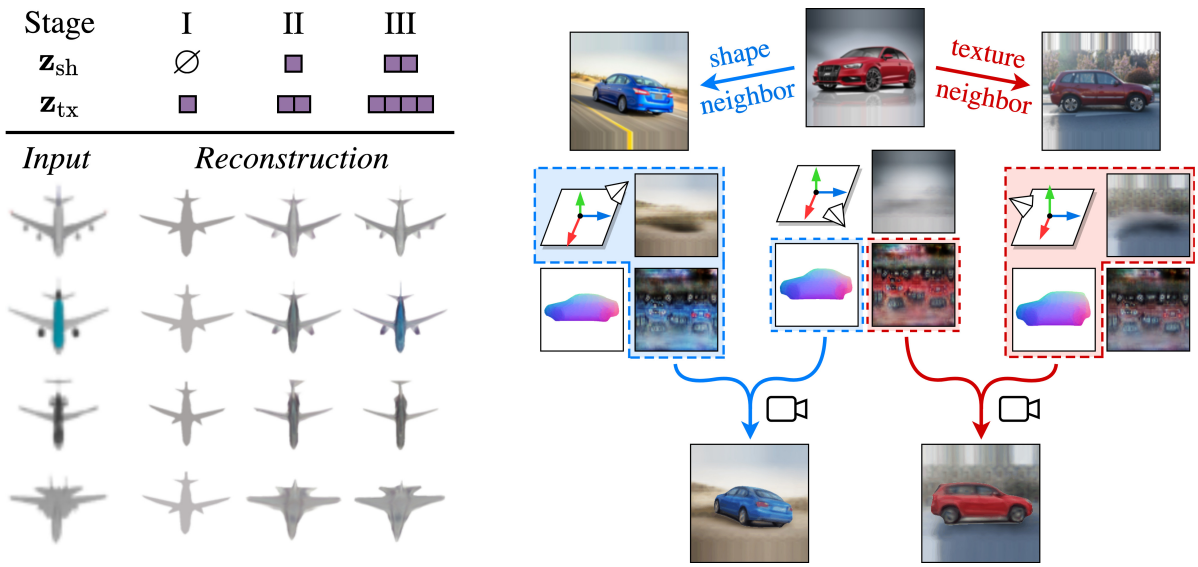
One of the most magical human perceptual abilities is being able to see the 3D world behind a 2D image – a mathematically impossible task! Indeed, the ancient Greeks were so incredulous at the possibility that humans could be "hallucinating" the third dimension, that they proposed the utterly implausible Emission Theory of Vision [Finger, 1994] (eye emitting light to "sense" the world) to explain it to themselves. In the history of computer vision, single-view reconstruction (SVR) has had an almost cult status as one of the holy grail problems [Hoiem et al., 2005,

2008; Saxena et al., 2009]. Recent advancements in deep learning methods have dramatically improved results in this area [Choy et al., 2016; Mescheder et al., 2019]. However, the best methods still require costly supervision at training time, such as multiple views [Liu et al., 2019; Niemeyer et al., 2020]. Despite efforts to remove such requirements, the works with the least supervision still rely on two signals limiting their applicability: (i) silhouettes and (ii) strong assumptions such as symmetries [Kanazawa et al., 2018; Hu et al., 2021], known template shapes [Goel et al., 2020; Tulsiani et al., 2020], or the absence of background [Wu et al., 2020]. Although crucial to achieve reasonable results, priors like silhouettes and symmetry can also harm the reconstruction quality: silhouette annotations are often coarse [Chen et al., 2019b] and small symmetry prediction errors can yield unrealistic reconstructions [Wu et al., 2020; Goel et al., 2020].

In this paper, we propose the most unsupervised approach to single-view reconstruction to date, which we demonstrate to be competitive for diverse datasets. Table 5.1 summarizes the differences between our approach and representative prior works. More precisely, we learn in an analysis-by-synthesis fashion a network that predicts for each input image: 1) a 3D shape parametrized as a deformation of an ellipsoid, 2) a texture map, 3) a camera viewpoint, and 4) a background image (Figure 5.1). Our main insight to remove the supervision and assumptions required by other methods is to leverage the consistency across different instances. First, we design a training procedure, *progressive conditioning*, which encourages the model to share elements between images by strongly constraining the variability of shape, texture and background at the beginning of training and progressively allowing for more diversity

Method	Supervision	Data	Output
[Wang et al., 2018], [Groueix et al., 2018], [Mescheder et al., 2019]	**3D**	ShapeNet	3D
[Yan et al., 2016], [Kato et al., 2018]	**MV**, **CK**, **S**	ShapeNet	3D
[Tulsiani et al., 2017b], [Liu et al., 2019], [Niemeyer et al., 2020]	**MV**, **CK**, **S**	ShapeNet	3D, **T**
[Zhang et al., 2021b]	**MV**, **CK**, **S**	Bird, Car, Horse	3D, **T**
[Insafutdinov and Dosovitskiy, 2018], [Tulsiani et al., 2018]	**MV**, **S**	ShapeNet	3D, **P**
[Vicente et al., 2014], [Kar et al., 2015], [Tulsiani et al., 2017b]	**CK**, **S**	Pascal3D	3D
[Kanazawa et al., 2018]	**CK**, **S**, **A**(†)	Bird, Car, Plane	3D, **T**
[Lin et al., 2020a], [Duggal and Pathak, 2022]	**CK**, **S**	ShapeNet, Bird, Car, Plane	3D, **T**
[Henderson et al., 2020]	**CK**, **A**(†)	ShapeNet, Bird, Car	3D, **T**
[Goel et al., 2020], [Tulsiani et al., 2020]	**S**, **A**(♢, †)	Animal, Car, Moto	3D, **T**, **P**
[Li et al., 2020]	**S**, **A**(↔, †)	Animal, Car, Moto	3D, **T**, **P**
[Wu et al., 2021]	**S**, **A**(‡)	Vase	3D, **T**, **P**
[Hu et al., 2021]	**S**, **A**(†)	ShapeNet, Animal, Moto	3D, **T**, **P**
[Wu et al., 2020]	**A**(⊠, ≺, †)	Face	**D**, **T**, **P**
[Henderson and Ferrari, 2019]	**A**(⊠, ∅)	ShapeNet	3D, **P**
Ours	None	ShapeNet, Animal, Car, Moto	3D, **T**, **P**

Table 5.1: **Comparison with selected works.** For each method, we outline the supervision and priors used (**3D**, **M**ulti-**V**iews, **C**amera or **K**eypoints, **S**ilhouettes, **A**ssumptions like ♢ template shape, † symmetry, ‡ solid of revolution, ↔ semantic consistency, ⊠ no/limited background, ≺ frontal view, ∅ no texture), which data it has been applied to and the model output (3D, **T**exture, **P**ose, **D**epth).



(a) Progressive conditioning

(b) Neighbor reconstruction

Figure 5.2: **Leveraging cross-instance consistency. (a)** Progressive conditioning amounts to gradually increasing, in a multi-stage fashion, the size of the conditioning latent spaces, here associated to shape $\mathbf{z}_{\text{sh}}$ and texture $\mathbf{z}_{\text{tx}}$. **(b)** We explicitly share the shape and texture models across neighboring instances by swapping their characteristics and applying a loss to associated neighbor reconstructions.

(Figure 5.2a). Second, we introduce a *neighbor reconstruction* loss, which explicitly enforces neighboring instances from different viewpoints to share the same shape or texture model (Figure 5.2b). Note that these simple yet effective techniques are data-driven and not specific to any dataset. Our only remaining assumption is the knowledge of the semantic class of the depicted object.

We also provide two technical insights that we found critical to learn our model without viewpoint and silhouette annotations: (i) a differentiable rendering formulation inspired by layered image models [Jojic and Frey, 2001; Monnier et al., 2021] which we found to perform better than the classical SoftRasterizer [Liu et al., 2019], and (ii) a new optimization strategy which alternates between learning a set of pose candidates with associated probabilities and learning all other components using the most likely candidate.

We validate our approach on the standard ShapeNet [Chang et al., 2015] benchmark, real image SVR benchmarks (Pascal3D+ Car [Xiang et al., 2014], CUB [Welinder et al., 2010]) as well as more complex real-world datasets (CompCars [Yang et al., 2015], LSUN Motorbike and Horse [Yu et al., 2016]). In all scenarios, we demonstrate results competitive with the best supervised methods.

Summary. We present UNICORN, a framework leveraging **UN**supervised cross-**I**nstance **CO**nsistency for 3D **R**econstructio**N**. Our main contributions are: 1) the most unsupervised SVR system to date, demonstrating state-of-the-art textured 3D reconstructions for both generic shapes and real images, and not requiring supervision or restrictive assumptions beyond a categorical image collection; 2) two data-driven techniques to enforce cross-instance consistency, namely progressive conditioning and neighbor reconstruction. Code and video results are available at: www.tmonnier.com/UNICORN.

5.2 Related work

We discuss works exploring cross-instance consistency and curriculum learning techniques, to which our progressive conditioning is related.

Cross-instance consistency. Although all methods learned on categorical image collections implicitly leverage the consistency across instances, few recent works explicitly explore such a signal. Inspired by Kulkarni et al. [2019], the SVR system of Hu et al. [2021] is learned by enforcing consistency between the interpolated 3D attributes of two instances and attributes predicted for the associated reconstruction. Yao et al. [2021] discovers 3D parts using the inconsistency of parts across instances. Closer to our approach, Navaneet et al. [2020] introduces a loss enforcing cross-silhouette consistency. Yet it differs from our work in two ways: (i) the loss operates on silhouettes, whereas our loss is adapted to image reconstruction by modeling background and separating two terms related to shape and texture, and (ii) the loss is used as a refinement on top of two cycle consistency losses for poses and 3D reconstructions, whereas we demonstrate results without additional self-supervised losses.

Curriculum learning. The idea of learning networks by "starting small" dates back to Elman [1993] where two curriculum learning schemes are studied: (i) increasing the difficulty of samples, and (ii) increasing the model complexity. We respectively coin them *curriculum sampling* and *curriculum modeling* for differentiation. Known to drastically improve the convergence speed [Bengio et al., 2009], curriculum sampling is widely adopted across various applications [Schroff et al., 2015b; Bengio et al., 2015; Ilg et al., 2017]. On the contrary, curriculum modeling is typically less studied although crucial to various methods. For example, Wang et al. [2018] perform SVR in a coarse-to-fine manner by increasing the number of mesh vertices, and Monnier et al. [2020] cluster images by aligning them with transformations that increase in complexity. We propose a new form of curriculum modeling dubbed *progressive conditioning* which enables us to avoid bad minima.

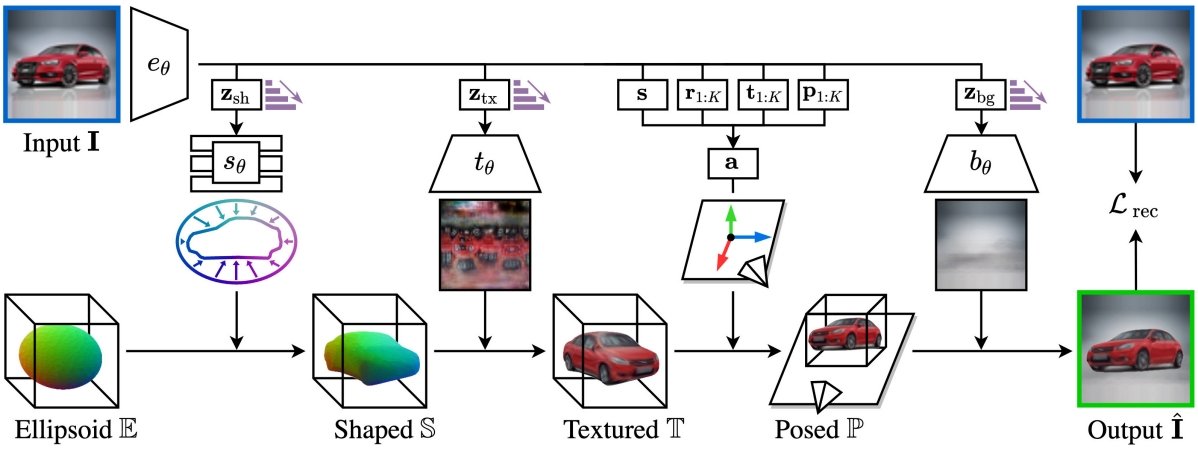


Figure 5.3: **Structured autoencoding.** Given an input, we predict parameters that are decoded into 4 factors (shape, texture, pose, background) and composed to generate the output. Progressive conditioning is represented with .

5.3 Approach

Our goal is to learn a neural network that reconstructs a textured 3D object from a single input image. We assume we have access to a raw collection of images depicting objects from the same category, without any further annotation. We propose to learn in an analysis-by-synthesis fashion by autoencoding images in a structured way (Figure 5.3). We first introduce our structured autoencoder (Section 5.3.1). We then present how we learn models consistent across instances (Section 5.3.2). Finally, we discuss one more technical contribution necessary to our system: an alternate optimization strategy for joint 3D and pose estimation (Section 5.3.3).

Notations. We use bold lowercase for vectors (e.g., $\mathbf{a}$), bold uppercase for images (e.g., $\mathbf{A}$), double-struck uppercase for meshes (e.g., $\mathbb{A}$), calligraphic uppercase for the main modules of our system (e.g., $\mathcal{A}$), lowercase indexed with generic parameters θ for networks (e.g., a_θ), and write $a_{1:N}$ the ordered set $\{a_1, \ldots, a_n\}$.

5.3.1 Structured autoencoding

Overview. Our approach can be seen as a structured autoencoder: it takes an image as input, computes parameters with an encoder, and decodes them into explicit and interpretable factors that are composed to generate an image. We model images as the rendering of textured meshes on top of background images. For a given image $\mathbf{I}$, our model thus predicts a shape, a texture, a pose and a background which are composed to get the reconstruction $\hat{\mathbf{I}}$, as shown in Figure 5.3. More specifically, the image $\mathbf{I}$ is fed to convolutional encoder networks e_θ

which output parameters $e_\theta(\mathbf{I}) = \{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}, \mathbf{a}, \mathbf{z}_{\text{bg}}\}$ used for the decoding part. $\mathbf{a}$ is a 9D vector including the object pose, while the dimension of the latent codes $\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}$ and $\mathbf{z}_{\text{bg}}$ will vary during training (see Section 5.3.2). In the following, we describe the decoding modules using these parameters to build the final image by generating a shape, adding texture, positioning it and rendering it over a background.

Shape deformation. We follow Tulsiani et al. [2020] and use the parametrization of AtlasNet [Groueix et al., 2018] where different shapes are represented as deformation fields applied to the unit sphere. We apply the deformation to an icosphere slightly stretched into an ellipsoid mesh $\mathbb{E}$ using a fixed anisotropic scaling. More specifically, given a 3D vertex $\mathbf{x}$ of the ellipsoid, our shape deformation module $\mathcal{S}_{\mathbf{z}_{\text{sh}}}$ is defined by $\mathcal{S}_{\mathbf{z}_{\text{sh}}}(\mathbf{x}) = \mathbf{x} + s_\theta(\mathbf{x}, \mathbf{z}_{\text{sh}})$, where s_θ is a Multi-Layer Perceptron taking as input the concatenation of a 3D point $\mathbf{x}$ and a shape code $\mathbf{z}_{\text{sh}}$. Applying this displacement to all the ellipsoid vertices enables us to generate a shaped mesh $\mathbb{S} = \mathcal{S}_{\mathbf{z}_{\text{sh}}}(\mathbb{E})$. We found that using an ellipsoid instead of a raw icosphere was very effective in encouraging the learning of objects aligned w.r.t. the canonical axes. Learning surface deformations is often preferred to vertex-wise displacements as it enables mapping surfaces, and thus meshes, at any resolution. For us, the mesh resolution is kept fixed and such a representation is a way to regularize the deformations.

Texturing. Following the idea of CMR [Kanazawa et al., 2018], we model textures as an image UV-mapped onto the mesh through the reference ellipsoid. More specifically, given a texture code $\mathbf{z}_{\text{tx}}$, a convolutional network t_θ is used to produce an image $t_\theta(\mathbf{z}_{\text{tx}})$, which is UV-mapped onto the sphere using spherical coordinates to associate a 2D point to every vertex of the ellipsoid, and thus to each vertex of the shaped mesh. We write $\mathcal{T}_{\mathbf{z}_{\text{tx}}}$ this module generating a textured mesh $\mathbb{T} = \mathcal{T}_{\mathbf{z}_{\text{tx}}}(\mathbb{S})$.

Affine transformation. To render the textured mesh $\mathbb{T}$, we define its position w.r.t. the camera. In addition, we found it beneficial to explicitly model an anisotropic scaling of the objects. Because predicting poses is difficult, we predict K poses candidates, defined by rotations $\mathbf{r}_{1:K}$ and translations $\mathbf{t}_{1:K}$, and associated probabilities $\mathbf{p}_{1:K}$. This involves learning challenges we tackle with a specific optimization procedure described in Section 5.3.3. At inference, we select the pose with highest probability. We combine the scaling and the most likely 6D pose in a single affine transformation module $\mathcal{A}_{\mathbf{a}}$. More precisely, $\mathcal{A}_{\mathbf{a}}$ is parametrized by $\mathbf{a} = \{\mathbf{s}, \mathbf{r}, \mathbf{t}\}$, where $\mathbf{s}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^3$ respectively correspond to the three scales of an anisotropic scaling, the three Euler angles of a rotation and the three coordinates of a translation. A 3D point $\mathbf{x}$ on the mesh is then transformed by $\mathcal{A}_{\mathbf{a}}(\mathbf{x}) = \text{rot}(\mathbf{r})\text{diag}(\mathbf{s})\mathbf{x} + \mathbf{t}$ where $\text{rot}(\mathbf{r})$ is the rotation matrix

(a) Degenerate background (b) Degenerate 3D object

Figure 5.4: **Degenerate solutions.** An SVR system learned by raw image autoencoding is prone to degenerate solutions through **(a)** the background or **(b)** the 3D object model. We alleviate the issue with cross-instance consistency.

associated to $\mathbf{r}$ and $\text{diag}(\mathbf{s})$ is the diagonal matrix associated to $\mathbf{s}$. Our module is applied to all points of the textured mesh $\mathbb{T}$ resulting in a posed mesh $\mathbb{P} = \mathcal{A}_{\mathbf{a}}(\mathbb{T})$.

Rendering with background. The final step of our process is to render the mesh over a background image. The background image is generated from a background code $\mathbf{z}_{\text{bg}}$ by a convolutional network b_θ. A differentiable module $\mathcal{B}_{\mathbf{z}_{\text{bg}}}$ renders the posed mesh $\mathbb{P}$ over this background image $b_\theta(\mathbf{z}_{\text{bg}})$ resulting in a reconstructed image $\hat{\mathbf{I}} = \mathcal{B}_{\mathbf{z}_{\text{bg}}}(\mathbb{P})$. We perform rendering through soft rasterization of the mesh. Because we observed divergence results when learning geometry from raw photometric comparison with the standard SoftRasterizer [Liu et al., 2019; Ravi et al., 2020], we propose two key changes: a layered aggregation of the projected faces and an alternative occupancy function. We present our custom differentiable rendering function in Appendix C.1.

5.3.2 Unsupervised learning with cross-instance consistency

We propose to learn our structured autoencoder without any supervision, by synthesizing 2D images and minimizing a reconstruction loss. Due to the unconstrained nature of the problem, such an approach typically yields degenerate solutions (Figure 5.4a and Figure 5.4b). While previous works leverage silhouettes and dataset-specific priors to mitigate this issue, we instead propose two unsupervised data-driven techniques, namely *progressive conditioning* (a training strategy) and *neighbor reconstruction* (a training loss). We thus optimize the shape, texture and background by minimizing for each image $\mathbf{I}$ reconstructed as $\hat{\mathbf{I}}$:

$$\mathcal{L}_{\text{3D}} = \mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_{\text{nbr}}\mathcal{L}_{\text{nbr}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}, \tag{5.1}$$

where λ_{nbr} and λ_{reg} are scalar hyperparameters, and $\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{nbr}}$ and $\mathcal{L}_{\text{reg}}$ are respectively the reconstruction, neighbor reconstruction, and regularization losses, described below. In all

experiments, we use $\lambda_{\text{nbr}} = 1$ and $\lambda_{\text{reg}} = 0.01$. Note that we optimize pose prediction using a slightly different loss in an alternate optimization scheme described in Section 5.3.3.

Reconstruction and regularization losses. Our reconstruction loss has two terms, a pixel-wise squared L_2 loss $\mathcal{L}_{\text{pix}}$ and a perceptual loss [Zhang et al., 2018] $\mathcal{L}_{\text{perc}}$ defined as an L_2 loss on the `relu3_3` layer of a pre-trained VGG16 [Simonyan and Zisserman, 2015], similar to Wu et al. [2020]. While pixel-wise losses are common for autoencoders, we found it crucial to add a perceptual loss to learn textures that are discriminative for the pose estimation. Our full reconstruction loss can be written $\mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}) = \mathcal{L}_{\text{pix}}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}}(\mathbf{I}, \hat{\mathbf{I}})$ and we use $\lambda_{\text{perc}} = 10$ in all experiments. While our deformation-based surface parametrization naturally regularizes the shape, we sometimes observe bad minima where the surface has folds. Following prior works [Liu et al., 2019; Chen et al., 2019b; Goel et al., 2020; Zhang et al., 2021a], we thus add a small regularization term $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{norm}} + \mathcal{L}_{\text{lap}}$ consisting of a normal consistency loss [Desbrun et al., 1999] $\mathcal{L}_{\text{norm}}$ and a Laplacian smoothing loss [Nealen et al., 2006] $\mathcal{L}_{\text{lap}}$.

Progressive conditioning. The goal of *progressive conditioning* is to encourage the model to share elements (*e.g.*, shape, texture, background) across instances to prevent degenerate solutions. Inspired by the curriculum learning philosophy [Elman, 1993; Wang et al., 2018; Monnier et al., 2020], we propose to do so by gradually increasing the latent space representing the shape, texture and background. Intuitively, restricting the latent space implicitly encourages maximizing the information shared across instances. For example, a latent space of dimension 0 (*i.e.*, no conditioning) amounts to learning a global representation that is the same for all instances, while a latent space of dimension 1 restricts all the generated shapes, textures or backgrounds to lie on a 1-dimensional manifold. Progressively increasing the size of the latent code during training can be interpreted as gradually specializing from category-level to instance-level knowledge. Figure 5.2a illustrates the procedure with example results where we can observe the progressive specialization to particular instances: reactors gradually appear/disappear, textures get more accurate. Because common neural network implementations have fixed-size inputs, we implement progressive conditioning by masking, stage-by-stage, a decreasing number of values of the latent code. All our experiments share the same 4-stage training strategy where the latent code dimension is increased at the beginning of each stage and the network is then trained until convergence. We use dimensions 0/2/8/64 for the shape code, 2/8/64/512 for the texture code and 4/8/64/256 for the background code. We provide real-image results for each stage in Appendix C.2.

Neighbor reconstruction. The idea behind *neighbor reconstruction* is to explicitly enforce consistency between different instances. Our key assumption is that neighboring instances with similar shape or texture exist in the dataset. If such neighbors are correctly identified, switching their shape or texture in our generation model should give similar reconstruction results. For a given input image, we hence propose to use its shape or texture attribute in the image formation process of neighboring instances and apply our reconstruction loss on associated renderings. Intuitively, this process can be seen as mimicking a multi-view supervision without actually having access to multi-view images by finding neighboring instances in well-designed latent spaces. Figure 5.2b illustrates the procedure with an example.

More specifically, let $\{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}, \mathbf{a}, \mathbf{z}_{\text{bg}}\}$ be the parameters predicted by our encoder for a given input training image $\mathbf{I}$, let Ω be a memory bank storing the images and parameters of the last M instances processed by the network. We write $\Omega^{(m)} = \{\mathbf{I}^{(m)}, \mathbf{z}_{\text{sh}}^{(m)}, \mathbf{z}_{\text{tx}}^{(m)}, \mathbf{a}^{(m)}, \mathbf{z}_{\text{bg}}^{(m)}\}$ each of these M instances and associated parameters. We first select the closest instance from the memory bank Ω in the texture (respectively shape) code space using the L_2 distance, $m_t = \text{argmin}_m \|\mathbf{z}_{\text{tx}} - \mathbf{z}_{\text{tx}}^{(m)}\|_2$ (respectively $m_s = \text{argmin}_m \|\mathbf{z}_{\text{sh}} - \mathbf{z}_{\text{sh}}^{(m)}\|_2$). We then swap the codes and generate the reconstruction $\hat{\mathbf{I}}_{\text{tx}}^{(m_t)}$ (respectively $\hat{\mathbf{I}}_{\text{sh}}^{(m_s)}$) using the parameters $\{\mathbf{z}_{\text{sh}}^{(m_t)}, \mathbf{z}_{\text{tx}}, \mathbf{a}^{(m_t)}, \mathbf{z}_{\text{bg}}^{(m_t)}\}$ (respectively $\{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}^{(m_s)}, \mathbf{a}^{(m_s)}, \mathbf{z}_{\text{bg}}^{(m_s)}\}$). Finally, we compute the reconstruction loss between the images $\mathbf{I}^{(m_t)}$ and $\hat{\mathbf{I}}_{\text{tx}}^{(m_t)}$ (respectively $\mathbf{I}^{(m_s)}$ and $\hat{\mathbf{I}}_{\text{sh}}^{(m_s)}$). Our full loss can thus be written:

$$\mathcal{L}_{\text{nbr}} = \mathcal{L}_{\text{rec}}(\mathbf{I}^{(m_t)}, \hat{\mathbf{I}}_{\text{tx}}^{(m_t)}) + \mathcal{L}_{\text{rec}}(\mathbf{I}^{(m_s)}, \hat{\mathbf{I}}_{\text{sh}}^{(m_s)}). \tag{5.2}$$

Note that we recompute the parameters of the selected instances with the current network state, to avoid uncontrolled effects of changes in the network state. Also note that, for computational reasons, we do not use this loss in the first stage where codes are almost the same for all instances.

To prevent latent codes from specializing by viewpoint, we split the viewpoints into V bins w.r.t. the rotation angle, uniformly sample a target bin for each input and look for the nearest instances only in the subset of instances within the target viewpoint range (see Appendix C.2 for details). In all experiments, we use $V = 5$ and a memory bank of size $M = 1024$.

5.3.3 Alternate 3D and pose learning

Because predicting 6D poses is hard due to self-occlusions and local minima, we follow prior works [Insafutdinov and Dosovitskiy, 2018; Henderson and Ferrari, 2019; Goel et al., 2020; Tulsiani et al., 2020] and predict multiple pose candidates and their likelihood. However, we identify failure modes in their optimization framework (detailed in Appendix C.2) and instead propose a new optimization that alternates between 3D and pose learning. More

specifically, given an input image $\mathbf{I}$, we predict K pose candidates $\{(\mathbf{r}_1, \mathbf{t}_1), \ldots, (\mathbf{r}_K, \mathbf{t}_K)\}$, and their associated probabilities $\mathbf{p}_{1:K}$. We render the model from the different poses, yielding K reconstructions $\hat{\mathbf{I}}_{1:K}$. We then alternate the learning between two steps: (i) the *3D-step* where shape, texture and background branches of the network are updated by minimizing $\mathcal{L}_{\text{3D}}$ using the pose associated to the highest probability, and (ii) the *P-step* where the branches of the network predicting candidate poses and their associated probabilities are updated by minimizing:

$$\mathcal{L}_{\text{P}} = \textstyle\sum_k \mathbf{p}_k \mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}_k) + \lambda_{\text{uni}} \mathcal{L}_{\text{uni}}, \tag{5.3}$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction loss described in Section 5.3.2, $\mathcal{L}_{\text{uni}}$ is a regularization loss on the predicted poses and λ_{uni} is a scalar hyperparameter. More precisely, we use $\mathcal{L}_{\text{uni}} = \sum_k |\bar{\mathbf{p}}_k - 1/K|$ where $\bar{\mathbf{p}}_k$ is the averaged probabilities for candidate k in a particular training batch. Similar to Henderson and Ferrari [2019], we found it crucial to introduce this regularization term to encourage the use of all pose candidates. In particular, this prevents a collapse mode where only few pose candidates are used. Note that we do not use the neighbor reconstruction loss nor the mesh regularization loss which are not relevant for viewpoints. In all experiments, we use $\lambda_{\text{uni}} = 0.02$.

Inspired by the camera multiplex of Goel et al. [2020], we parametrize rotations with the classical Euler angles (azimuth, elevation and roll) and rotation candidates correspond to offset angles w.r.t. reference viewpoints. Since in practice elevation has limited variations, our reference viewpoints are uniformly sampled along the azimuth dimension. Note that compared to Goel et al. [2020], we do not directly optimize a set of pose candidates per training image, but instead learn a set of K predictors for the entire dataset. We use $K = 6$ in all experiments.

5.4 Experiments

We validate our approach in two standard setups. It is first quantitatively evaluated on ShapeNet where state-of-the-art methods use multiple views as supervision. Then, we compare it on standard real-image benchmarks and demonstrate its applicability to more complex datasets. Finally, we present an ablation study.

5.4.1 Evaluation on the ShapeNet benchmark

We compare our approach to state-of-the-art methods using multi-views, viewpoints and silhouettes as supervision. Our method is instead learned without supervision. For all compared methods, one model is trained per class. We adhere to community standards [Kato et al., 2018; Liu et al., 2019; Niemeyer et al., 2020] and use the renderings and splits from Kato

et al. [2018] of the ShapeNet dataset [Chang et al., 2015]. It corresponds to a subset of 13 classes of 3D objects, each object being rendered into a 64×64 image from 24 viewpoints uniformly spaced along the azimuth dimension. We evaluate all methods using the standard Chamfer-L_1 distance [Mescheder et al., 2019; Niemeyer et al., 2020], where predicted shapes are pre-aligned using our gradient-based version of the Iterative Closest Point (ICP) [Besl and McKay, 1992] with anisotropic scaling. Indeed, compared to competing methods having access to the ground-truth viewpoint during training, we need to predict it for each input image in addition to the 3D shape. This yields to both shape/pose ambiguities (*e.g.*, a small nearby object or a bigger one far from the camera) and small misalignment errors that dramatically degrade the performances. We provide evaluation details as well as results without ICP in Appendix C.3.

We report quantitative results and compare to the state of the art in Table 5.2, where methods using multi-views are visually separated. We evaluate the pre-trained weights for SDF-SRN [Lin et al., 2020a] and train the models from scratch using the official implementation for DVR [Niemeyer et al., 2020]. We tried evaluating SMR [Hu et al., 2021] but could not reproduce the results. We do not compare to TARS [Duggal and Pathak, 2022] which is based on SDF-SRN and share the same performances. Our approach achieves results that are on average better than the state-of-the-art methods supervised with silhouette and viewpoint annotations. This is a strong result: while silhouettes are trivial in this benchmark, learning without viewpoint annotations is extremely challenging as it involves solving the pose estimation and shape reconstruction problems simultaneously. For some categories, our performances are even better than DVR supervised with multiple views. This shows that our system learned on raw images generates 3D reconstructions comparable to the ones obtained from methods using geometry cues like silhouettes and multiple views. Note that for the lamp category, our method predicts degenerate 3D shapes; we hypothesize this is due to their rotation invariance which makes the viewpoint estimation ambiguous.

We visualize and compare the quality of our 3D reconstructions in Figure 5.5. The first three examples correspond to examples advertised in DVR [Niemeyer et al., 2020], the last two corresponds to examples we selected. Our method generates textured meshes of high-quality across all these categories. The geometry obtained is sharp and accurate, and the predicted texture mostly corresponds to the input.

5.4.2 Results on real images

Pascal3D+ Car and CUB benchmarks. We compare our approach to state-of-the-art SVR methods on real images, where multiple views are not available. All competing methods use silhouette supervision and output meshes that are symmetric. CMR [Kanazawa et al., 2018] additionally use keypoints, UCMR [Goel et al., 2020] and IMR [Tulsiani et al., 2020] starts

Method	**Ours**	SDF-SRN	DVR	DVR
MV				✓
CK		✓	✓	✓
S		✓	✓	✓
airplane	**0.110**	0.128	0.114	**0.111**
bench	**0.159**	-	0.255	**0.176**
cabinet	**0.137**	-	0.254	**0.158**
car	0.168	**0.150**	0.203	**0.153**
chair	**0.253**	0.262	0.371	**0.205**
display	**0.220**	-	0.257	**0.163**
lamp	0.523	-	**0.363**	**0.281**
phone	**0.127**	-	0.191	**0.076**
rifle	**0.097**	-	0.130	**0.083**
sofa	**0.192**	-	0.321	**0.160**
speaker	**0.224**	-	0.312	**0.215**
table	**0.243**	-	0.303	**0.230**
vessel	**0.155**	-	0.180	**0.151**
mean	**0.201**	-	0.250	**0.166**

Table 5.2: **ShapeNet comparison.** We report Chamfer-L_1 $\downarrow$ obtained after ICP, **best** results are highlighted. Supervisions are: Multi-Views, Camera or Keypoints, Silhouettes.

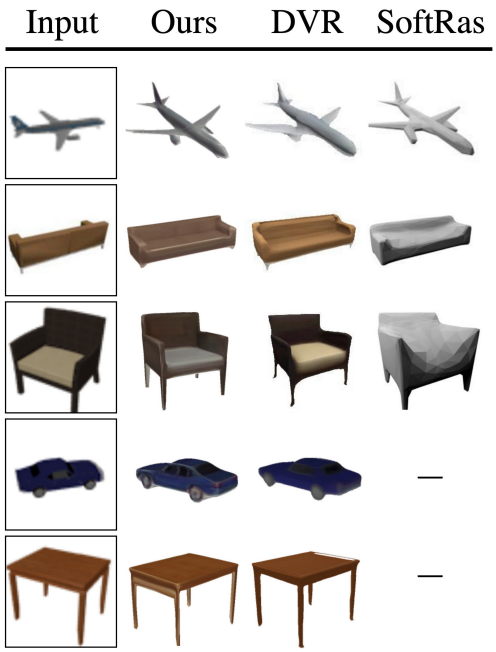


Figure 5.5: **Visual comparisons.** We compare to DVR [Niemeyer et al., 2020] and SoftRas [Liu et al., 2019] learned with full supervision (MV, CK, S).

learning from a given template shape; we *do not* use any of these and directly learn from raw images. We strictly follow the community standards [Kanazawa et al., 2018; Goel et al., 2020; Tulsiani et al., 2020] and use the train/test splits of Pascal3D+ Car [Xiang et al., 2014] (5000/220 images) and CUB-200-2011 [Welinder et al., 2010] (5964/2874 images). Images are square-cropped around the object using bounding box annotations and resized to 64×64.

A quantitative comparison is summarized in Table 5.3, where we report 3D IoU, Chamfer-L_1 (with ICP alignment), Mask IoU for Pascal3D+ Car, and Percentage of Correct Keypoints thresholded at $\alpha = 0.1$ (PCK@0.1) [Kulkarni et al., 2019], Mask IoU for CUB. Our approach yields competitive results across all metrics although it does not rely on any supervision used by other works. On Pascal3D+ Car, we achieve significantly better results than UCMR for Chamfer-L_1 and Mask IoU, which we argue are less biased metrics than the standard 3D IoU [Tulsiani et al., 2017b; Kanazawa et al., 2018] computed on unaligned shapes (see Appendix C.3). On CUB, our approach achieves reasonable results that are however slightly worse than the state of the art. We hypothesize this is linked to our pose regularization term encouraging the use of all viewpoints whereas these bird images clearly lack back views.

We qualitatively compare our approach to the state of the art in Figure 5.6. For each input, we show the mesh rendered from two viewpoints. For our car results, we additionally show meshes with synthetic textures to emphasize correspondences. Qualitatively, our approach

	Supervision			Pascal3D+ Car			CUB-200-2011	
Method	CK	S	A	3D IoU ↑	Ch-L_1 ↓	Mask IoU ↑	PCK@0.1 ↑	Mask IoU ↑
CMR [Kanazawa et al., 2018]	✓	✓	✓	64	-	-	48.3	70.6
IMR [Tulsiani et al., 2020]		✓	✓	-	-	-	53.5	-
UMR [Li et al., 2020]		✓	✓	62	-	-	58.2	73.4
UCMR [Goel et al., 2020]		✓	✓	**67.3**	0.172	73.7	-	63.7
SMR [Hu et al., 2021]		✓	✓	-	-	-	**62.2**	**80.6**
Ours				65.9	**0.163**	**83.9**	49.0	71.4

Table 5.3: **Real-image quantitative comparisons.** Supervision corresponds to Camera or Keypoints, Silhouettes, Assumptions (see Table 5.1 for details).

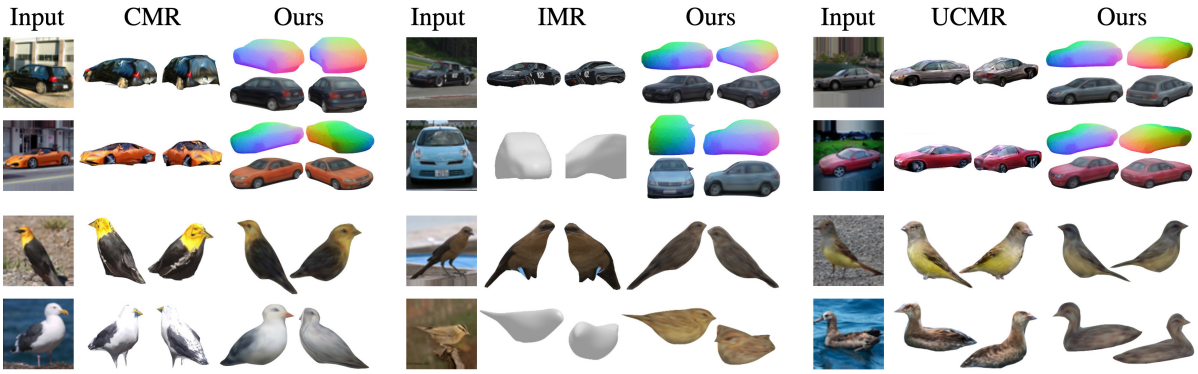
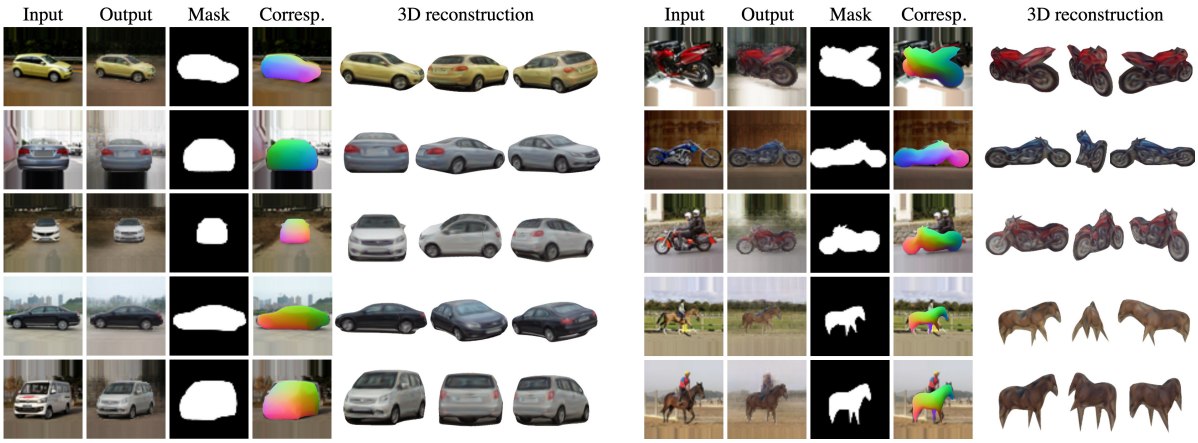


Figure 5.6: **Real-image comparisons.** We show reconstructions on Pascal3D+ Cars (top) and CUB (bottom) and compare to CMR [Kanazawa et al., 2018], IMR [Tulsiani et al., 2020], UCMR [Goel et al., 2020].

yields results on par with prior works both in terms of geometric accuracy and overall realism. Although the textures obtained in Tulsiani et al. [2020] look more accurate, they are modeled as pixel flows, which has a clear limitation when synthesizing unseen texture parts. Note that we do not recover details like the bird legs which are missed by prior works due to coarse silhouette annotations. We hypothesize we also miss them because they are hardly consistent across instances, *e.g.*, legs can be bent in multiple ways.

Real-word datasets. Motivated by 3D-aware image synthesis methods that are learned in-the-wild [Nguyen-Phuoc et al., 2019; Niemeyer and Geiger, 2021], we investigate whether our approach can be applied to real-world datasets where silhouettes are not available and images are not methodically cropped around the object. We adhere to standards from the 3D-aware image synthesis community [Nguyen-Phuoc et al., 2019; Niemeyer and Geiger, 2021] and apply our approach to 64×64 images of CompCars [Yang et al., 2015]. In addition, we provide results for the more difficult scenario of LSUN images [Yu et al., 2016] for motorbikes and horses. Because many LSUN images are noise, we filter the datasets as follows: we manually



(a) CompCars [Yang et al., 2015] (b) LSUN Motorbike and Horse [Yu et al., 2016]

Figure 5.7: **Real-world dataset results.** From left to right, we show: input and output images, the predicted mask, a correspondence map and the mesh rendered from 3 viewpoints. Note that for LSUN Horse, the geometry quality is low and outlines our approach limitations (see text). Best viewed digitally.

select 16 reference images with different poses, find the nearest neighbors from the first 200k images in a pre-trained ResNet-18 [He et al., 2016] feature space, and keep the top 2k for each reference image. We repeat the procedure with flipped reference images yielding 25k images.

Our results are shown in Figure 5.7. For each input image, we show from left to right: the output image, the predicted mask, a correspondence map, and the 3D reconstruction rendered from the predicted viewpoint and two other viewpoints. Although our approach is trained to synthesize images, these are all natural by-products. While the quality of our 3D car reconstructions is high, the reconstructions obtained for LSUN images lack some realism and accuracy (especially for horses), thus outlining limitations of our approach. However, our segmentation and correspondence maps emphasize our system ability to accurately localize the object and find correspondences, even when the geometry is coarse.

Limitations. Even if our approach is a strong step towards generic unsupervised SVR, we can outline three main limitations. First, the lack of different views in the data harms the results (*e.g.*, most CUB birds have concave backs); this can be linked to our uniform pose regularization term which is not adequate in these cases. Second, complex textures are not predicted correctly (*e.g.*, motorbikes in LSUN). Although we argue it could be improved by more advanced autoencoders, the neighbor reconstruction term may prevent unique textures to be generated. Finally, despite its applicability to multiple object categories and diverse datasets, our multi-stage progressive training is cumbersome and an automatic way of progressively specializing to instances is much more desirable.

Model	Full	w/o $\mathcal{L}_{\text{nbr}}$	w/o PC
airplane	0.110	0.124	**0.107**
bench	**0.159**	0.188	0.206
car	**0.168**	0.179	0.173
chair	**0.253**	0.319	0.527
table	**0.243**	0.246	0.598
mean	**0.187**	0.211	0.322

Table 5.4: **Ablation results on ShapeNet** [Chang et al., 2015]**.**

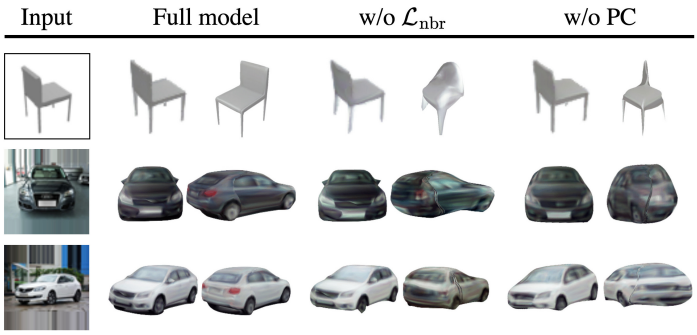


Figure 5.8: **Ablation visual results.** For each input, we show the mesh rendered from two viewpoints.

5.4.3 Ablation study

We analyze the influence of our neighbor reconstruction loss $\mathcal{L}_{\text{nbr}}$ and progressive conditioning (PC) by running experiments without each component.

First, we provide quantitative results on ShapeNet in Table 5.4. When $\mathcal{L}_{\text{nbr}}$ is removed, the results are worse for almost all categories, outlining that it is important to the predicted geometry accuracy. When PC is removed, results are comparable to the full model for airplane and car but much worse for chair and table. Indeed, they involve more complex shapes and our system falls into a bad minimum with degenerate solutions, a scenario that is avoided with PC.

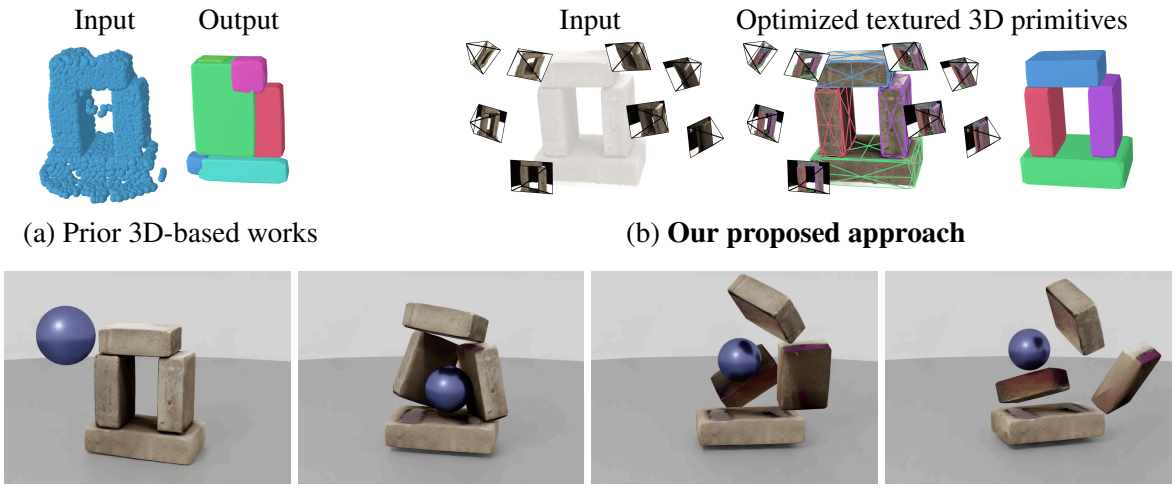
Second, we perform a visual comparison on ShapeNet and CompCars examples in Figure 5.8. For each input, we show the mesh rendered from the predicted viewpoint and a different viewpoint. When $\mathcal{L}_{\text{nbr}}$ is removed, we observe that the reconstruction seen from the predicted viewpoint is correct but it is either wrong for chairs and degraded for cars when seen from the other viewpoint. Indeed, the neighbor reconstruction explicitly enforces the unseen reconstructed parts to be consistent with other instances. When PC is removed, we observe degenerate reconstructions where the object seen from a different viewpoint is not realistic.

5.5 Conclusion

We presented UNICORN, an unsupervised SVR method which, in contrast to all prior works, learns from raw images only. We demonstrated it yields high-quality results for diverse shapes as well as challenging real-world image collections. This was enabled by two key contributions aiming at leveraging consistency across different instances: our *progressive conditioning* training strategy and *neighbor reconstruction* loss. We believe our work includes both an important step forward for unsupervised SVR and the introduction of a valuable conceptual insight.

Chapter 6

Differentiable Blocks World



(a) Prior 3D-based works

(b) **Our proposed approach**

(c) Application: physics-based simulations

Figure 6.1: **Differentiable Blocks World. (a)** Prior works fit primitives to point clouds and typically fail for real data where ground-truth point clouds are extremely noisy and incomplete. **(b)** We propose using calibrated multi-view images instead and simultaneously tackle 3D decomposition and 3D reconstruction by rendering learnable textured primitives in a differentiable manner. **(c)** Such a textured decomposition is highly compact and user-friendly: it enables us to do physics-based simulations, *e.g.*, throwing a ball at the discovered primitives.

6.1 Introduction

Recent multi-view modeling approaches, building on Neural Radiance Fields [Mildenhall et al., 2020], capture scenes with astonishing accuracy by optimizing a dense occupancy and color model. However, they do not incorporate any notion of objects, they are not easily interpretable

for a human user or a standard 3D modeling software, and they are not useful for physical understanding of the scene. In fact, even though these approaches can achieve a high-quality 3D reconstruction, the recovered content is nothing but a soup of colorful particles! In contrast, we propose an approach that recovers textured primitives, which are compact, actionable, and interpretable.

More concretely, our method takes as input a collection of calibrated images of a scene, and optimizes a set of primitive meshes parametrized by superquadrics [Barr, 1981] and their UV textures to minimize a rendering loss. The approach we present is robust enough to work directly from a random initialization. One of its key components is the optimization of a transparency parameter for each primitive, which helps in dealing with occlusions as well as handling varying number of primitives. This notably requires adapting standard differentiable renderers to deal with transparency. We also show the benefits of using a perceptual loss, a total variation regularization on the textures and a parsimony loss favoring the use of a minimal number of primitives.

Our scene representation harks back to the classical Blocks World ideas [Roberts, 1963]. An important difference is that the Blocks World-inspired approaches are typically bottom-up, leveraging low-level image features, such as edges [Roberts, 1963], super-pixels [Gupta et al., 2010], or more recently learned features [Xiao et al., 2012; Kluger et al., 2021], to infer 3D blocks. In contrast, we perform a direct top-down optimization of 3D primitives and texture using a rendering loss, starting from a random initialization in the spirit of analysis-by-synthesis. Unlike related works that fit primitives to 3D point clouds [Binford, 1971; Barr, 1981; Tulsiani et al., 2017a; Li et al., 2019b; Wu et al., 2022; Liu et al., 2022; Loiseau et al., 2023] (Figure 6.1a), our approach, dubbed *Differentiable Blocks World* (or DBW), does not require any 3D reconstruction *a priori* but instead operates directly on a set of calibrated input images, leveraging photometric consistency across different views (Figure 6.1b). This makes our approach more robust since methods based on 3D are very sensitive to noise in the reconstructions and have difficulties dealing with incomplete objects. Our setting is similar to existing NeRF-like approaches, but our model is able to recover a significantly more interpretable and parsimonious representation. In particular, such an interpretable decomposition allows us to easily play with the discovered scene, *e.g.*, by performing physics-based simulations (Figure 6.1c). Code and video results are available on our project webpage: www.tmonnier.com/DBW.

6.2 Approach

Given a set of N views $\mathbf{I}_{1:N}$ of a scene associated with camera poses $\mathbf{c}_{1:N}$, our goal is to decompose the 3D scene into geometric primitives that best explain the images. We explicitly

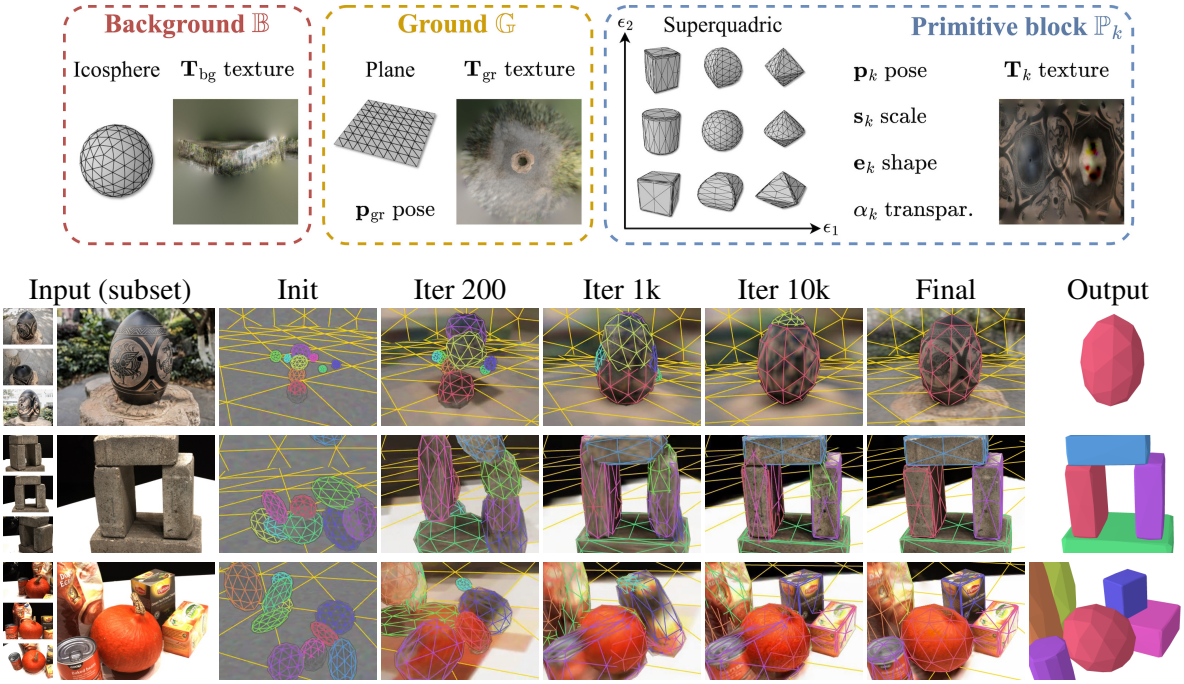


Figure 6.2: **Overview**. **(top)** We model the world as an explicit set of learnable textured meshes that are assembled together in the 3D space. **(bottom)** Starting from a random initialization, we optimize such a representation through differentiable rendering by photometric consistency across the different views.

model the scene as a set of transparent superquadric meshes, whose parameters, texture and number are optimized to maximize photoconsistency through differentiable rendering. Note that compared to recent advances in neural volumetric representations [Niemeyer et al., 2020; Mildenhall et al., 2020; Yu et al., 2021], we *do not* use any neural network and directly optimize meshes, which are straightforward to use in computer graphic pipelines.

Notations. We use bold lowercase for vectors (*e.g.*, $\mathbf{a}$), bold uppercase for images (*e.g.*, $\mathbf{A}$), double-struck uppercase for meshes (*e.g.*, $\mathbb{A}$) and write $a_{1:N}$ the ordered set $\{a_1, \ldots, a_n\}$.

6.2.1 Parametrizing a world of blocks

We propose to represent the world scene as an explicit set of textured meshes positioned in the 3D space. Figure 6.2 summarizes our modeling and the parameters updated (top) during the optimization (bottom). Specifically, we model each scene as a union of primitive meshes: (i) an icosphere $\mathbb{B}$ modeling a background dome and centered on the scene, (ii) a plane $\mathbb{G}$ modeling the ground, and (iii) K primitive blocks $\mathbb{P}_{1:K}$ in the form of superquadric meshes, where K is fixed and refers to a maximum number of blocks. Unless mentioned otherwise, we arbitrarly use $K = 10$. We write the resulting scene mesh $\mathbb{B} \cup \mathbb{G} \cup \mathbb{P}_1 \cup \ldots \cup \mathbb{P}_K$.

The goal of the background dome is to model things far from the cameras that can be well approximated with a planar surface at infinity. In practice, we consider an icosphere with a fixed location and a fixed scale that is much greater than the scene scale. On the contrary, the goal of the planar ground and the blocks is to model the scene close to the cameras. We thus introduce rigid transformations modeling locations that will be updated during the optimization. Specifically, we use the 6D rotation parametrization of Zhou et al. [2019] and associate to each block k a pose $\mathbf{p}_k = \{\mathbf{r}_k, \mathbf{t}_k\} \in \mathbb{R}^9$ such that every point of the block $\mathbf{x} \in \mathbb{R}^3$ is transformed into world space by $\mathbf{x}_{\text{world}} = \text{rot}(\mathbf{r}_k)\mathbf{x} + \mathbf{t}_k$, where $\mathbf{t}_k \in \mathbb{R}^3$, $\mathbf{r}_k \in \mathbb{R}^6$ and rot maps a 6D vector to a rotation matrix [Zhou et al., 2019]. Similarly, we associate a rigid transformation $\mathbf{p}_{\text{gr}} = \{\mathbf{r}_{\text{gr}}, \mathbf{t}_{\text{gr}}\}$ to the ground plane. We next describe how we model variable number of blocks via transparency values and the parametrization of blocks' shape and texture.

Block existence through transparency. Modeling a variable number of primitives is a difficult task as it involves optimizing over a discrete random variable. Recent works tackle the problem using reinforcement learning [Tulsiani et al., 2017a], probabilistic approximations [Paschalidou et al., 2019] or greedy algorithms [Monnier et al., 2021], which often yield complex optimization strategies. In this work, we instead propose to handle variable number of primitive blocks by modeling meshes that are *transparent*. Specifically, we associate to each block k a learnable transparency value α_k, parametrized with a sigmoid, that can be pushed towards zero to change the effective number of blocks. Such transparencies are not only used in our rendering process to softly model the blocks existence and occlusions (Section 6.2.2), but also in regularization terms during our optimization, *e.g.*, to encourage parsimony in the number of blocks used (Section 6.2.3).

Superquadric block shape. We model blocks with superquadric meshes. Introduced by Barr [1981] and revived recently by Paschalidou et al. [2019], superquadrics define a family of parametric surfaces that exhibits a strong expressiveness with a small number of continuous parameters, thus making a good candidate for primitive fitting by gradient descent. More concretely, we derive a superquadric mesh from a unit icosphere. For each vertex of the icosphere, its spherical coordinates $\eta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\omega \in [-\pi, \pi]$ are mapped to the superquadric surface through the parametric equation [Barr, 1981]:

$$\Phi(\eta, \omega) = \begin{bmatrix} s_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ s_2 \sin^{\epsilon_1} \eta \\ s_3 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \end{bmatrix}, \tag{6.1}$$

where $\mathbf{s} = \{s_1, s_2, s_3\} \in \mathbb{R}^3$ represents an anisoptropic scaling and $\mathbf{e} = \{\epsilon_1, \epsilon_2\} \in \mathbb{R}^2$ defines the shape of the superquadric. Both $\mathbf{s}$ and $\mathbf{e}$ are updated during the optimization process. Note that by design, each vertex of the icosphere is mapped continuously to a vertex of the superquadric mesh, so the icosphere connectivity - and thus the icosphere faces - is transferred to the superquadric mesh.

Texturing model. We use texture mapping to model scene appearance. Concretely, we optimize $K + 2$ texture images $\{\mathbf{T}_{\text{bg}}, \mathbf{T}_{\text{gr}}, \mathbf{T}_{1:K}\}$ which are UV-mapped onto each mesh triangle using pre-defined UV mappings. Textures for the background and the ground are trivially obtained using respectively spherical coordinates of the icosphere and a simple plane projection. For a given block k, each vertex of the superquadric mesh is associated to a vertex of the icosphere. Therefore, we can map the texture image $\mathbf{T}_k$ onto the superquadric by first mapping it to the icosphere using a fixed UV map computed with spherical coordinates, then mapping the icosphere triangles to the superquadric ones (see Appendix D.3 for details).

6.2.2 Differentiable rendering

In order to optimize our scene parameters to best explain the images, we propose to leverage recent mesh-based differentiable renderers [Liu et al., 2019; Chen et al., 2019b; Ravi et al., 2020]. Similar to them, our differentiable rendering corresponds to the soft rasterization of the mesh faces followed by a blending function. In contrast to existing mesh-based differentiable renderers, we introduce the ability to account for transparency. Intuitively, our differentiable rendering can be interpreted as an alpha compositing of the transparent colored faces of the mesh. In the following, we write pixel-wise multiplication with $\odot$ and the division of image-sized tensors corresponds to pixel-wise division.

Soft rasterization. Given a 2D pixel location $\mathbf{u}$, we model the influence of the face j projected onto the image plane with the 2D occupancy function of Chen et al. [2019b] that we modify to incorporate the transparency value α_{k_j} associated to this face. Specifically, we write the occupancy function as:

$$\mathcal{O}_j^{\text{2D}}(\mathbf{u}) = \alpha_{k_j} \exp\left(\min\left(\frac{\Delta_j(\mathbf{u})}{\sigma},\ 0\right)\right), \tag{6.2}$$

where σ is a scalar hyperparameter modeling the extent of the soft mask of the face and $\Delta_j(\mathbf{u})$ is the signed Euclidean distance between pixel $\mathbf{u}$ and projected face j, such that $\Delta_j(\mathbf{u}) < 0$ if pixel $\mathbf{u}$ is outside face j and $\Delta_j(\mathbf{u}) \geq 0$ otherwise. We consider the faces belonging to the

background and the ground to be opaque, *i.e.*, use a transparency of 1 for all their faces in the occupancy function.

Blending through alpha compositing. For each pixel, we find all projected faces with an occupancy greater than a small threshold at this pixel location, and sort them by increasing depth. Denoting by L the maximum number of faces per pixel, we build image-sized tensors for occupancy $\mathbf{O}_\ell$ and color $\mathbf{C}_\ell$ by associating to each pixel the ℓ-th intersecting face attributes. The color is obtained through barycentric coordinates, using clipped barycentric coordinates for locations outside the face. Different to most differentiable renderers and as advocated by Monnier et al. [2022], we directly interpret these tensors as an ordered set of RGBA image layers and blend them through traditional alpha compositing [Porter and Duff, 1984]:

$$\mathcal{C}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}) = \sum_{\ell=1}^{L} \left(\prod_{p<\ell}^{L} (1 - \mathbf{O}_p) \right) \odot \mathbf{O}_\ell \odot \mathbf{C}_\ell \ . \tag{6.3}$$

We found this simple alpha composition to behave better during optimization than the original blending function used in Liu et al. [2019]; Chen et al. [2019b]; Ravi et al. [2020]. This is notably in line with recent advances in differentiable rendering like NeRF [Mildenhall et al., 2020] which can be interpreted as alpha compositing points along the rays.

6.2.3 Optimizing a differentiable blocks world

We optimize our scene parameters by minimizing a rendering loss across batches of images using gradient descent. Specifically, for each image $\mathbf{I}$, we build the scene mesh as described in Section 6.2.1 and use the associated camera pose to render an image $\hat{\mathbf{I}}$ using the rendering process detailed in Section 6.2.2. We optimize an objective function defined as:

$$\mathcal{L} = \mathcal{L}_{\text{render}} + \lambda_{\text{parsi}} \mathcal{L}_{\text{parsi}} + \lambda_{\text{TV}} \mathcal{L}_{\text{TV}} + \lambda_{\text{over}} \mathcal{L}_{\text{over}} \ , \tag{6.4}$$

where $\mathcal{L}_{\text{render}}$ is a rendering loss between $\mathbf{I}$ and $\hat{\mathbf{I}}$, $\lambda_{\text{parsi}}, \lambda_{\text{TV}}, \lambda_{\text{over}}$ are scalar hyperparameters and $\mathcal{L}_{\text{parsi}}, \mathcal{L}_{\text{TV}}, \mathcal{L}_{\text{over}}$ are regularization terms respectively encouraging parsimony in the use of primitives, favoring smoothness in the texture maps and penalizing the overlap between primitives. Our rendering loss is composed of a pixel-wise MSE loss $\mathcal{L}_{\text{MSE}}$ and a perceptual LPIPS loss [Zhang et al., 2018] $\mathcal{L}_{\text{perc}}$ such that $\mathcal{L}_{\text{render}} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{perc}} \mathcal{L}_{\text{perc}}$. In all experiments, we use $\lambda_{\text{parsi}} = 0.01$, $\lambda_{\text{perc}} = \lambda_{\text{TV}} = 0.1$ and $\lambda_{\text{over}} = 1$. Figure 6.2 (bottom) shows the evolution of our renderings throughout the optimization.

Encouraging parsimony and texture smoothness. We found that regularization terms were critical to obtain meaningful results. In particular, the raw model typically uses the maximum number of blocks available to reconstruct the scene, thus over-decomposing the scene. To adapt the number of blocks per scene and encourage parsimony, we use the transparency values as a proxy for the number of blocks used and penalize the loss by $\mathcal{L}_{\text{parsi}} = \sum_k \sqrt{\alpha_k}/K$. We also use a total variation (TV) penalization [Rudin and Oshe, 1994] on the texture maps to encourage uniform textures. Given a texture map $\mathbf{T}$ of size $U \times V$ and denoting by $\mathbf{T}[u, v] \in \mathbb{R}^3$ the RGB values of the pixel at location (u, v), we define:

$$\mathcal{L}_{\text{tv}}(\mathbf{T}) = \frac{1}{UV} \sum_{u,v} \left(\left\| \mathbf{T}[u+1, v] - \mathbf{T}[u, v] \right\|_2^2 + \left\| \mathbf{T}[u, v+1] - \mathbf{T}[u, v] \right\|_2^2 \right), \tag{6.5}$$

and write $\mathcal{L}_{\text{TV}} = \mathcal{L}_{\text{tv}}(\mathbf{T}_{\text{bg}}) + \mathcal{L}_{\text{tv}}(\mathbf{T}_{\text{gr}}) + \sum_k \mathcal{L}_{\text{tv}}(\mathbf{T}_k)$ the final penalization.

Penalizing overlapping blocks. We introduce a regularization term encouraging primitives to not overlap. Because penalizing volumetric intersections of superquadrics is difficult and computationally expensive, we instead propose to use a Monte Carlo alternative, by sampling 3D points in the scene and penalizing points belonging to more than λ blocks, in a fashion similar to Paschalidou et al. [2021]. Following Paschalidou et al. [2021], λ is set to 1.95 so that blocks could slightly overlap around their surface thus avoiding unrealistic floating blocks. More specifically, considering a block k and a 3D point $\mathbf{x}$, we define a soft 3D occupancy function $\mathcal{O}_k^{\text{3D}}$ as:

$$\mathcal{O}_k^{\text{3D}}(\mathbf{x}) = \alpha_k \, \text{sigmoid} \left(\frac{1 - \Psi_k(\mathbf{x})}{\tau} \right), \tag{6.6}$$

where τ is a temperature hyperparameter and Ψ_k is the superquadric inside-outside function [Barr, 1981] associated to the block k, such that $\Psi_k(\mathbf{x}) \leq 1$ if $\mathbf{x}$ lies inside the superquadric and $\Psi_k(\mathbf{x}) > 1$ otherwise. Given a set of M 3D points Ω, our final regularization term can be written as:

$$\mathcal{L}_{\text{over}} = \frac{1}{M} \sum_{\mathbf{x} \in \Omega} \max \left(\sum_{k=1}^{K} \mathcal{O}_k^{\text{3D}}(\mathbf{x}), \ \lambda \right). \tag{6.7}$$

Note that in practice, for better efficiency and accuracy, we only sample points in the region where blocks are located, which can be identified using the block poses $\mathbf{p}_{1:K}$.

Optimization details. We found that two elements were key to avoid bad local minima during optimization. First, while transparent meshes enable differentiability w.r.t. the number of primitives, we observed a failure mode where two semi opaque meshes model the same 3D region. To prevent this behavior, we propose to inject gaussian noise before the sigmoid in the transparencies $\alpha_{1:K}$ to create stochasticity when values are not close to the sigmoid saturation,

Method	Input	Chamfer Distance (CD) per scene										Mean CD	Mean #P
		S24	S31	S40	S45	S55	S59	S63	S75	S83	S105		
EMS [Liu et al., 2022]	3D	6.77	5.93	3.36	6.91	6.52	3.50	4.72	7.08	7.25	6.10	5.82	7.4
MBF [Ramamonjisoa et al., 2022]	3D	3.73	4.79	4.31	3.95	3.26	4.00	3.66	3.92	3.97	**4.25**	3.98	16.4
Ours (random)	Img	5.41	**3.13**	1.57	4.93	3.08	3.66	**3.40**	2.78	3.94	4.85	3.67	**4.6**
Ours (auto)	Img	**3.25**	**3.13**	**1.16**	**3.02**	**2.98**	**2.32**	**3.40**	**2.78**	**3.43**	5.21	**3.07**	5.0
EMS [Liu et al., 2022] + ps	3D	6.32	4.11	2.98	4.94	4.26	3.03	3.60	5.44	3.24	4.43	4.23	8.3
MBF [Ramamonjisoa et al., 2022] + ps	3D	**3.35**	**2.95**	**2.61**	**2.19**	**2.53**	**2.47**	**1.97**	**2.60**	**2.60**	**3.27**	**2.65**	29.9

Table 6.1: **Quantitative results on DTU.** We use the official DTU evaluation to report Chamfer Distance (CD) between 3D reconstruction and ground-truth, **best** results are highlighted. We also highlight the average number of primitives found (#P) in green (smaller than 10) or red (larger than 10). Our performances correspond to a single random run (random) and a run automatically selected among 5 runs using the minimal rendering loss (auto). We augment concurrent methods with a preprocessing step (+ ps) removing the ground from the 3D input.

and thus encourage values that are close binary. Second, another failure mode we observed is one where the planar ground is modeling the entire scene. We avoid this by leveraging a two-stage curriculum learning scheme, where texture maps are downscaled by 8 during the first stage. We empirically validate these two contributions in Section 6.3.3. Optimizing our model on a scene takes 4 hours on a single NVIDIA RTX 2080 Ti GPU. We provide all other implementation details in Appendix D.3.

6.3 Experiments

6.3.1 DTU benchmark

Benchmark details. DTU [Jensen et al., 2014] is an MVS dataset containing 80 forward-facing scenes captured in a controlled indoor setting, where the 3D ground-truth points are obtained through a structured light scanner. We evaluate on 10 scenes (S24, S31, S40, S45, S55, S59, S63, S75, S83, S105) that have different geometries and a 3D decomposition that is relatively intuitive. We use standard processing practices [Yariv et al., 2020, 2021; Darmon et al., 2022], resize the images to 400×300 and run our model with $K = 10$ on all available views for each scene (49 or 64 depending on the scenes). We use the official evaluation presented in Jensen et al. [2014], which computes the Chamfer distance between the ground-truth points and points sampled from the 3D reconstruction, filtered out if not in the neighborhood of the ground-truth points. We use the official implementations to apply EMS [Liu et al., 2022] and MonteboxFinder (MBF) [Ramamonjisoa et al., 2022] on the

ground-truth point clouds and follow the papers' recommendations (normalization, default hyperparameters).

Results. We compare our Chamfer distance performances to state-of-the-art 3D decomposition methods in Table 6.1. For each method, we report the input used and highlight the average number of discovered primitives #P in green (smaller than 10) or red (larger than 10). Intuitively, overly large numbers of primitives lead to less intuitive and manipulative scene representations. Our performances correspond to a single random run (random) and a run automatically selected among 5 runs using the minimal rendering loss (auto). We augment concurrent methods with a preprocessing step (+ ps) using RANSAC to remove the planar ground from the 3D input. Overall, we obtain results that are much more satisfactory than prior works. On the one hand, EMS outputs a reasonable number of primitives but has a high Chamfer distance reflecting bad 3D reconstructions. On the other hand, MBF yields a lower Chamfer distance (even better than ours with the preprocessing step) but it outputs a significantly higher number of primitives, thus reflecting over-decompositions.

Our approach is qualitatively compared to EMS and MBF (augmented with the preprocessing step) in Figure 6.3. Because the point clouds are noisy and incomplete (see 360° renderings in Appendix D.2), EMS and MBF struggle to find reasonable 3D decompositions: EMS misses some important parts, while MBF over-decomposes the 3D into piecewise planar surfaces. On the contrary, our model is able to output meaningful 3D decompositions with varying numbers of primitives and very different shapes. Besides, ours is the only approach that recovers the scene appearance (last column). Also note that it produces a complete 3D scene, despite being only optimized on forward-facing views.

6.3.2 Real-life data and applications

We present qualitative results on real-life captures in Figure 6.4. The first row corresponds to the *Campanile* scene from Nerfstudio repository [Tancik et al., 2023] and the last four rows correspond to BlendedMVS scenes [Yao et al., 2020] that were selected in Yariv et al. [2021]. We adapt their camera conventions to ours and resize the images to roughly 400×300. From left to right, we show a subset of the input views, a rendering overlaid with the primitive edges, the primitives, as well as two novel view synthesis results. For each scene, we run our model 5 times and automatically select the results with the minimal rendering loss. We set the maximum number of primitives to $K = 10$, except the last row where it is increased to $K = 50$ due to the scene complexity. These results show that despite its simplicity, our approach is surprisingly robust. Our method is still able to compute 3D decompositions that capture both appearances

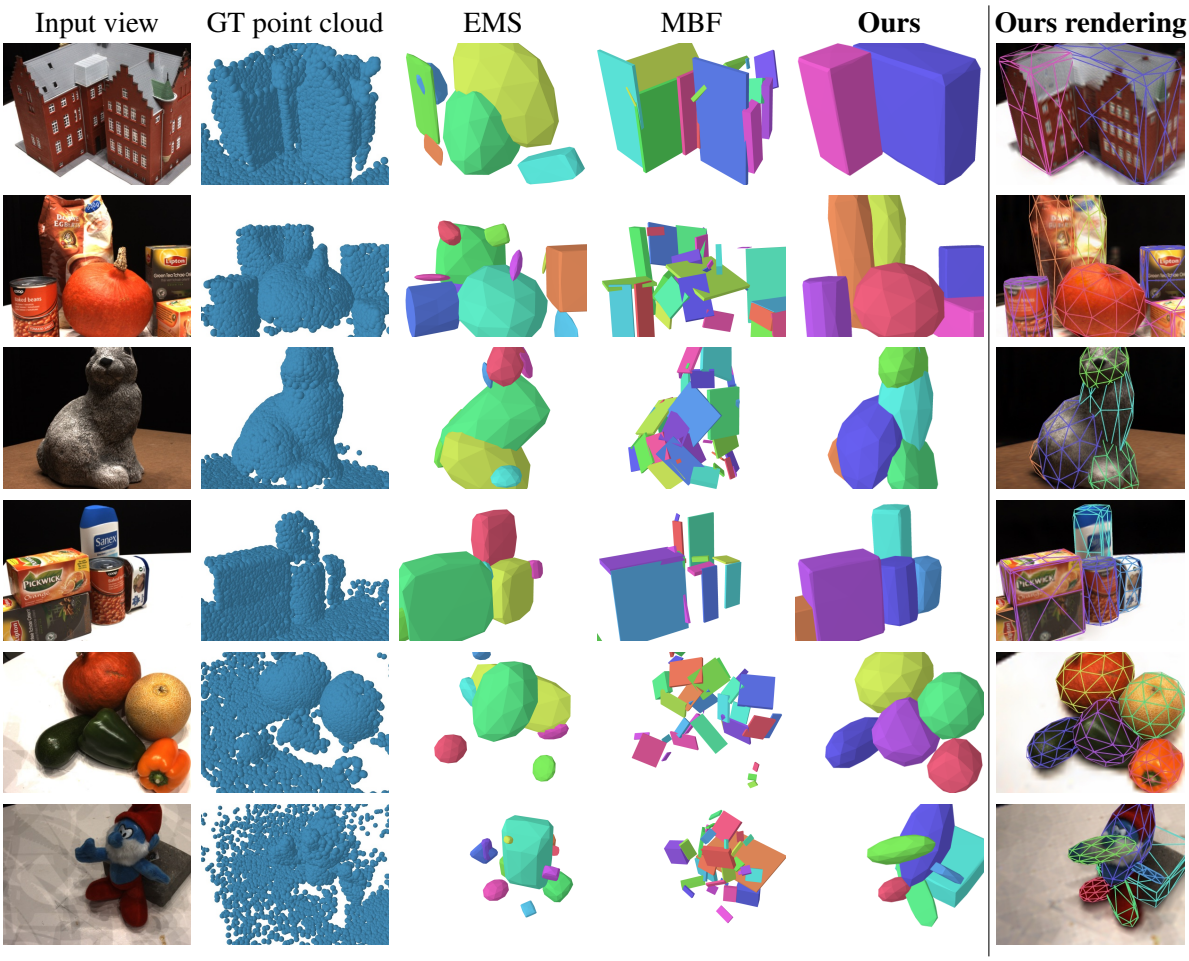


Figure 6.3: **Qualitative comparisons on DTU.** We compare our model to state-of-the-art methods (augmented with a preprocessing step to remove the 3D ground) which, unlike ours, find primitives in the ground-truth point cloud that is noisy and incomplete. Additionally, our approach is the only one able to capture the scene appearance (last column).

and meaningful geometry on a variety of scene types. In addition, increasing the maximum number of primitives K allows us to easily adapt the decomposition granularity (last row).

In Figure 6.5, we demonstrate other advantages of our approach. First, compared to NeRF-based approaches like Nerfacto [Tancik et al., 2023] which only reconstruct visible regions, our method performs amodal scene completion (first row). Second, such a textured decomposition allows to easily edit the 3D scene (second row). Finally, our primitive meshes enable straightforward physics-based simulations (bottom).

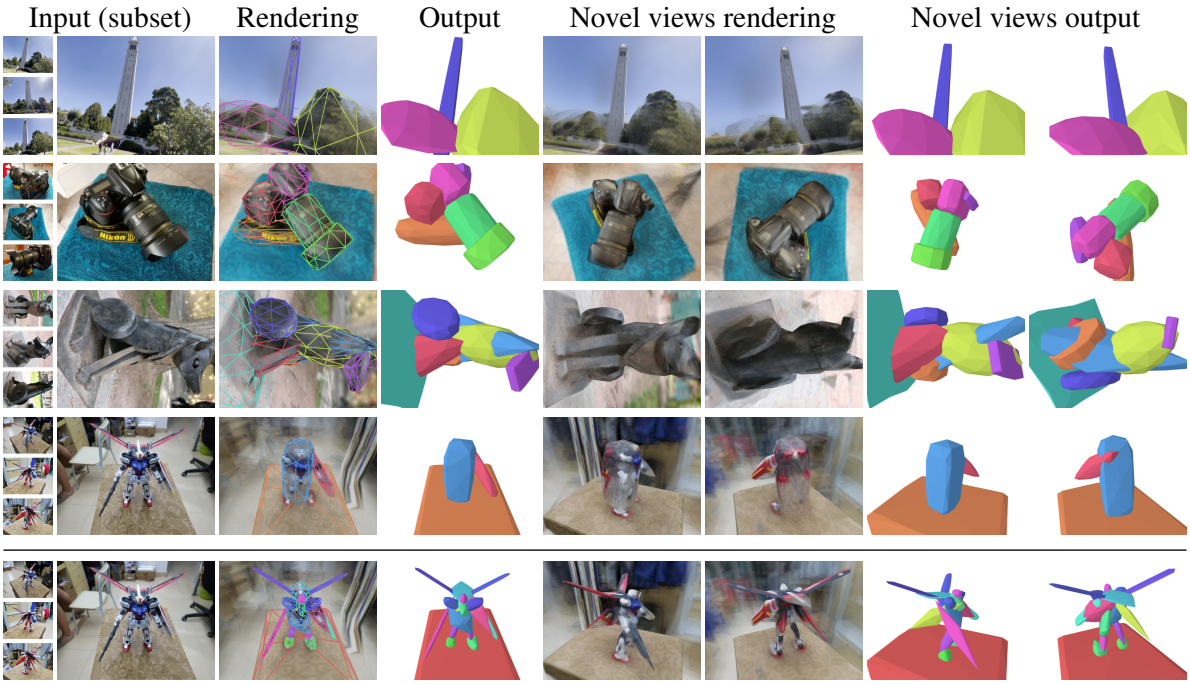


Figure 6.4: **Qualitative results on real-life data.** We run our default model ($K = 10$) on scenes from Nerfstudio [Tancik et al., 2023] (first row) and BlendedMVS [Yao et al., 2020] (all other rows). The last row corresponds to results where the maximum number of primitives is increased to $K = 50$, yielding 17 effective primitives found.

6.3.3 Analysis

Ablation study on DTU. In Table 6.2, we assess our model's key components by removing one component at a time and computing the performance averaged over the 10 DTU scenes. We report the final number of primitives, Chamfer distance and rendering metrics. We highlight the varying number of primitives in green (smaller than 5) and red (larger than 5). Results are averaged over five runs with standard deviations. Overall, each component except $\mathcal{L}_{\text{parsi}}$ consistently improves the quality of the 3D reconstruction and the renderings. $\mathcal{L}_{\text{parsi}}$ successfully limits the number of primitives (and thus, primitive duplication and over-decomposition) at a very small quality cost.

Limitations. Despite its simplicity and robustness, our approach has some limitations. First, our optimization is still prone to bad local minima. Although our automatic selection among several runs is effective, introducing data-driven priors to overcome such local minima would be an interesting future direction. Second, our texturing model does not adapt to the scene geometry thus yielding efficiency and resolution issues. Indeed, large regions in the renderings

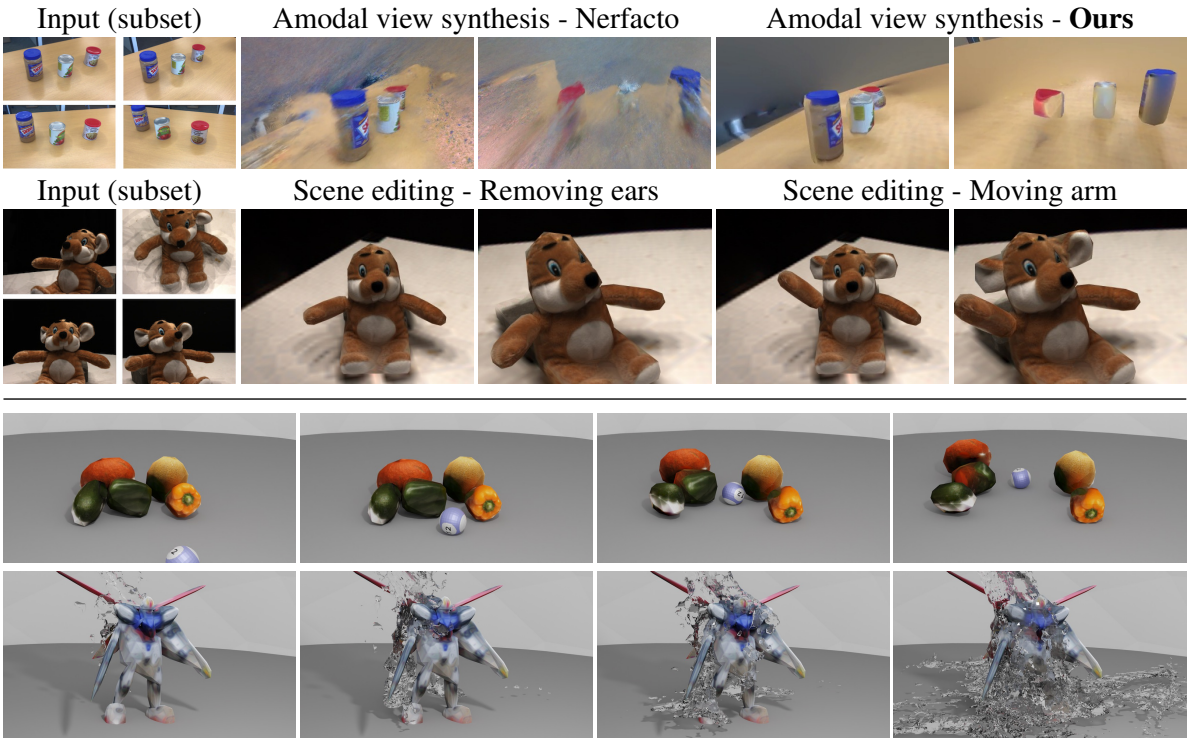


Figure 6.5: **Applications.** Amodal completion (1st row), scene editing (2nd) and physics-based simulations like throwing a ball or pouring water (bottom).

do not necessarily correspond to large regions in the texture images. Finally, our approach does not model lighting or dynamic objects.

6.4 Conclusion

We present an end-to-end approach that successfully computes a primitive-based 3D reconstruction given a set of calibrated images. We show its applicability and robustness through various benchmarks, where our approach obtains better performances than methods leveraging 3D data. We believe our work could be an important step towards more interpretable multi-view modeling.

Method	#P ↓	CD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Complete model	4.60 ± 0.23	$\mathbf{3.63 \pm 0.23}$	$\underline{20.5 \pm 0.2}$	$\underline{73.5 \pm 0.6}$	$\underline{23.9 \pm 0.5}$
w/o $\mathcal{L}_{\text{parsi}}$	8.86 ± 0.27	$\underline{3.65 \pm 0.22}$	$\mathbf{20.6 \pm 0.1}$	$\mathbf{73.7 \pm 0.4}$	$\mathbf{23.2 \pm 0.4}$
w/o $\mathcal{L}_{\text{over}}$	4.38 ± 0.19	3.80 ± 0.30	20.4 ± 0.3	73.2 ± 0.7	24.1 ± 0.7
w/o curriculum	4.66 ± 0.30	3.99 ± 0.17	20.4 ± 0.2	72.7 ± 0.5	24.5 ± 0.4
w/o noise in $\alpha_{1:K}$	$\underline{3.60 \pm 0.21}$	4.13 ± 0.28	20.0 ± 0.2	72.0 ± 0.6	25.6 ± 0.6
w/o $\mathcal{L}_{\text{TV}}$	4.04 ± 0.18	4.58 ± 0.42	19.7 ± 0.3	70.8 ± 1.3	26.5 ± 1.2
w/o $\mathcal{L}_{\text{perc}}$	$\mathbf{3.22 \pm 0.17}$	4.80 ± 0.20	19.7 ± 0.1	72.7 ± 0.3	40.0 ± 0.4

Table 6.2: **Ablation study on DTU.** We report metrics averaged over five runs: number of primitives (#P), Chamfer Distance (CD) and rendering metrics (PSNR in dB and SSIM, LPIPS in %). **Best** and <u>second best</u> are highlighted, #P variability is emphasized in green (smaller than 5) and red (larger than 5).

Chapter 7

Conclusion

In conclusion, we summarize the contributions presented in this manuscript and present future works related to our contributions to unsupervised image analysis.

7.1 Contributions summary

We have presented three key contributions and the effectiveness of each contribution has been demonstrated for at least two different image analysis problems, typically related to both 2D and 3D perception.

Object class modeling with prototypes and deep transformations. We have introduced a new object class modeling approach which represents the class of an image, a 2D object or a 3D shape, with a prototype that is transformed using deep learning to model the different instances. This conceptual idea has first been presented in Chapter 3 in the context of image clustering and it has allowed us to turn a failing K-means algorithm into a state-of-the-art method performing clustering in an interpretable manner. In particular, compared to feature-based analysis, our synthesis-based method enables the end user to understand how and why images are grouped together, by simply visualizing the prototypes and the predicted transformations. Then, we have demonstrated the applicability of this idea to 3D shape modeling in Chapter 5 by computing prototypes and transformations in 3D. Specifically, it has allowed us to design the most unsupervised SVR system at the date of the work publication and to demonstrate results on par with more supervised approaches for diverse datasets.

Discovery by composition of learnable elements. We have proposed a new way to discover elements in images by representing an image collection with a set of learnable elements composed together to synthesize the images and updated by gradient descent. This idea has

first been introduced in Chapter 4 to discover recurrent objects in the image collection. In this chapter, learnable elements are modeled as a small dictionary of sprites which are alpha composited together to synthesize the images. We have not only demonstrated results on par with the best concurrent methods generating images with neural networks, but we have also emphasized how our approach provides much more interpretability. Then, this idea has been applied in Chapter 6 to discover 3D elements from a collection of multi-view images depicting a scene. In this chapter, learnable elements are modeled as textured primitive meshes which are composed together to generate the final scene. We have demonstrated the superiority of our rendering-based approach over prior state-of-the-art methods fitting primitives in 3D point clouds, both in terms of reconstruction performances and applicability to real-world scenarios.

Advances in mesh-based differentiable rendering. We have presented a more technical contribution related a new differentiable mesh rendering which can be formulated as the alpha compositing of the mesh faces in an increasing depth order. We have developed this formulation in Chapter 5 and it has enabled us to learn an SVR model without silhouette annotations, which has not been demonstrated before by prior mesh-based works. Besides, we have demonstrated the applicability of such a formulation in Chapter 6 by introducing the possibility to optimize meshes that are transparent in a seamless manner. In particular, it has allowed us to design a model of a scene as a composition of a variable number of meshes.

7.2 Future works

Our contributions to unsupervised image analysis open interesting research directions that remain unsolved. We discuss three examples of future works based on our contributions.

Class modeling - Application to other modalities. Our object class modeling formulated in Chapter 3 and based on prototypes and deep transformations is a conceptual approach that can be applied to other modalities. We have introduced the general formulation in the context of image modeling and its effectiveness has been demonstrated to learn 3D meshes represented by images from the same object class. Yet, because such a framework does not rely on strong assumptions on the input data, we believe it can be applied to other data modalities, where samples of a same class can be approximated with meaningful transformations of a prototypical instance. In particular, the works of Loiseau et al. [2021, 2022b]; Vincent et al. [2023] successfully apply this idea to 3D data, audio signal and satellite time series respectively.

Discovering object models in the wild. In Chapter 4, we have demonstrated that a 2D modeling of objects can effectively be used to discover recurrent objects in toy datasets like CLEVR and Tetrominoes. Even though we showcased results on real photographs from Instagram collections, our 2D modeling only works for concepts that can be well approximated by a prototypical image that is transformed, like the façade of a building. Hence, our approach would fail to model common 3D objects presenting (i) strong geometry variations like chairs or (ii) deformable parts like horses. Incorporating an explicit 3D object model as we have proposed in Chapter 5 is one attempt to accurately model the 3D variations of such objects. However, our approach works for particular image collections, where images represent a single centered object reflecting the same object class. Although these ideas could be conceptually extended to discover object models from unfiltered real-world image collections, modeling a variable number of 3D objects per image poses clear optimization and computational issues. In a sense, these issues were partially handled in our work in Chapter 6, which studies a much simpler setting where we assume the access to multi-view images and where 3D objects are modeled as semantic-free geometric primitives.

Another orthogonal but promising way to discover object models in the wild is the paradigm of *object-centric representation learning*, which aims at decomposing an image into a set of latent features, each implicitly associated to an object in the image. Ideally, objects reflecting the same concept would be close to each other in the feature space. However, most works in this direction like SlotAttention showcase results for synthetic images where object instances are visually separated in a given image, but they typically do not study if the discovered objects can be separated into meaningful concepts. This raises the question whether the model does not simply learn to localize salient elements, without any recognition ability. Beyond its analytic usefulness, being able to identify objects should actually help the model localize them: visually separating a human riding bike seems much easier when the concepts of human and bike are known. In addition, despite efforts from the most advanced works like Sharma et al. [2023], all these approaches are still limited to synthetically generated images, and the reasons they fail on real-world image collections are yet to be determined.

Reconstructing actionable complex 3D scenes. The approach we have proposed in Chapter 6 is a first step towards computing actionable 3D reconstructions, where 3D components are identified and can be manipulated independently. Despite the variety of results we have showcased, our approach mostly works for objects that are centered and isolated and it would typically fail to capture complex yet common environments like a living room or an outdoor street scene. One solution would be to initialize a much larger number of primitives - *i.e.*, from tens to several thousands primitives - which poses two critical issues in our current approach.

First, it raises memory issues in particular linked with the optimization of a texture map for each primitive. Second and most importantly, this assumption is orthogonal with the idea of computing a parsimonious representation of a scene, where the number of 3D components found is minimal. In a sense, this version of our approach can be related to the recent 3D Gaussian Splatting work from Kerbl et al. [2023] (where ellipsoidal primitive meshes with spherical harmonics are optimized instead of superquadric primitive meshes with texture maps) and one could expect similar non-parsimonious decomposition results. For example, a raw plane would likely be represented with hundreds of tiny primitives. Computing a 3D reconstruction with the rendering fidelity of Kerbl et al. [2023] while maintaining a parsimonious and interpretable representation is an interesting open direction.

Appendices

A Deep Transformation-Invariant Clustering

In this appendix, we provide details about the datasets (Appendix A.1), proof of transformation invariance (Appendix A.2), as well as additional insights about our model (Appendix A.3).

A.1 Dataset descriptions

Table A.1 summarizes details about the following datasets:

- **MNIST** and **MNIST-test** [LeCun et al., 1998] which respectively correspond to full and test subset of MNIST dataset. They depict binary white handwritten digits centered over a black background.
- **USPS** [Hastie et al., 2001] is a handwritten digit dataset from USPS composed of greyscale images.
- **Fashion-MNIST** [Xiao et al., 2017] is a 10-class clothing dataset composed of greyscale images of cloth over black background. Classes are: T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot.
- **FRGC** [Phillips et al., 2005] is a colored face dataset. We use a subset of this dataset introduced in [Yang et al., 2016], where 20 subjects are selected and each image is cropped and resized to a constant size of 32×32.
- **SVHN** [Netzer et al., 2011] is composed of digits extracted from house numbers cropped from Google Street View images. Following standard practice for clustering, we use both labeled samples (99,289) and unlabeled extra samples (~530k) for training and evaluate on the labeled subset only.
- **affNIST-test** is the test split of affNIST (https://www.cs.toronto.edu/ tijmen/affNIST/) an augmented dataset of MNIST where random affine transformations are applied.
- **MNIST-1k**: we randomly sampled 1,000 images from the test split of MNIST.
- **MNIST-color**: we augmented MNIST with random colors for background and foreground.

Dataset	Samples	Classes	Dimension	Transformation sequence
Standard				
MNIST	70,000	10	1×28×28	aff-morpho-tps
MNIST-test	10,000	10	1×28×28	aff-morpho-tps
USPS	9,298	10	1×16×16	col-aff-tps
Fashion-MNIST	70,000	10	1×28×28	col-aff-tps
FRGC	2,462	20	3×32×32	col-aff-tps
SVHN	99,289 + unlabeled extra	10	3×28×28	col-proj
Augmented				
MNIST-1k	1,000	10	1×28×28	aff-morpho-tps
MNIST-color	70,000	10	3×28×28	col-aff-tps
affNIST-test	320,000	10	1×40×40	aff-morpho-tps
Real photographs				
All	1k to 15k	-	3×128×128	col-proj

Table A.1: **Datasets and transformation sequences used**

A.2 Transformation invariance

We consider N image samples $x_{1:N}$, K prototypes $c_{1:K}$ and a group of parametric transformations $\{\mathcal{T}_\beta,\ \beta \in B\}$. For $\beta_1, \beta_2 \in B$, we write $\beta_1\beta_2 \in B$ the element such that $\mathcal{T}_{\beta_1\beta_2} = \mathcal{T}_{\beta_1} \circ \mathcal{T}_{\beta_2}$. We have, for any $\alpha_1, \ldots, \alpha_K \in B$:

$$\mathcal{L}_{\text{TI}}(\{c_1, \ldots, c_K\}) = \mathcal{L}_{\text{TI}}(\{\mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\alpha_K}(c_K)\}).$$

Indeed:

$$\begin{aligned}
\mathcal{L}_{\text{TI}}(\{\mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\alpha_K}(c_K)\}) &= \sum_{i=1}^{N} \min_{\{\beta_1,\ldots,\beta_K\}\in B^K} l(x_i,\ \{\mathcal{T}_{\beta_1} \circ \mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\beta_K} \circ \mathcal{T}_{\alpha_K}(c_K)\}) \\
&= \sum_{i=1}^{N} \min_{\{\beta_1,\ldots,\beta_K\}\in B^K} l(x_i,\ \{\mathcal{T}_{\beta_1\alpha_1}(c_1), \ldots, \mathcal{T}_{\beta_K\alpha_K}(c_K)\}) \\
&= \sum_{i=1}^{N} \min_{\{\beta'_1,\ldots,\beta'_K\}\in B^K} l(x_i,\ \{\mathcal{T}_{\beta'_1}(c_1), \ldots, \mathcal{T}_{\beta'_K}(c_K)\}) \\
&= \mathcal{L}_{\text{TI}}(\{c_1, \ldots, c_K\}),
\end{aligned}$$

using the variable change $\beta'_k = \beta_k\alpha_k$, which is possible because for any $\alpha \in B$, $\alpha B = B$ as we assumed to have a group of transformations.

In some specific cases, the loss is also invariant to the samples, in particular when the loss l is invariant to joint transformation of the prototype and the samples, i.e. for any $\beta \in B$, $l(x_i, \{c_1, \ldots, c_K\} = l(\mathcal{T}_\beta(x_i), \{\mathcal{T}_\beta(c_1), \ldots, \mathcal{T}_\beta(c_K)\}$. This is the case for example for K-means

			MNIST		MNIST-test		USPS		F-MNIST		FRGC		SVHN
Method	Runs	Stat	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
DTI K-means	10	avg	97.3	94.0	96.6	94.6	86.4	88.2	61.2	63.7	39.6	48.7	36.4 / 44.5*
	10	std	0.1	0.1	4.1	1.5	4.1	1.6	2.0	0.3	1.7	2.2	1.9 / 9.6*
	10	min	97.1	93.8	84.9	90.4	83.2	87.1	57.4	63.2	35.9	43.9	34.5 / 37.0*
	10	median	97.3	94.0	97.9	95.1	85.0	87.4	61.9	63.3	40.2	49.3	35.8 / 39.6*
	10	max	97.5	94.2	98.0	95.3	96.4	92.0	63.3	64.2	41.1	51.4	39.6 / 62.6*
	10	minLoss	97.2	93.8	98.0	95.3	89.8	89.5	57.4	64.1	41.1	49.7	39.6 / 62.6*
DTI GMM	10	avg	95.9	93.2	97.8	94.7	84.5	87.2	59.6	62.2	40.1	48.9	36.7 / 57.4*
	10	std	3.9	1.5	0.1	0.2	2.0	0.8	4.7	2.4	1.4	1.5	2.3 / 5.1*
	10	min	84.7	89.1	97.7	94.4	82.0	86.3	56.1	59.7	38.4	46.8	34.0 / 49.9*
	10	median	97.1	93.7	97.8	94.7	84.3	87.1	57.2	60.9	39.6	49.1	36.4 / 57.4*
	10	max	97.3	93.9	98.0	95.1	87.3	89.0	68.2	66.3	41.9	51.1	39.5 / 64.6*
	10	minLoss	97.1	93.7	98.0	95.1	87.3	89.0	68.2	66.3	41.6	51.1	39.5 / 63.3*

Table A.2: **Detailed quantitative results.** We report statistics of our results on standard clustering benchmarks. For SVHN, we also report results with our Gaussian weighted loss ($\star$).

with a group of isometric transformations (e.g. rigid transformations), and it is also the case for GMM with the group of affine transformations applied to both the mean and covariance mixture parameters.

Note that we also tried to transform the samples to match the prototypes, which would lead to an invariance to sample transformation. However, a trivial solution to corresponding optimization problem is to learn "empty" prototypes and transformations of the samples into empty images. For examples, for the MNIST case with affine transformations, we observed that completely black prototypes were learned and any digit was transformed into a black image. Although a regularization term could have prevented such behaviour, we argue that keeping raw samples as target and transforming the prototypes is simpler and effective.

A.3 Model insights

Detailed quantitative results. We report detailed quantitative results for standard clustering benchmarks in Table A.2.

Accuracy and loss correlation. Similar to standard K-means and GMM, there is a variation in performances depending on the random initialization. We experimentally found that: (i) runs seem to be mainly grouped into distinct modes, each corresponding to roughly the same clustering quality; (ii) a run with a low loss usually leads to high clustering performances. We launched 100 runs on MNIST-test dataset and plot the loss with respect to the accuracy for

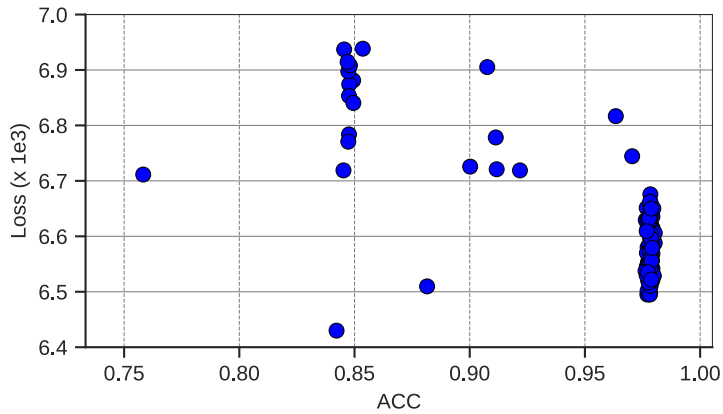
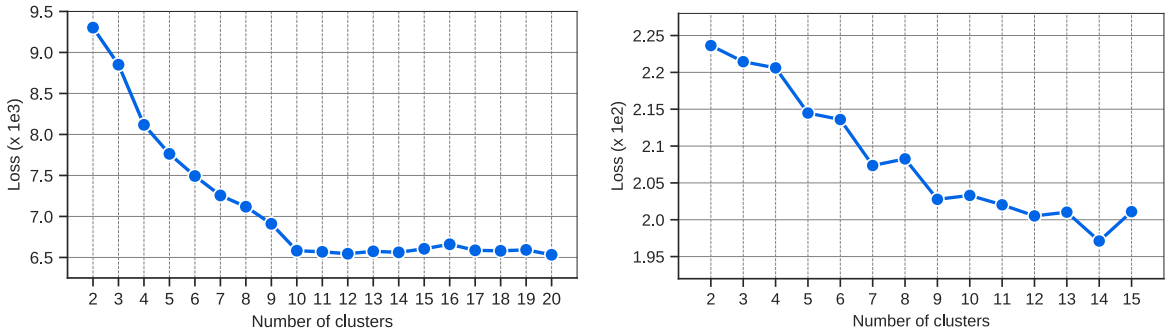


Figure A.1: **Accuracy/loss correlation.** We report loss and accuracy for DTI K-means on MNIST-test.

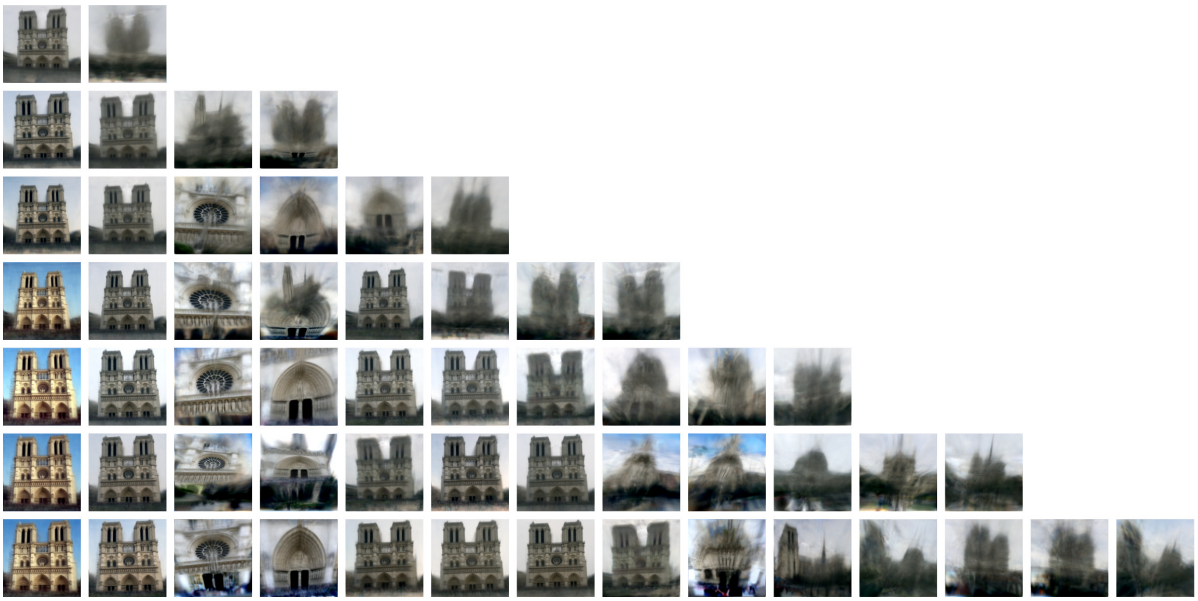
each run in Figure A.1. Except 2 outliers for the 100 runs, the runs with lower loss correspond to the runs with better performances. This is verified in most of our experiments, where the minLoss criterion clearly improves over the average performance.

Effect of the number of clusters K. Similar to many clustering methods, the selection of the number of clusters is a challenge. We investigated if a purely quantitative analysis could be applied to select K. In Figure A.2a, we plot the loss of DTI-Kmeans as a function of the number of clusters for MNIST-test (left) and Notre-Dame (right). For MNIST-test, it is clear an elbow method could be applied to select the good number of clusters. For Notre-Dame, the quantitative analysis is not as conclusive but in this case, the correct number of clusters is not clearly defined. In practice, we did not find the qualitative results on internet photo collections to be very sensitive to this choice, as shown in Figure A.2b where learned prototypes are mostly consistent across the different clustering results.

Constraining color transformation. While evaluating our approaches on real photograph collections, we experimentally observed that a full affine color transformation module (12 parameters) was too flexible and as a result, prototypes were able to learn different patterns hidden in each color channel. In Figure A.3, we show each R, G and B channel as a greyscale image for two prototypes learned using a full affine color transformation module. One can see that a second pattern is hidden in particular in the green channels. To avoid this effect, we restricted the color transformation module to be a diagonal affine transformation corresponding to 6 parameters in total.



(a) Loss w.r.t varying numbers of clusters for MNIST-test (left) and Notre-Dame (right)



(b) Prototypes learned on Notre-Dame for different numbers of clusters

Figure A.2: **Effect of K. (a)** We report the loss of DTI K-means for different numbers of clusters. For MNIST-test (left), the loss is averaged over 5 runs and for Notre-Dame (right), the loss corresponds to a single run. **(b)** We show the prototypes learned on Notre-Dame for even numbers of clusters.

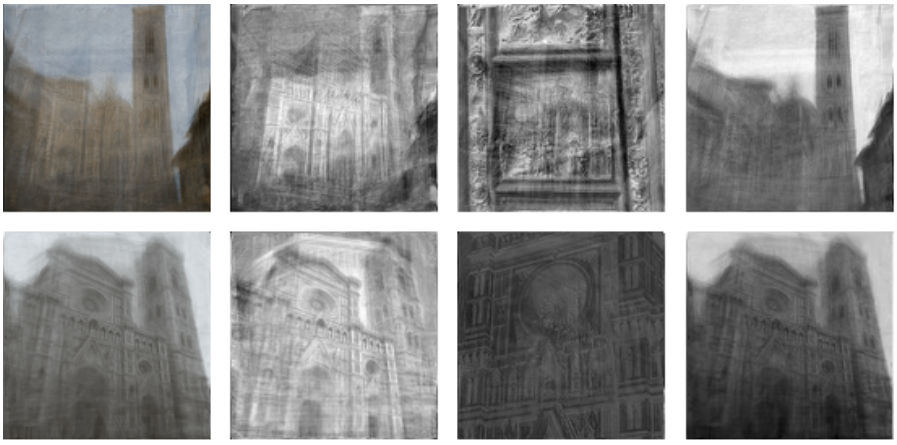


Figure A.3: **Learned prototypes and RGB decomposition.** Two examples of learned prototypes (first column) on Florence cathedral collection from Li and Snavely [2018] using a full color transformation module (12 parameters). The 3 right columns correspond to R, G and B channels rescaled between 0 and 1. Note how the green channel is used to hide a completely different pattern from the other 2 channels.

B Discovering Objects With Sprite Modeling

In this appendix, we provide quantitative semantic segmentation results (Appendix B.1), insights of the model (Appendix B.2), training details (Appendix B.3) and additional qualitative results (Appendix B.4).

B.1 Semantic segmentation results

We provide a quantitative evaluation of our approach in an unsupervised semantic segmentation setting. We do not compare to state-of-the-art approaches as none of them explicitly model nor output categories for objects. We argue that modeling categories for discovered objects is crucial to analyse and understand scenes, and thus advocate such quantitative semantic evaluation to assess the quality of any object-based image decomposition algorithm.

Evaluation. Motivated by standard practices from supervised semantic segmentation and clustering benchmarks, we evaluate our unsupervised object semantic segmentation results by computing the mean accuracy (mACC) and the mean intersection-over-union (mIoU) across all classes (including background). We first compute the global confusion matrix on the same 320 images used for object instance segmentation evaluation. Then, we reorder the matrix with a cluster-to-class mapping computed using the Hungarian algorithm [Kuhn and Yaw, 1955]. Finally, we average accuracy and IoU over all classes, including background, yielding mACC and mIoU.

B.2 Model insights

Effect of K. Similar to standard clustering methods, our results are sensitive to the assumed number of sprites K. A purely quantitative analysis could be applied to select K, *e.g.*, in Figure B.1 we plot the loss as a function of the number of sprites for Tetrominoes and it is clear an elbow method can be applied to correctly select 19 sprites. Qualitatively, using more sprites

Dataset	K	mACC	mIoU
Tetrominoes [Greff et al., 2019]	20	99.5 ± 0.2	99.1 ± 0.4
Multi-dSprites [Kabra et al., 2019]	4	$91.3^{\triangle} \pm 0.9$	$84.0^{\triangle} \pm 1.4$
CLEVR6 [Johnson et al., 2017; Greff et al., 2019]	7	73.9 ± 2.1	56.3 ± 2.9

Table B.1: **Semantic segmentation results.** We evaluate our approach for unsupervised object semantic segmentation results by computing the mean accuracy (mACC) and the mean intersection-over-union (mIoU) across all classes (including background).

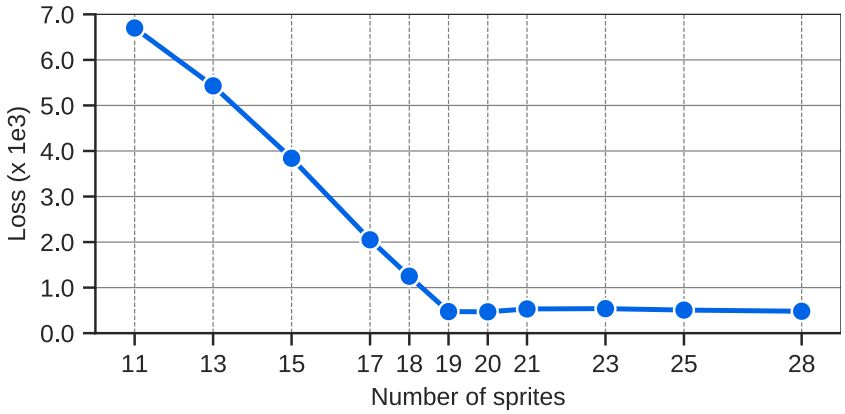


Figure B.1: **Effect of K.** We report the loss obtained for varying number of sprites on Tetrominoes, where the ground-truth number of different shapes is 19.

than the ground truth number typically yields duplicated sprites which we think is not that harmful. For example, we use an arbitrary number of sprites (40) for the Instagram collections and we have not found the discovered sprites to be very sensitive to this choice.

Effect of λ. The hyperparameter λ controls the weight of the regularization term that counts the number of non-empty sprites used. In Figure B.2, we show qualitative results obtained for different values of λ on Multi-dSprites. When λ is zero or small (here $\lambda = 10^{-5}$), the optimization typically falls into bad local minima where multiple layers attempt to reconstruct the same object. Increasing the penalization ($\lambda = 10^{-4}$) prevents this phenomenon by encouraging reconstructions using the minimal number of non-empty sprites. When $\lambda = 10^{-3}$, the penalization is too strong and some objects are typically missed (last example).

Computational cost. Training our method on Tetrominoes, Multi-dSprites and CLEVR6 respectively takes approximately 5 hours, 3 days and 3.5 days on a single Nvidia GeForce RTX 2080 Ti GPU. Our approach is quite memory efficient and for example on CLEVR6, we can use a batch size of up to 128 on a single V100 GPU with 16GB of RAM as opposed to 4 in Greff et al. [2019] and 64 in Locatello et al. [2020].

B.3 Training details

Architecture. We use the same parameter predictor network architecture for all the experiments. It is composed of a shared ResNet [He et al., 2016] backbone truncated after the average pooling and followed by separate Multi-Layer Perceptrons (MLPs) heads predicting sprite transformation parameters for each layer as well as the occlusion matrix. For the

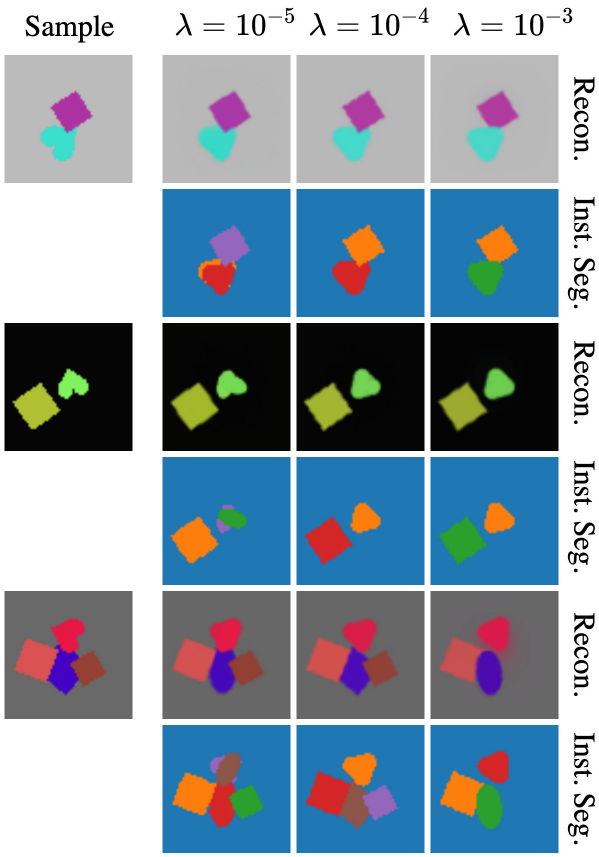


Figure B.2: **Effect of λ.** We show reconstructions and instance segmentations for different values of λ on Multi-dSprites.

ResNet backbone, we use mini ResNet-32[1] (64 features) for images smaller than 65×65 and ResNet-18 (512 features) otherwise. When modeling large numbers of objects (> 3), we increase the representation size by replacing the global average pooling by adaptive ones yielding $4 \times 4 \times 64$ features for mini ResNet-32 and $2 \times 2 \times 512$ for ResNet-18. Each MLP has the same architecture, with two hidden layers of 128 units.

Transformation sequences. Similar to DTI-Clustering [Monnier et al., 2020], we model complex image transformations as a sequence of transformation modules which are successively applied to the sprites. Most of the transformation modules we used are introduced in Monnier et al. [2020], namely affine, projective and TPS modules modeling spatial transformations and a color transformation module. We augment the collection of modules with two additional spatial transformations implemented with spatial transformers [Jaderberg et al., 2015]:

[1] https://github.com/akamaster/pytorch_resnet_cifar10

Dataset	$\mathcal{T}^{\text{lay}}$	$\mathcal{T}^{\text{spr}}$	$\mathcal{T}^{\text{bkg}}$
Tetrominoes [Greff et al., 2019]	col-pos	-	-
Multi-dSprites [Kabra et al., 2019]	col-pos	sim	col
CLEVR6 [Johnson et al., 2017; Greff et al., 2019]	col-pos	proj	col
GTSRB-8 [Stallkamp et al., 2012]	-	col-proj-tps	col-proj-tps
SVHN [Netzer et al., 2011]	-	col-proj-tps	col-proj-tps
Weizmann Horse [Borenstein and Ullman, 2004]	-	col-proj-tps	col-proj-tps
Instagram collections [Monnier et al., 2020]	-	col-proj	col-proj

Table B.2: **Transformation sequences used.**

- a *positioning module* parametrized by a translation vector and a scale value (3 parameters) and used to model coarse layer-wise object positioning,
- a *similarity module* parametrized by a translation vector, a scale value and a rotation angle (4 parameters).

The transformation sequences used for each dataset are given in Table B.2. All transformations for the multi-object benchmarks are selected to mimic the way images were synthetically generated. For real images, we use the col-proj-tps default sequence when the ground truth number of object categories is well defined and the col-proj sequence otherwise. Visualizing sprites and transformations helps understanding the results and adapting the transformations accordingly.

Implementation details. Both sprite parameters and predictors are learned jointly and end-to-end using Adam optimizer [Kingma and Ba, 2015] with a 10^{-6} weight decay on the network parameters. Background, sprite appearances and masks are respectively initialized with averaged images, constant value and Gaussian weights. To constrain sprite parameters in values close to $[0, 1]$, we use a $\mathrm{softclip}$ function implemented as a piecewise linear function yielding identity inside $[0, 1]$ and an affine function with 0.01 slope outside $[0, 1]$. We experimentally found it tends to work better than (i) a traditional clip function which blocks gradients and (ii) a sigmoid function which leads to very small gradients. Similar to Monnier et al. [2020], we adopt a curriculum learning strategy of the transformations by sequentially adding transformation modules during training at a constant learning rate until convergence, then use a multi-step policy by multiplying the learning rate by 0.1 once convergence has been reached. For the experiments with a single object on top of a background, we use an initial learning rate of 10^{-3} and reduce it once. For the multi-object experiments, because spatial transformations are much stronger, we use an initial value of 10^{-4} and first train global layer-wise transformations, using frozen sprites during the first epochs (initialized with constant value for appearances and

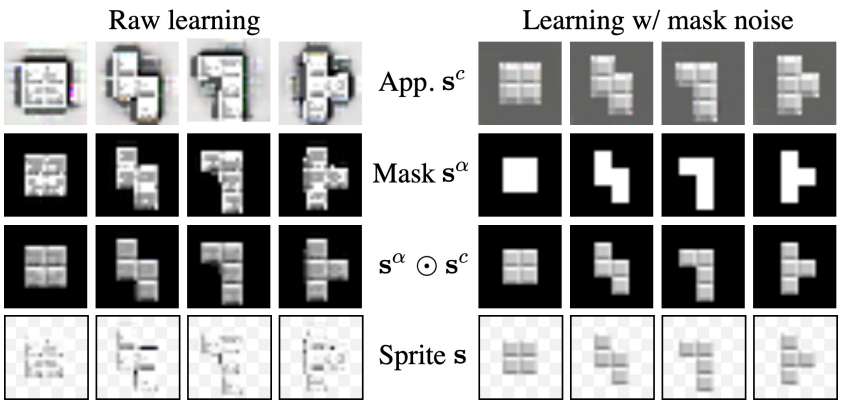


Figure B.3: **Learning binary masks.** We compare results on Tetrominoes obtained with (right) and without (left) injecting noise in masks.

Gaussian weights for masks). Once such transformations are learned, we learn sprite-specific transformations if any and reduce after convergence the learning rate for the network parameters only. Additionally, in a fashion similar to Monnier et al. [2020], we perform sprite and predictor reassignment when corresponding sprite has been used for reconstruction less than $20/K\%$ of the images layers. We use a batch size of 32 for all experiments, except for GTSRB-8 and SVHN where a batch size of 128 is used.

Learning binary masks. There is an ambiguity between learned mask and color values in our raw image formation model. In Figure B.3, we show examples of sprites learned following two settings: (i) a raw learning and (ii) a learning where we constrain mask values to be binary. Although learned appearance images $\mathbf{s}^c$ (first row) and masks $\mathbf{s}^\alpha$ (second row) are completely different, applying the masks onto appearances (third row) yields similar images, and thus similar reconstructions of sample images. However, resulting sprites (last row) demonstrate that the spatial extent of objects is not well defined when learning without any constraint.

Since constraining the masks to binary values actually resolves ambiguity and forces clear layer separations, we follow the strategy adopted by Tieleman [2014] and Kosiorek et al. [2019] to learn binary values, and propose to inject during training uniform noise $\in [-0.4, 0.4]$ into the masks before applying our $\mathrm{softclip}$. Intuitively, such stochasticity prevents the masks from learning appearance aspects and can only be reduced with values close to 0 and 1. We experimentally found this approach tends to work better than (i) explicit regularization penalizing values outside of $\{0, 1\}$ *e.g.* with a $x \to x^2(1-x)^2$ function and (ii) a varying temperature parameter in a sigmoid function as advocated by Concrete/Gumbel-Softmax distributions [Maddison et al., 2017; Jang et al., 2017].

We compare our results obtained with and without injecting noise into the masks on Tetrominoes, where shapes have clear appearances. Quantitatively, while our full model reaches almost a perfect score for both ARI-FG and ARI metrics (resp. 99.6% and 99.8%), these performances averaged over 5 runs are respectively 77.8% and 89.1% when noise is not injected into the masks during learning. We show qualitative comparisons in Figure B.3. Note that the masks learned with noise injection are binary and sharp, whereas the ones learned without noise contain some appearance patterns.

B.4 Additional qualitative results

We provide more qualitative results on the multi-object synthetic benchmarks, namely Tetrominoes (Figure B.4), Multi-dSprites (Figure B.5) and CLEVR6 (Figure B.6). For each dataset, we first show the discovered sprites (at the top), with colored borders to identify them in the semantic segmentation results. We then show 10 random qualitative decompositions. From left to right, each row corresponds to: input sample, reconstruction, semantic segmentation where colors refer to the sprite border colors, instance segmentation where colors correspond to different object instances, and full image decomposition layers where the borders are colored with respect to their instance mask color. Note how we manage to successfully separate the object instances as well as identify their categories and spatial extents.

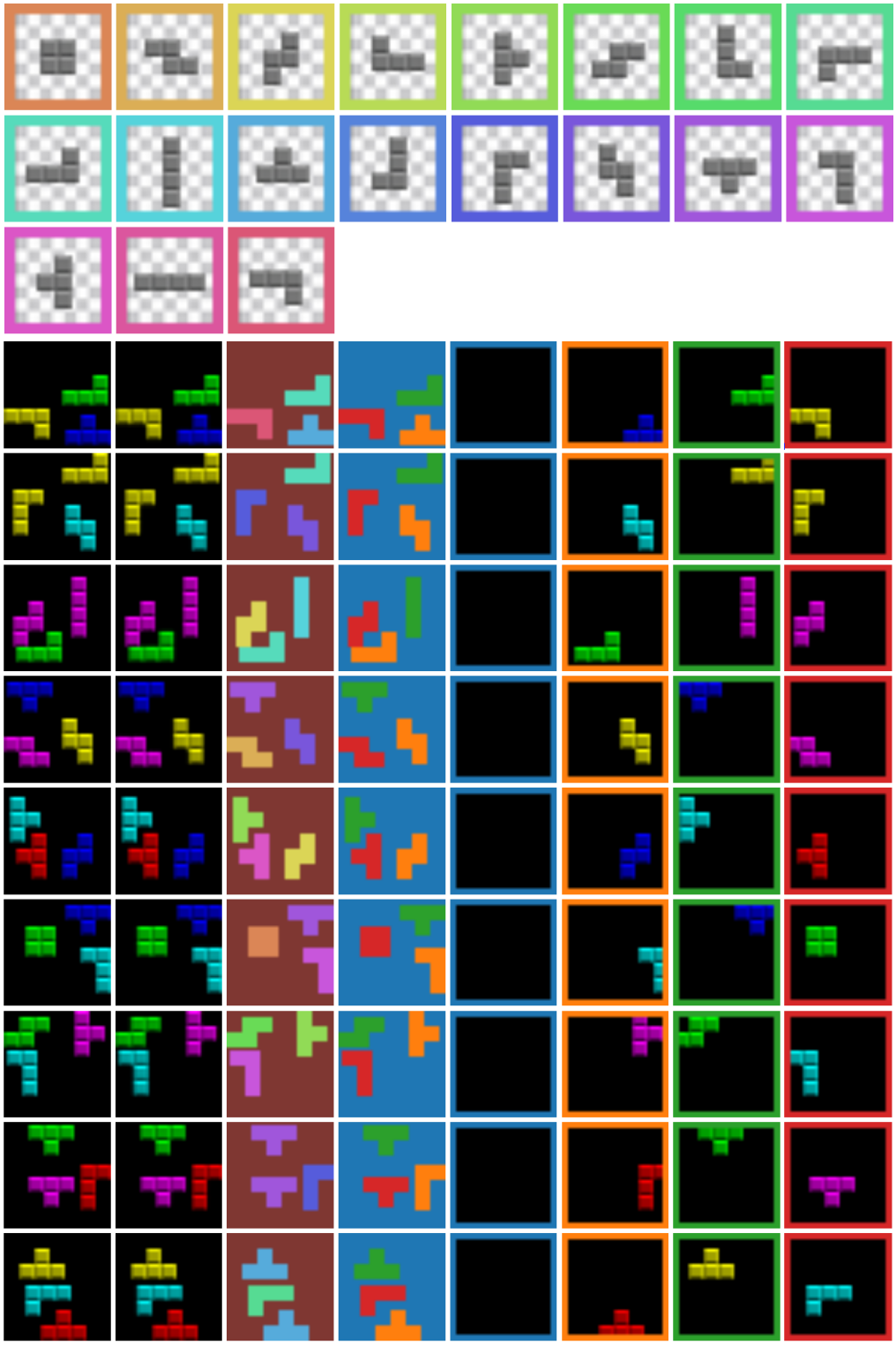


Figure B.4: **Tetrominoes results.** We show discovered sprites (top) and 10 random decomposition results (bottom).

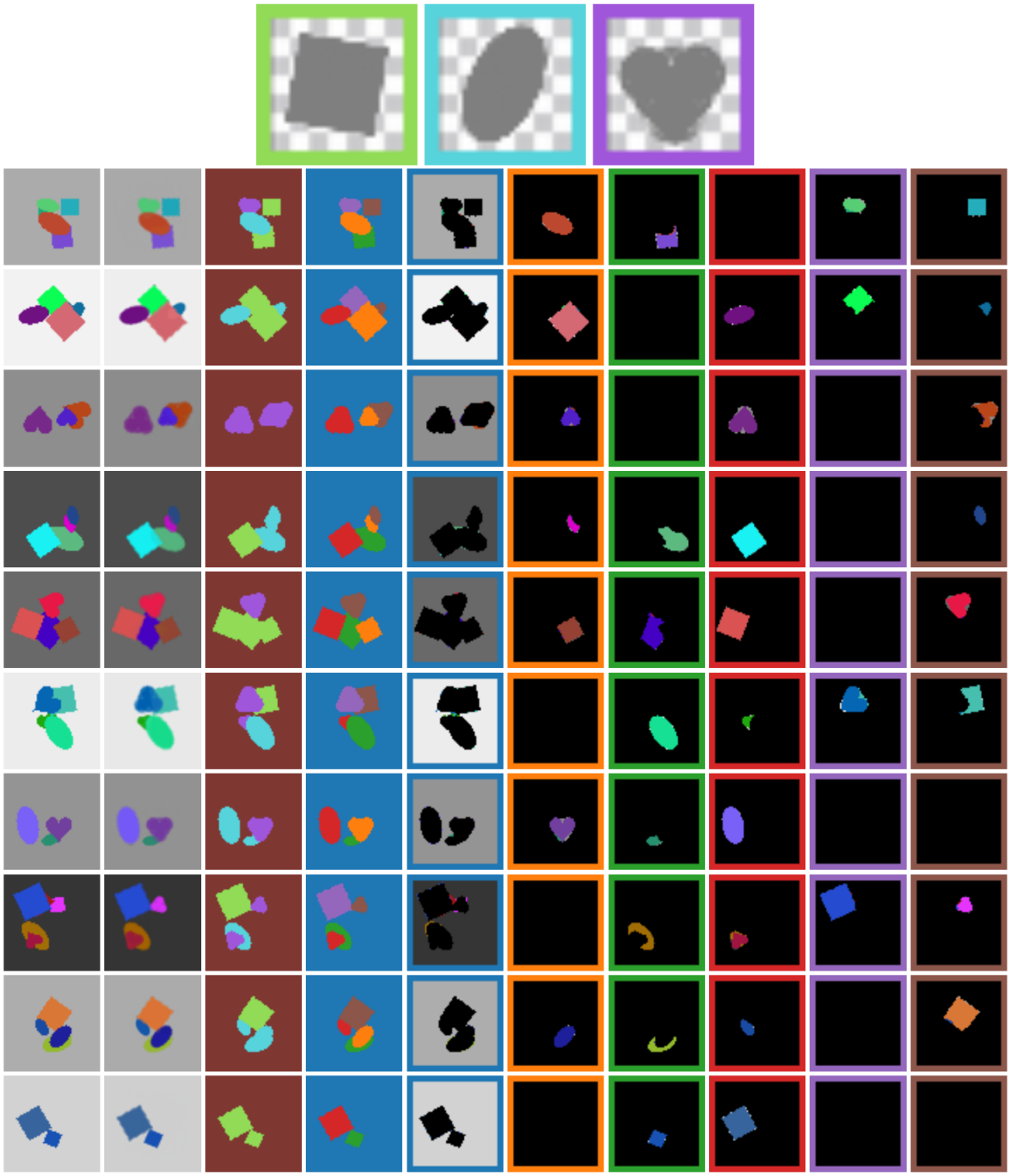


Figure B.5: **Multi-dSprites results.** We show discovered sprites (top) and 10 random decomposition results (bottom).

Figure B.6: **CLEVR6 results.** We show discovered sprites (top) and 10 random decomposition results (bottom).

C Single-View Reconstruction Without Supervision

In this appendix, we present our custom differentiable rendering function (Appendix C.1), insights of our model (Appendix C.2), additional details about the quantitative evaluation (Appendix C.3) and implementation details (Appendix C.4).

C.1 Custom differentiable rendering

Our output images correspond to the soft rasterization of a textured mesh on top of a background image. We observe divergence results when learning geometry from raw photometric comparison with the standard SoftRasterizer [Liu et al., 2019] and thus propose two key changes. In the following, given a mesh $\mathbb{M}$ and a background $\mathbf{B}$, we describe our rendering function $\mathcal{R}$ producing the image $\hat{\mathbf{I}} = \mathcal{R}(\mathbb{M}, \mathbf{B})$. We first present SoftRasterizer formulation and its limitations, then introduce our modifications. In the following, we write pixel-wise multiplication with $\odot$ and the division of image-sized tensors corresponds to pixel-wise division.

SoftRasterizer formulation. Given a 2D pixel location i, the influence of a face j is modeled by an occupancy function:

$$\mathcal{O}_{\text{SR}}(i, j) = \text{sigmoid}\left(\frac{\nu(i, j)}{\sigma}\right), \tag{1}$$

where σ is a temperature, $\nu(i, j)$ is the signed Euclidean distance between pixel i and projected face j. Let us call $L - 1$ the maximum number of faces intersecting the ray associated to a pixel and sort, for each pixel, the intersecting faces by increasing depth. Image-sized maps for occupancy $\mathbf{O}_\ell$, color $\mathbf{C}_\ell$ and depth $\mathbf{D}_\ell$ are built associating to each pixel the ℓ-th intersecting face attributes. Background is modeled as additional maps such that $\mathbf{O}_L = 1, \mathbf{C}_L = \mathbf{B}$, and $\mathbf{D}_L = d_{\text{bg}}$ is a constant, far from the camera. The SoftRasterizer's aggregation function $\mathcal{C}_{\text{SR}}$ merges them to render the final image $\hat{\mathbf{I}}$:

$$\mathcal{C}_{\text{SR}}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}, \mathbf{D}_{1:L}) = \sum_{\ell=1}^{L} \frac{\mathbf{O}_\ell \odot \exp(\mathbf{D}'_\ell/\gamma)}{\sum_k \mathbf{O}_k \odot \exp(\mathbf{D}'_k/\gamma)} \odot \mathbf{C}_\ell, \tag{2}$$

where γ is a temperature parameter, $\mathbf{D}'_\ell = \frac{d_{\text{far}} - \mathbf{D}_\ell}{d_{\text{far}} - d_{\text{near}}}$ and d_{near}, d_{far} correspond to near/far cut-off distances. This formulation hence relies on 5 hyperparameters (σ, γ, d_{near}, d_{far}, d_{bg}) and default values are $\sigma = \gamma = 10^{-4}$, $d_{\text{near}} = 1$, $d_{\text{far}} = 100$ and $\frac{d_{\text{far}} - d_{\text{bg}}}{d_{\text{far}} - d_{\text{near}}} = \epsilon = 10^{-3}$.

The formulation introduced in Equation (2) has one main limitation: gradients don't flow well through $\mathbf{O}_{1:L}$ obtained by soft rasterization, and thus vertex positions cannot be optimized

by raw photometric comparison. The simple case of a single face on a black background gives:

$$\hat{\mathbf{I}} = \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma} + e^{\epsilon/\gamma}} \odot \mathbf{C}_1 \approx \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}} \odot \mathbf{C}_1 = \mathbf{C}_1, \tag{3}$$

for almost all $\mathbf{O}_1, \mathbf{D}'_1$. Indeed, considering $x, \eta > 0$, we have $xe^{(\epsilon+\eta)/\gamma} \gg e^{\epsilon/\gamma}$ iif $x \gg e^{-\eta/\gamma}$. Even in the best case where $\eta = \epsilon = 10^{-3}$ (*i.e.*, the object is close to d_{far}), this holds for all $x \gg e^{-10} \approx 4 \times 10^{-5}$! We found that tuning γ was not sufficient to mitigate the issue, one would have to tune $\gamma, d_{\text{near}}, d_{\text{far}}, d_{\text{bg}}$ simultaneously to enable the optimization of the vertex positions.

Our layered formulation. Inspired by layered image models [Jojic and Frey, 2001; Monnier et al., 2021], we propose to model the rendering of a mesh as the layered composition of its projected face attributes. More specifically, given occupancy $\mathbf{O}_{1:L}$ and color $\mathbf{C}_{1:L}$ maps, we render an image $\hat{\mathbf{I}}$ through the classical recursive alpha compositing:

$$\mathcal{C}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}) = \sum_{\ell=1}^{L} \left[\prod_{k<\ell}^{L} (1 - \mathbf{O}_k) \right] \odot \mathbf{O}_\ell \odot \mathbf{C}_\ell. \tag{4}$$

This formulation has a clear interpretation where color maps are overlaid on top of each other with a transparency corresponding to their occupancy map. Note that we choose to drop the explicit depth dependency as all 3D coordinates (including depth) of a vertex already receive gradients by 3D-to-2D projection. Our layered aggregation used together with the SoftRasterizer's occupancy function $\mathcal{O}_{\text{SR}}$ results in face inner-borders that are visually unpleasant. We thus instead use the occupancy function introduced in Chen et al. [2019b] defined by:

$$\mathcal{O}(i, j) = \exp(\min(0, \frac{\nu(i, j)}{\sigma})). \tag{5}$$

Compared to $\mathcal{O}_{\text{SR}}$, this function yields constant occupancy of 1 inside the faces. In addition to its simplicity, our differential renderer has two main advantages compared to SoftRasterizer. First, gradients can directly flow through occupancies $\mathbf{O}_{1:L}$ and the vertex positions can be updated by photometric comparison. Second, our formulation involves only one hyperparameter (σ) instead of five, making it easier to use.

C.2 Model insights

Progressive conditioning. Figure C.1 shows the results obtained on CompCars [Yang et al., 2015] at the end of each stage of the training. Given an input image (leftmost column), we

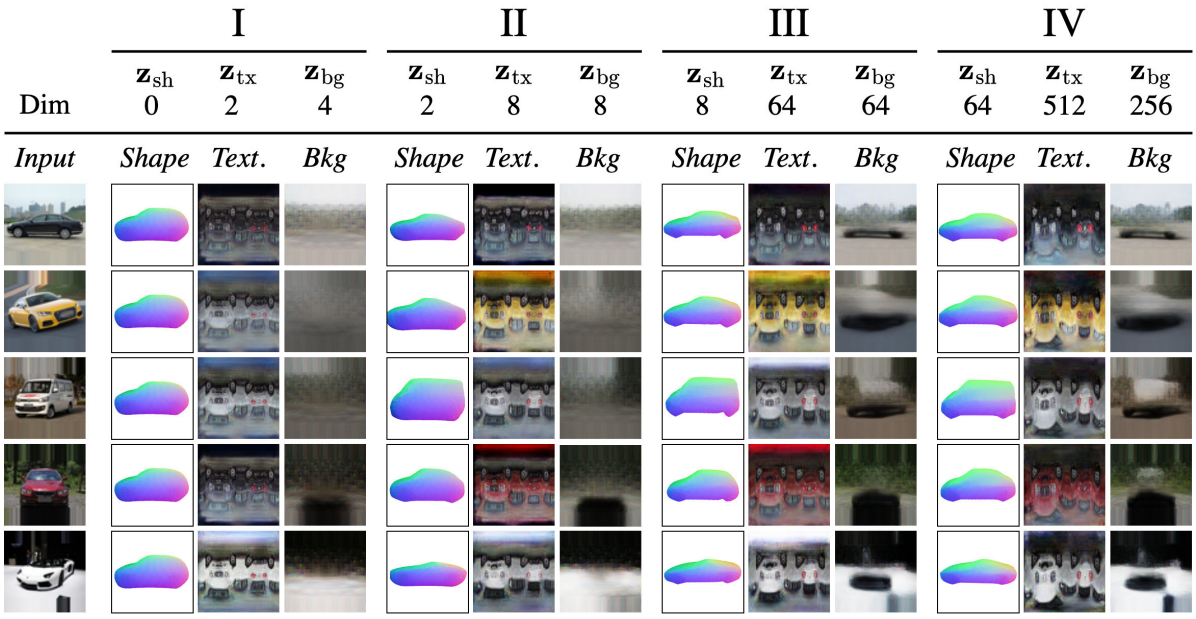
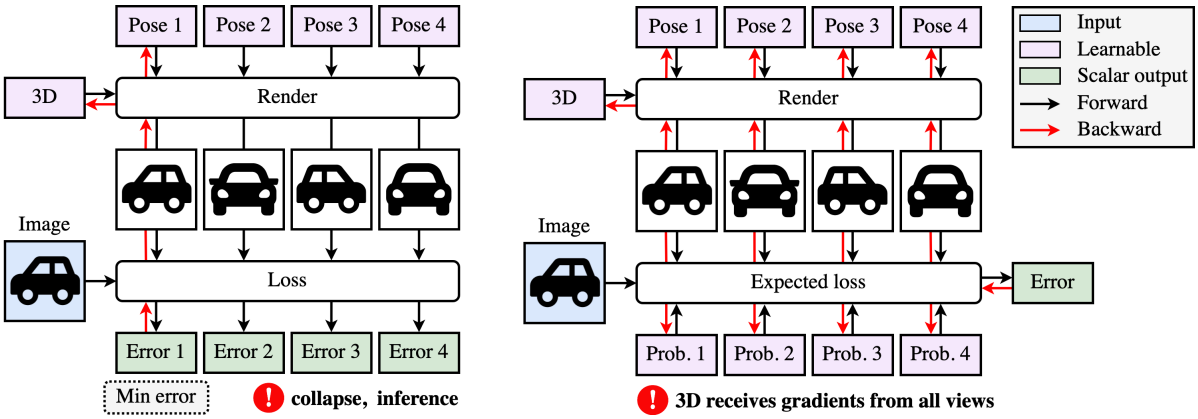


Figure C.1: **Progressive conditioning on CompCars.** Given an input image (leftmost column), we show for each training stage, from left to right, a side view of the predicted shape, the texture image and the background image.

show for each training stage the predicted outputs. From left to right, they correspond to a side view of the shape, the texture image and the background image. We can observe that all shape, texture and background models gradually specialize to the instance represented in the input. In particular, this allows us to start with a weak background model to avoid degenerate solutions and to end up with a powerful background model to improve the reconstruction quality. Also note how all the texture images are aligned.

Neighbor reconstruction. When computing the neighbor reconstructions, we explicitly find neighbors that have a viewpoint different from the predicted viewpoint. More specifically, for a given input, we compute the angle between the predicted rotation matrix and all rotation matrices of the memory bank. Following standard conventions, such an angle lies in $[0°, 180°]$. Then, we select a target angle range as follows: we split the range of angles $[20°, 180°]$ into a partition of V uniform and continuous bins, and we uniformly sample one of the V angle ranges. Finally, we look for neighbors in the subset of instances having an angle within the selected range. In all experiments, we use $V = 5$.

We use a total angle range of $[20°, 180°]$ instead of $[0°, 180°]$ to remove instances having a similar pose. Note that we first tried to find neighbors of different poses without further constraint (which amounts to using $V = 1$) but we found that learned latent codes were specialized by viewpoints, *e.g.*, front / back view images corresponding to a shape mode with



(a) Optimization in Insafutdinov and Dosovitskiy [2018]

(b) Optimization in Henderson and Ferrari [2019]; Goel et al. [2020]; Tulsiani et al. [2020]

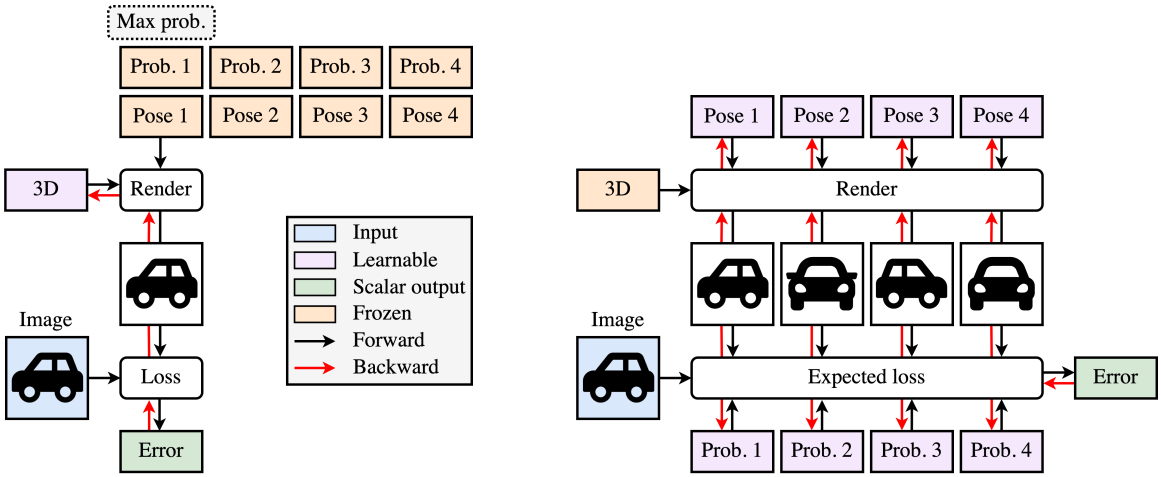
Figure C.2: **Prior optimizations for joint 3D/pose learning.**

unrealistic side views, and side view images corresponding to a shape mode with unrealistic front / back views.

Joint 3D and pose learning. We analyze prior works on joint 3D and pose learning, illustrated in Figure C.2, and compare them with our proposed optimization scheme, illustrated in Figure C.3. Prior optimization schemes can be split in two groups: (i) learning through the minimal error reconstruction [Insafutdinov and Dosovitskiy, 2018], and (ii) learning through an expected error [Henderson and Ferrari, 2019; Goel et al., 2020; Tulsiani et al., 2020].

In Insafutdinov and Dosovitskiy [2018], all reconstructions associated to the different pose candidates are computed and both 3D and poses are updated using the reconstruction yielding the minimal error (see Figure C.2a). We identified two major issues. First, because the other poses are not updated for a given input, we observed that a typical failure case corresponds to a collapse mode where only a single pose (or a small subset of poses) is used for all inputs. Indeed, there is no particular constraint that encourages the use of all pose candidates. Second, inference is not efficient as the object has to be rendered from all poses to find the correct object pose.

In Henderson and Ferrari [2019]; Goel et al. [2020]; Tulsiani et al. [2020], 3D and poses are updated using an expected reconstruction loss (see Figure C.2b). While this allows to constrain the use of all pose candidates with a regularization on the predicted probabilities, we identified one major weakness common to these frameworks. Because the 3D receives gradients from all views, we observed a typical failure case where the 3D tries to fit the target input from all pose candidates yielding inaccurate texture and geometry. We argue such behaviour was not observed in previous works as they typically use a symmetry prior which prevents it from



(a) 3D-step (b) P-step

Figure C.3: **Our alternate 3D / pose optimization.** Compared to prior works, we propose an optimization that alternates between 2 steps. **(a)** We update the 3D using the most likely pose candidate (3D-step). **(b)** We update the pose candidates and associated probabilities using the expected loss (P-step).

happening. Note that CMR [Goel et al., 2020] proposes to directly optimize for each training image a set of parameters corresponding to the pose candidates. This procedure not only involves memory issues as the number of parameters scales linearly with the number of training images, but also inference problems for new images. To mitigate the issue, they propose to use the learned poses as ground-truth to train an additional network that performs pose estimation given a new image.

In contrast, our proposed alternate optimization, illustrated in Figure C.3, leverages the best of both worlds: (i) 3D receives gradients from the most likely reconstruction, and (ii) all poses are updated using an expected loss. In practice, we alternate the optimization every new batch of inputs, and we define one iteration as either a a 3D-step or a P-step.

C.3 Quantitative evaluation

ICP alignment for better 3D evaluation. We align shapes using our gradient-based version of the Iterative Closest Point (ICP) [Besl and McKay, 1992] with anisotropic scaling before evaluating 3D reconstructions. For consistency, we use the same protocol across benchmarks and advocate to do so for future comparisons. First, meshes are centered and normalized so that they exactly fit inside the cube of unit length $[-0.5, 0.5]^3$; this is important to obtain results that are comparable. Second, we sample 100k points on the mesh surfaces. Third, we run our ICP implementation which minimizes by gradient descent the Chamfer-L_2 distance between the point clouds by jointly optimizing 3 translation parameters, 6 rotation parameters [Zhou et al.,

2019] and 3 scaling parameters. In practice, we use Adam optimizer [Kingma and Ba, 2015], a learning rate of 0.01 and 100 iterations. Note that we use this gradient-based version of ICP instead of the classical iterative formulation as we found it to diverge when optimizing scale.

We argue that an ICP pre-processing is crucial for an unbiased 3D reconstruction evaluation and provide real examples in Figure C.4 to support our claim. Rows correspond to different transformations of the same canonical shape, and for each row, we show: the transformation used, the resulting 3D shape, a rendering example as well as Chamfer-L_1 distance to the canonical shape. We overlay the visuals with green contours representing the canonical shape and the canonical rendering for easier comparisons. We can make two important observations. First, although all the transformed shapes are excellent 3D reconstructions of the canonical shape, they result in dramatically poor performances. As a comparison, these performances are similar to our ShapeNet results with ICP when the model outputs degenerate reconstructions. Pre-processing the shapes using an ICP with anisotropic scaling mitigates this issue. Second, as shown by the rendering examples, for all these different shapes, we can find a pose that yields almost identical renderings. This hence emphasizes the numerous shape/pose ambiguities that arise from a given rendered image. As a result, it is extremely unlikely that a fully unsupervised SVR system predicting from a single image both the 3D shape and the pose will recover the exact pair of shape/pose used for annotations. In this case, the cameras used for rendering are the same and we do not even consider focal variations, which raises even more ambiguities.

ShapeNet results without ICP. For completeness, we provide quantitative results obtained without ICP on the ShapeNet benchmark in Table C.1. We indicate the supervision used and visually separate methods using multi-view supervision. We report (i) results from category-agnostic versions (Cat. agn) of DVR [Niemeyer et al., 2020] and SoftRas [Liu et al., 2019] presented in Niemeyer et al. [2020] and (ii) divergence results obtained by removing silhouette supervision from DVR and SoftRas.

C.4 Implementation details

Network architecture. We use the same neural network architecture for all experiments. The encoder is composed of 4 CNN backbones followed by separate Multi-Layer Perceptron (MLP) heads predicting a rendering parameter. More specifically, the 4 backbones are respectively used to predict: (i) shape code $\mathbf{z}_{\text{sh}}$ and scale $\mathbf{s}$; (ii) texture code $\mathbf{z}_{\text{tx}}$; (iii) background code $\mathbf{z}_{\text{bg}}$; (iv) rotations $\mathbf{r}_{1:K}$, translations $\mathbf{t}_{1:K}$, and pose probabilities $\mathbf{p}_{1:K}$. Note that using a shared backbone instead of separated ones also yields great results and is advocated for decreasing the memory footprint and training time; the major benefit from using separated backbones is to produce slightly more detailed textures and background. We follow prior works in

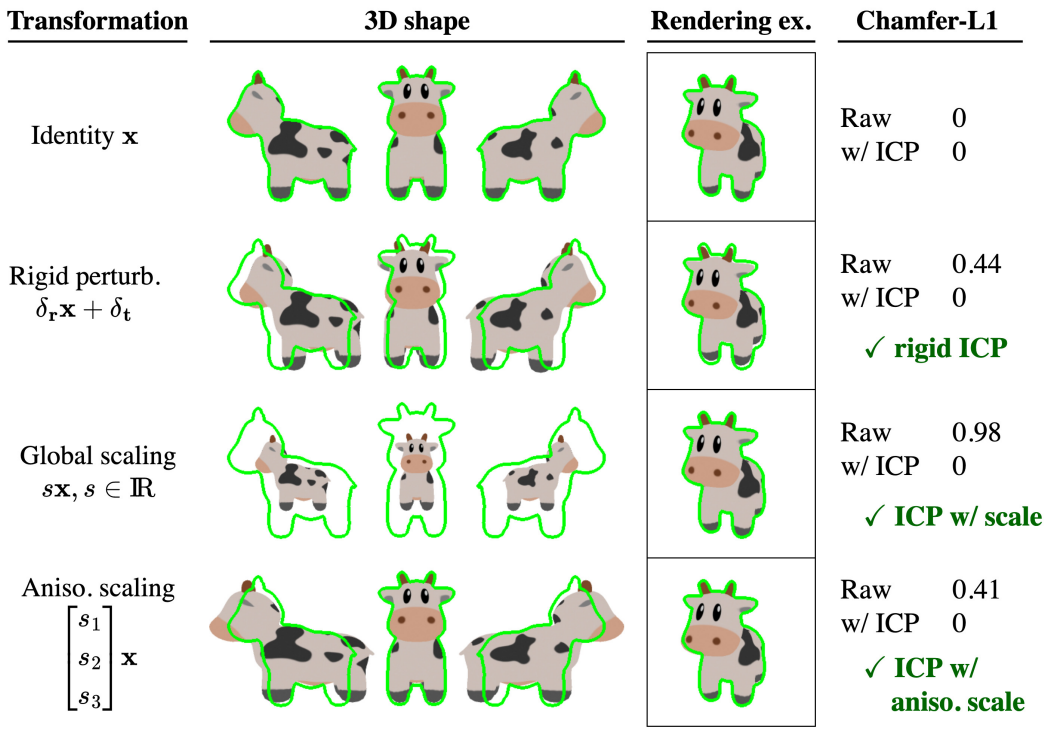


Figure C.4: **3D reconstruction evaluation with and without ICP.** Rows correspond to results obtained for transformed versions of a canonical shape and columns correspond to, from left to right, the transformation used, resulting 3D shape, a rendering example and Chamfer-L_1 distance to the canonical shape. Green contours represent the shape and rendering from the canonical object for visual comparisons.

SVR [Groueix et al., 2018; Mescheder et al., 2019; Niemeyer et al., 2020; Goel et al., 2020] and use randomly initialized ResNet-18 [He et al., 2016] as backbone. Each MLP head has the same architecture with 3 hidden layers of 128 units and ReLU activations. The last layer of the MLP heads for shape, texture and background codes is initialized to zero to avoid discontinuity when increasing the size of the latent codes. The final activation of the MLP heads for scale, rotation, and translation is a $\tanh$ function and the output is scaled and shifted using predefined constants in order to control their range (see Table C.2 for selected ranges). The learnable parts of the decoder are the shape deformation network s_θ and the two CNN generators t_θ and b_θ which respectively output 64×64 images for texture and background. The MLP modeling the deformations has 3 hidden layers, ReLU activations and 128 units for real images; we use an increased number of units for ShapeNet (512) which provides a small boost in performances. The CNN generators share the same architecture which is identical to the generator used in GIRAFFE [Niemeyer and Geiger, 2021]. We refer the reader to Niemeyer and Geiger [2021] for details.

Method	**Ours**	SDF-SRN	DVR	DVR	SoftRas	DVR	DVR	SoftRas
Cat. agn.							✓	✓
MV				✓	✓	✓	✓	✓
CK		✓	✓	✓	✓	✓	✓	✓
S		✓	✓			✓	✓	✓
airplane	0.186	0.173	**0.157**	Div.	Div.	0.151	0.190	**0.149**
bench	**0.257**	-	0.386	Div.	Div.	0.232	**0.210**	0.241
cabinet	**0.284**	-	0.849	Div.	Div.	0.257	**0.220**	0.231
car	0.251	**0.177**	0.282	Div.	Div.	0.198	**0.196**	0.221
chair	0.543	**0.333**	0.464	Div.	Div.	**0.249**	0.264	0.338
display	**0.344**	-	0.968	Div.	Div.	0.281	**0.255**	0.284
lamp	0.987	-	**0.688**	Div.	Div.	0.386	0.413	**0.381**
phone	**0.456**	-	1.412	Div.	Div.	0.147	0.148	**0.131**
rifle	**0.504**	-	0.528	Div.	Div.	**0.131**	0.175	0.155
sofa	**0.335**	-	0.665	Div.	Div.	**0.218**	0.224	0.407
speaker	**0.356**	-	0.535	Div.	Div.	0.321	**0.289**	0.320
table	**0.351**	-	0.442	Div.	Div.	0.283	**0.280**	0.374
vessel	**0.384**	-	0.400	Div.	Div.	**0.220**	0.245	0.233
mean	**0.403**	-	0.598	Div.	Div.	**0.236**	0.239	0.266

Table C.1: **ShapeNet comparison without ICP.** We report Chamfer-L_1 ↓, supervisions are: Multi-Views, Camera or Keypoints, Silhouettes. We separate methods using multi-views and **best** results are highlighted in each group.

Other design choices. In all experiments, the predefined anisotropic scaling used to deform the icosphere into an ellipsoid is $[1, 0.7, 0.7]$. In Table C.2, we detail other design choices that are specific to all categories of ShapeNet [Chang et al., 2015] (second column) or all real-image datasets (third column). This notably includes a predetermined global scaling of the ellipsoid, a camera defined by a focal length f or a field of view (fov), as well as scaling, translation and rotation ranges.

Training. In all experiments, we use a batch size of 32 images of size 64×64 and Adam optimizer [Kingma and Ba, 2015] with a constant learning rate of 10^{-4} that is divided by 5 at the very end of the training for a few epochs. The training corresponds to 4 stages where latent code dimensions are increased at the beginning of each stage and the network is then trained until convergence. We use dimensions 0/2/8/64 for the shape code, 2/8/64/512 for the texture code, and 4/8/64/256 for the background code if any. In line with the curriculum modeling of Monnier et al. [2020], we found it beneficial for the first stage to gradually increase the model complexity: we first learn to position the fixed ellipsoid in the image, then we allow

Design type	ShapeNet	Real-image
ellipsoid scale	0.4	0.6
camera	$f = 3.732$	fov $= 30°$
$\mathbf{s}_x/\mathbf{s}_y/\mathbf{s}_z$	1 ± 0.5	1 ± 0.3
$\mathbf{t}_x/\mathbf{t}_y$	0 ± 0.5	0 ± 0.3
$\mathbf{t}_z$ (depth)	2.732	2.732 ± 0.3
$\mathbf{r}_a$ (azimuth)	$[0°, 360°]$	$[0°, 360°]$
$\mathbf{r}_e$ (elevation)	$30°$	$[-10°, 30°]$
$\mathbf{r}_r$ (roll)	$0°$	$[-30°, 30°]$

Table C.2: **Design choices.** Following standard practices [Liu et al., 2019; Niemeyer et al., 2020] on ShapeNet [Chang et al., 2015], we keep the default rendering values used to generate the images for the focal length f, the distance to the camera $\mathbf{t}_z$ and the elevation $\mathbf{r}_e$. For real images, we keep the classical value of 2.732 for the distance to the camera $\mathbf{t}_z$ and use a field of view (fov) of $30°$. Note that we did not finetune these parameters, they were selected once through visual comparisons on a toy example.

the ellipsoid to be deformed, and finally we allow scale variabilities. In particular, we found this procedure prevents the model to learn prototypical shapes with unrealistic proportions. In the following, we describe other training details specific to ShapeNet [Chang et al., 2015] benchmark and real-image datasets.

ShapeNet dataset. We use the same training strategy for all categories. We train the first stage for 50k iterations, and each of the other stage for 250k iterations, where one iteration corresponds to either a 3D-step or a P-step of our alternate optimization. We do not learn a background model as all images are rendered on top of a white background. However, we found that our system learned in such synthetic setting was prone to a bad local minimum where the predicted textures have white regions that accommodate for wrong shape prediction. Intuitively, this is expected as the system has no particular signal to distinguish white background regions from white object parts. To mitigate the issue, we constrain our texture model as follows: (i) during the first stage, the predicted texture image is averaged to yield a constant texture, and (ii) during the other stages, we occasionally use averaged textures instead of the real ones. More specifically, we sample a Bernoulli variable with probability $p = 0.2$ at each iteration and average the predicted texture image in case of success. We found this simple procedure to work well to resolve such shape/texture ambiguity.

Real-image datasets. We use the same training strategy for all real-image datasets. We train each stage for roughly 750k iterations, where one iteration either corresponds to a 3D-step or a P-step of our alternate optimization. Learning our structured autoencoder in such

real-image scenario, without silhouette nor symmetry constraints, is very challenging. We found our system sometimes falls into a bad local minimum where the texture model is specialized by viewpoints, *e.g.*, dark cars always correspond to a frontal view and light cars always correspond to a back view. To alleviate the issue, we encourage uniform textures by occasionally using averaged textures instead of the real ones during rendering, as done on the ShapeNet benchmark. More specifically, we sample a Bernoulli variable with probability $p = 0.2$ at each iteration and average the predicted texture image in case of success. We observed that it was very effective in practice, and we also found it helped preventing the object texture from modeling background regions. We do not use such technique in the last stage to increase the texture accuracy.

D Differentiable Blocks World

In this appendix, we present additional video results (Appendix D.1), details about the DTU benchmark (Appendix D.2) and implementation details (Appendix D.3).

D.1 Additional video results

We present additional results in the form of videos at: www.tmonnier.com/DBW. Videos are separated in different sections depending on the experiment type. First, we provide view synthesis videos (rendered using a circular camera path), further outlining the quality of both our renderings and our primitive-based 3D reconstruction. Second, we include videos for physics-based simulations. Such simulations were produced through Blender by simply uploading our output primitive meshes. Note that for modeling primitive-specific motions in Blender (*e.g.*, in our teaser figure), primitives should not overlap at all, thus requiring a small preprocessing step to slightly move the primitives for a clear separation. Because each primitive is its own mesh, this operation is easily performed within Blender. Finally, we provide video results where we perform scene editing and compare our amodal view synthesis results to the ones of Nerfacto introduced in Nerfstudio [Tancik et al., 2023]. Models for amodal synthesis are optimized on a homemade indoor scene built from a forward-facing capture only. We use Nerfstudio for data processing and data convention.

D.2 DTU benchmark

In Figure D.1, we show for each scene a subset of the input images as well as 360° renderings of the GT point clouds obtained through a structured light scanner. To compute performances, we use the evaluation: https://github.com/jzhangbs/DTUeval-python.

D.3 Implementation details

Icosphere and superquadric UV mapping. We use spherical coordinates that we correct to build our texture mapping for the unit icosphere. Figure D.2 shows our process with an example. Specifically, we retrieve for each vertex its spherical coordinates $\eta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\omega \in [-\pi, \pi]$ which are linearly mapped to the UV space $[0, 1]^2$. Because such parametrization presents discontinuities and strong triangle deformations at the poles, we perform two corrections. First, we fix discontinuities by copying the border pixels involved (using a circular padding on the texture image) and introducing new 2D vertices such that triangles do not overlap anymore. Second, we avoid distorted triangles at the poles by creating for each triangle, a new 2D vertex

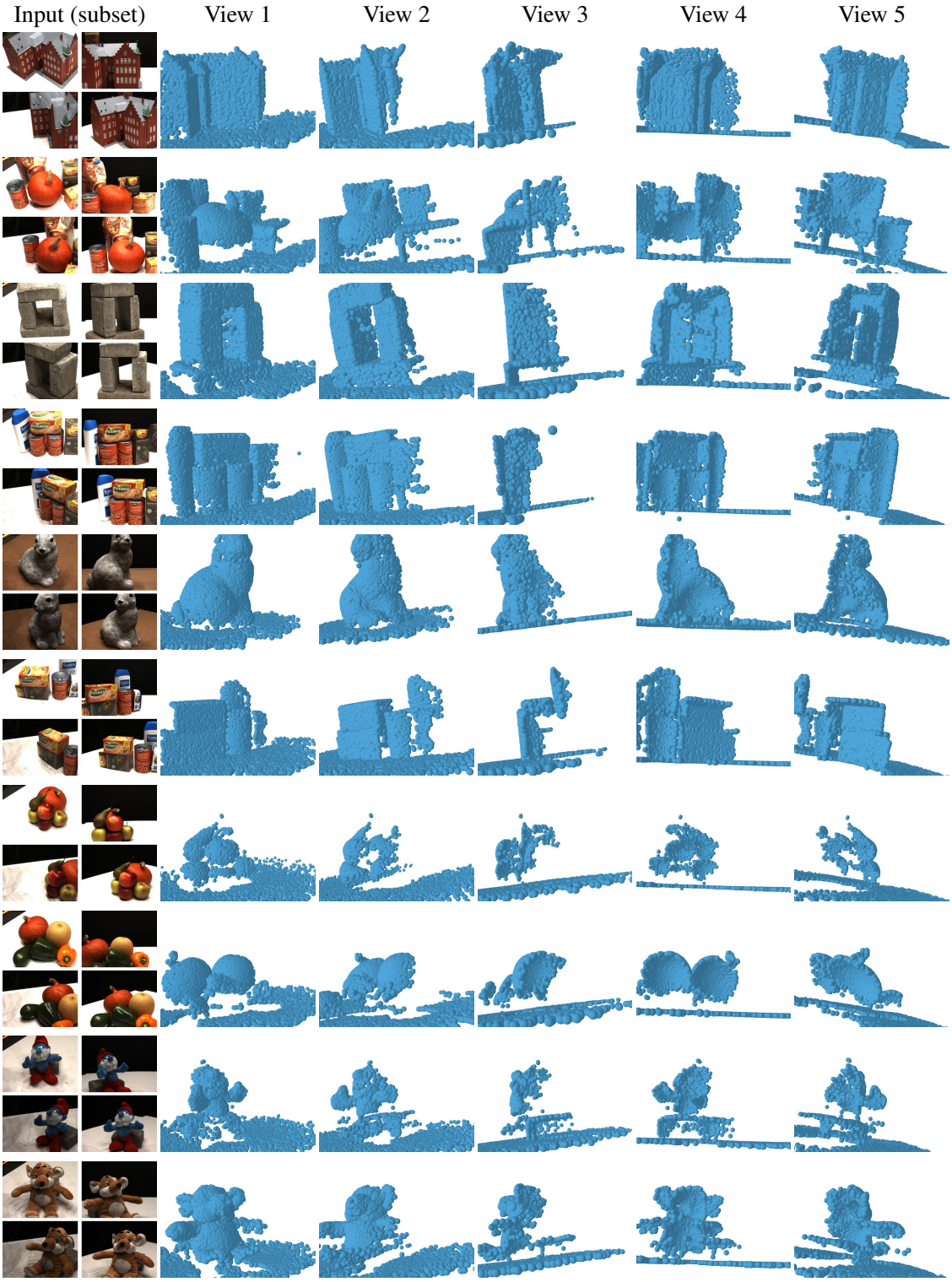
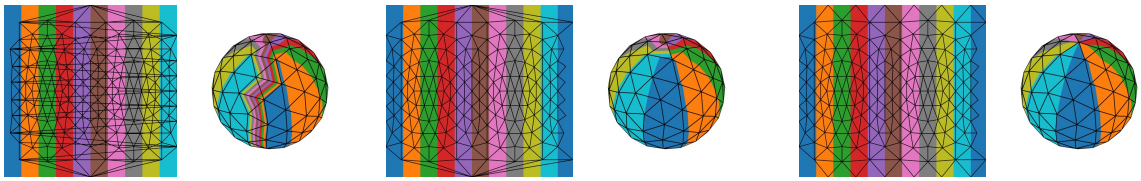


Figure D.1: **DTU scenes with ground-truth.** We show a subset of the input images as well as renderings of the GT point clouds. From top to bottom, scenes are: `S24`, `S31`, `S40`, `S45`, `S55`, `S59`, `S63`, `S75`, `S83`, `S105`.



(a) Raw spherical coordinates (b) Fixing border discontinuities (c) Fixing distortions at the poles

Figure D.2: **Our icosphere UV-mapping.** We illustrate different UV parametrizations using raw spherical coordinates **(a)** as well as our modified coordinates to fix discontinuities **(b)** and to prevent distortions at the poles **(c)**. For each parametrization, we show the texture image with the face edges representing the UV-mapping as well as a rendering example of the associated icosphere.

positioned in the middle of the other two vertices. As detailed in the main paper, we derive a superquadric mesh from a unit icosphere in such a way that each vertex of the icosphere is continuously mapped to the superquadric vertex. As a result, the texture mapping defined for the icosphere is directly transferred to our superquadric meshes without any modification.

Design choices. Except constants related to the world scale, orientation and position in the 3D space w.r.t. to the known cameras, all our experiments share the same design choices. Specifically, all the following design choices are defined for a canonical 3D scene assumed to be centered and mostly contained in the unit cube, with a y-axis orthogonal to the ground and pointing towards the sky. We roughly estimate the scene-specific constants related to the world scale and pose (through coarse visual comparisons or using the camera locations), and apply them to our final scene model to account for the camera conventions.

The background corresponds to a level-2 icosphere (320 faces), the ground plane is subdivided into 128 uniform faces (for visual purposes) and superquadric meshes are derived from level-1 icospheres (80 faces). The scale for the background and the ground is set to 10. The ground is initialized perpendicular to the y-axis and positioned at $[0, -0.9, 0]$.The poses of our primitive blocks are initialized using a Gaussian distribution for the 3D translation and a random 6D vector for the rotation such that rotations are uniformly distributed on the unit sphere. We parametrize their scale with an exponential added to a minimum scale value of 0.2 to prevent primitives from becoming too small. These scales are initialized with a uniform distribution in $[0.5, 1.5]$ and multiplied by a constant block scale ratio of 0.25 to yield primitives smaller than the scene scale. The superquadric shape parameters are implemented with a sigmoid linearly mapped in $[0.1, 1.9]$ and are initialized at 1 (thus corresponding to a raw icosphere). Transparency values are parametrized with a sigmoid and initialized at 0.5. All texture images have a size of 256×256, are parametrized using a sigmoid and are initialized with small Gaussian noises added to gray images.

Optimization details. All our experiments share the same optimization details. We use Pytorch3D framework [Ravi et al., 2020] to build our custom differentiable rendering process and use the default hyperparameter $\sigma = 10^{-4}$. Our model is optimized using Adam [Kingma and Ba, 2015] with a batch size of 4 for roughly a total of 25k iterations. We use learning rates of 0.05 for the texture images and 0.005 for all other parameters, and divide them by 10 for the last 2k iterations. Following our curriculum learning process, we optimize the model for the first 10k iterations by downsampling all texture images by 8. Then, we optimize using the full texture resolution during the next 10k iterations. Finally, to further increase the rendering quality, we threshold the transparency values at 0.5 to make them binary, remove regularization terms related to transparencies (*i.e.*, $\mathcal{L}_{\text{parsi}}$ and $\mathcal{L}_{\text{over}}$), divide the weights for the other terms $\mathcal{L}_{\text{perc}}$ and $\mathcal{L}_{\text{TV}}$ by 10, decrease the smoothness rendering parameter σ to 5×10^{-6} and finetune our model for the final 5k iterations. In particular, this allows the model to output textures that are not darken by non-binary transparencies. During the optimization, we systematically kill blocks reaching a transparency lower than 0.01 and at inference, we only show blocks with a transparency greater than 0.5. Similar to [Paschalidou et al., 2021], we use $\lambda = 1.95$ and $\tau = 0.005$ in our overlap penalization.

Bibliography

Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.-A., and Hjelm, R. D. [2019]. Unsupervised State Representation Learning in Atari. In *NeurIPS*. 49

Annunziata, R., Sagonas, C., and Cali, J. [2019]. Jointly Aligning Millions of Images with Deep Penalised Reconstruction Congealing. In *ICCV*. 24

Arandjelović, R. and Zisserman, A. [2019]. Object Discovery with a Copy-Pasting GAN. *arXiv:1905.11369 [cs.CV]*. 26

Ba, J. L., Kiros, J. R., and Hinton, G. E. [2016]. Layer Normalization. *arXiv:1607.06450 [stat.ML]*. 16

Barr, A. H. [1981]. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*. 32, 82, 84, 87

Bay, H., Tuytelaars, T., and Van Gool, L. [2006]. SURF: Speeded Up Robust Features. In *ECCV*. 18

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. [2015]. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *NIPS*. 68

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. [2009]. Curriculum learning. In *ICML*. 68

Besl, P. and McKay, N. D. [1992]. A method for registration of 3-D shapes. *TPAMI*. 75, 118

Biederman, I. [1987]. Recognition-by-components: a theory of human image understanding. *Psychological review*. 31

Binford, T. [1971]. Visual Perception by Computer. In *IEEE Conference on Systems and Control*. 31, 82

Bolles, R. C. and Fischler, M. A. [1981]. A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data. In *IJCAI*. 32

Bookstein, F. [1989]. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *TPAMI*. 41

Borenstein, E. and Ullman, S. [2004]. Learning to Segment. In *ECCV*. 50, 60, 62, 108

Bottou, L. and Bengio, Y. [1995]. Convergence Properties of the K-Means Algorithms. In *NIPS*. 23, 39, 53

Bouchard, G. and Triggs, B. [2005]. Hierarchical Part-based Visual Object Categorization. In *CVPR*. 22

Brice, C. R. and Fennema, C. L. [1970]. Scene analysis using regions. *Artificial Intelligence*. 7, 24

Brown, M., Szeliski, R., and Winder, S. [2005]. Multi-Image Matching Using Multi-Scale Oriented Patches. In *CVPR*. 18

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. [2019]. MONet: Unsupervised Scene Decomposition and Representation. *arXiv:1901.11390 [cs.CV]*. 26, 49, 52, 55, 57

Burl, M. C., Weber, M., and Perona, P. [1998]. A probabilistic approach to object recognition using local photometry and global geometry. In *ECCV*. 22

Cao, L. and Fei-Fei, L. [2007]. Spatially coherent latent topic model for concurrent object segmentation and classification. In *ICCV*. 25

Carneiro, G. and Lowe, D. [2006]. Sparse Flexible Models of Local Features. In *ECCV*. 22

Caron, M. [2021]. *Self-Supervised Learning of Deep Visual Representations*. Phd thesis, Université Grenoble Alpes. 20

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. [2018]. Deep Clustering for Unsupervised Learning of Visual Features. In *ECCV*. 41, 42

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. [2021]. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*. 19

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. [2015]. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012 [cs.CV]*. 3, 11, 32, 67, 75, 79, 121, 122

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. [2017]. Deep Adaptive Image Clustering. In *ICCV*. 22

Chang, K.-Y., Liu, T.-L., and Lai, S.-H. [2011]. From Co-saliency to Co-segmentation: An Efficient and Fully Unsupervised Energy Minimization Model. In *CVPR*. 60

Chaperon, T. and Goulette, F. [2001]. Extracting cylinders in full 3D data using a random sampling method and the gaussian image. In *VMV*. 32

Chen, M., Artières, T., and Denoyer, L. [2019a]. Unsupervised Object Segmentation by Redrawing. In *NeurIPS*. 26

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. [2020]. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 19

Chen, W., Gao, J., Ling, H., Smith, E. J., Lehtinen, J., Jacobson, A., and Fidler, S. [2019b]. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *NeurIPS*. 20, 28, 66, 72, 85, 86, 115

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. [2016]. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *NIPS*. 22, 43

Cho, M., Kwak, S., Schmid, C., and Ponce, J. [2015]. Unsupervised Object Discovery and Localization in the Wild. In *CVPR*. 25

Chopra, S., Hadsell, R., and LeCun, Y. [2005]. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *CVPR*. 22

Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. [2016]. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *ECCV*. 5, 28, 66

Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T. F. Y., Dideriksen, T., Arora, H., Guillaumin, M., and Malik, J. [2022]. ABO: Dataset and Benchmarks for Real-World 3D Object Understanding. In *CVPR*. 3

Cox, M., Sridharan, S., Lucey, S., and Cohn, J. [2008]. Least Squares Congealing for Unsupervised Alignment of Images. In *CVPR*. 23

Cox, M., Sridharan, S., Lucey, S., and Cohn, J. [2009]. Least-Squares Congealing for Large Numbers of Images. In *ICCV*. 23

Crawford, E. and Pineau, J. [2019]. Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks. In *AAAI*. 27

Cremers, D. and Kolev, K. [2011]. Multiview Stereo and Silhouette Consistency via Convex Functionals over Convex Domains. *TPAMI*. 30

Criminisi, A., Reid, I., and Zisserman, A. [2000]. Single view metrology. *IJCV*. 28

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. [2004]. Visual Categorization with Bags of Keypoints. In *ECCV*. 21

Dalal, N. and Triggs, B. [2005]. Histograms of Oriented Gradients for Human Detection. In *CVPR*. 18

Darmon, F., Bascle, B., Devaux, J.-C., Monasse, P., and Aubry, M. [2022]. Improving neural implicit surfaces geometry with patch warping. In *CVPR*. 31, 88

Debevec, P. E., Taylor, C. J., and Malik, J. [1996]. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*. 32

Delage, E., Honglak Lee, and Ng, A. [2006]. A Dynamic Bayesian Network Model for Autonomous 3D Reconstruction from a Single Indoor Image. In *CVPR*. 28

Dempster, A. P., Laird, N. M., and Rubin, D. B. [1977]. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*. 23, 36, 37, 43

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei [2009]. ImageNet: A large-scale hierarchical image database. In *CVPR*. 19

Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. [2019]. Learning elementary structures for 3d shape generation and matching. In *NeurIPS*. 32

Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H. [1999]. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*. 72

Dizaji, K. G., Herandi, A., Deng, C., Cai, W., and Huang, H. [2017]. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In *ICCV*. 22, 43

Doersch, C., Gupta, A., and Efros, A. A. [2015]. Unsupervised Visual Representation Learning by Context Prediction. In *ICCV*. 19

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. [2014]. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *ICML*. 19

Donahue, J., Krähenbühl, P., and Darrell, T. [2017]. Adversarial Feature Learning. In *ICLR*. 19

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. [2014]. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*. 19, 23

Duchi, J., Hazan, E., and Singer, Y. [2011]. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR*. 16

Duggal, S. and Pathak, D. [2022]. Topologically-Aware Deformation Fields for Single-View 3D Reconstruction. In *CVPR*. 29, 66, 75

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. [2017]. Adversarially Learned Inference. In *ICLR*. 19

Elman, J. L. [1993]. Learning and development in neural networks: The importance of starting small. *Cognition*. 68, 72

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. [2020]. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. In *ICLR*. 26, 55

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. [2016]. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *NIPS*. 27, 49

Faktor, A. and Irani, M. [2012]. "Clustering by Composition" - Unsupervised Discovery of Image Categories. In *ECCV*. 25

Faugeras, O. and Keriven, R. [1998]. Complete dense stereovision using level set methods. In *ECCV*. 30

Felzenszwalb, P. F. and Huttenlocher, D. P. [2005]. Pictorial Structures for Object Recognition. *IJCV*. 21

Fergus, R., Perona, P., and Zisserman, A. [2003]. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *CVPR*. 22

Fergus, R., Perona, P., and Zisserman, A. [2005]. A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition. In *CVPR*. 21

Fergus, R., Perona, P., and Zisserman, A. [2007]. Weakly Supervised Scale-Invariant Learning of Models for Visual Recognition. *IJCV*. 22

Finger, S. [1994]. *Origins of neuroscience: a history of explorations into brain function*. Oxford University Press. 65

Fischler, M. A. and Bolles, R. C. [1981]. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*. 32

Fischler, M. A. and Elschlager, R. A. [1973]. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*. 21

Fouhey, D. F., Gupta, A., and Hebert, M. [2013]. Data-Driven 3D Primitives for Single Image Understanding. In *ICCV*. 32

Freeman, W. T. and Adelson, E. H. [1991]. The Design and Use of Steerable Filters. *TPAMI*. 18

Freeman, W. T. and Roth, M. [1995]. Orientation Histograms for Hand Gesture Recognition. In *IEEE International Workshop on Automatic Face and Gesture Recognition*. 18

Frey, B. J. and Jojic, N. [1999]. Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm. In *CVPR*. 23, 36

Frey, B. J. and Jojic, N. [2002]. Fast, large-scale transformation-invariant clustering. In *NIPS*. 23, 36

Frey, B. J. and Jojic, N. [2003]. Transformation-Invariant Clustering Using the EM Algorithm. *TPAMI*. 23, 36

Fu, Q., Xu, Q., Ong, Y.-S., and Tao, W. [2022]. Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction. In *NeurIPS*. 31

Fukushima, K. [1969]. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*. 15

Furukawa, Y. and Hernández, C. [2015]. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*. 30

Furukawa, Y. and Ponce, J. [2007]. Accurate, Dense, and Robust Multi-View Stereopsis. In *CVPR*. 30

Gadelha, M., Maji, S., and Wang, R. [2017]. 3D Shape Induction from 2D Views of Multiple Objects. In *3DV*. 29

Galliani, S., Lasinger, K., and Schindler, K. [2015]. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In *ICCV*. 30

Gao, J., Chen, W., Xiang, T., Tsang, C. F., Jacobson, A., McGuire, M., and Fidler, S. [2020]. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In *NeurIPS*. 31

Geiger, A. and Wang, C. [2015]. Joint 3D Object and Layout Inference from a Single RGB-D Image. In *GCPR*. 32

Gepperth, A. and Pfülb, B. [2019]. Gradient-based training of Gaussian Mixture Models in High-Dimensional Spaces. *arXiv:1912.09379 [cs.LG]*. 40

Gidaris, S., Bursuc, A., Puy, G., Komodakis, N., Cord, M., and Perez, P. [2021]. OBoW: Online Bag-of-Visual-Words Generation for Self-Supervised Learning. In *CVPR*. 19

Gidaris, S., Singh, P., and Komodakis, N. [2018]. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*. 19

Goel, S., Gkioxari, G., and Malik, J. [2022]. Differentiable Stereopsis: Meshes from Multiple Views Using Differentiable Rendering. In *CVPR*. 8, 31

Goel, S., Kanazawa, A., and Malik, J. [2020]. Shape and Viewpoint without Keypoints. In *ECCV*. 8, 29, 66, 72, 73, 74, 75, 76, 77, 117, 118, 120

Goodfellow, I., Bengio, Y., and Courville, A. [2016]. *Deep Learning*. MIT Press. 14

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. [2014]. Generative Adversarial Nets. In *NIPS*. 7, 19

Grauman, K. and Darrell, T. [2005]. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*. 21

Grauman, K. and Darrell, T. [2006]. Unsupervised Learning of Categories from Sets of Partially Matching Image Features. In *CVPR*. 21, 25

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. [2019]. Multi-Object Representation Learning with Iterative Variational Inference. In *ICML*. 11, 26, 49, 50, 52, 55, 56, 57, 105, 106, 108

Greff, K., Rasmus, A., Berglund, M., Hao, T., Valpola, H., and Schmidhuber, J. [2016a]. Tagger: Deep Unsupervised Perceptual Grouping. In *NIPS*. 26

Greff, K., Srivastava, R. K., and Schmidhuber, J. [2016b]. Binding via Reconstruction Clustering. In *ICLR Workshops*. 26

Greff, K., van Steenkiste, S., and Schmidhuber, J. [2017]. Neural Expectation Maximization. In *NIPS*. 26, 40

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. [2020]. Bootstrap your own latent: A new approach to self-supervised Learning. In *NeurIPS*. 19

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. [2018]. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*. 5, 28, 66, 70, 120

Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., and Tan, P. [2020]. Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching. In *CVPR*. 30

Guédon, A., Monnier, T., Monasse, P., and Lepetit, V. [2023]. MACARONS: Mapping And Coverage Anticipation with RGB Online Self-Supervision. In *CVPR*. 12

Gupta, A., Efros, A. A., and Hebert, M. [2010]. Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics. In *ECCV*. 28, 32, 82

Haralick, R. M., Sternberg, S. R., and Zhuang, X. [1987]. Image Analysis Using Mathematical Morphology. *TPAMI*. 41

Hartley, R. and Zisserman, A. [2003]. *Multiple View Geometry in Computer Vision*. Cambridge University Press. 30

Hastie, T., Tibshirani, R., and Friedman, J. [2001]. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. 14, 44, 99

Häusser, P., Plapp, J., Golkov, V., Aljalbout, E., and Cremers, D. [2018]. Associative Deep Clustering: Training a Classification Network with no Labels. In *GCPR*. 23, 43, 60

He, K., Zhang, X., Ren, S., and Sun, J. [2016]. Deep Residual Learning for Image Recognition. In *CVPR*. 15, 41, 78, 106, 120

Henderson, P. and Ferrari, V. [2019]. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. *IJCV*. 29, 66, 73, 74, 117

Henderson, P., Tsiminaki, V., and Lampert, C. H. [2020]. Leveraging 2D Data to Learn Textured 3D Mesh Generation. In *CVPR*. 29, 66

Henzler, P., Mitra, N. J., and Ritschel, T. [2019]. Escaping Plato's Cave: 3D Shape From Adversarial Rendering. In *ICCV*. 29

Hinton, G. E., Krizhevsky, A., and Wang, S. D. [2011]. Transforming Auto-Encoders. In *ICANN*. 22

Hinton, G. E., Osindero, S., and Teh, Y.-W. [2006]. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*. 7

Hinton, G. E. and Salakhutdinov, R. R. [2006]. Reducing the Dimensionality of Data with Neural Networks. *Science*. 19, 22

Ho, J., Jain, A., and Abbeel, P. [2020]. Denoising Diffusion Probabilistic Models. In *NeurIPS*. 7

Hoiem, D., Efros, A. A., and Hebert, M. [2005]. Geometric Context from a Single Image. In *ICCV*. 28, 32, 65

Hoiem, D., Efros, A. A., and Hebert, M. [2007]. Recovering Surface Layout from an Image. *IJCV*. 28, 32

Hoiem, D., Efros, A. A., and Hebert, M. [2008]. Putting Objects in Perspective. *IJCV*. 66

Hosseini, R. and Sra, S. [2015]. Matrix Manifold Optimization for Gaussian Mixtures. In *NIPS*. 40

Hu, T., Wang, L., Xu, X., Liu, S., and Jia, J. [2021]. Self-Supervised 3D Mesh Reconstruction From Single Images. In *CVPR*. 29, 66, 68, 75, 77

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. [2017]. Learning Discrete Representations via Information Maximizing Self-Augmented Training. In *ICML*. 23, 43, 45, 59, 60

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. [2017]. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *CVPR*. 68

Insafutdinov, E. and Dosovitskiy, A. [2018]. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In *NIPS*. 28, 66, 73, 117

Ioffe, S. [2017]. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In *NIPS*. 16

Ioffe, S. and Szegedy, C. [2015]. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. 16

Ivakhnenko, A. G. and Lapa, V. G. [1966]. *Cybernetic Predicting Devices*. CCM Information Corporation. 15

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. [2015]. Spatial Transformer Networks. In *NIPS*. 24, 27, 41, 52, 107

Jang, E., Gu, S., and Poole, B. [2017]. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*. 109

Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., and Aanaes, H. [2014]. Large Scale Multi-view Stereopsis Evaluation. In *CVPR*. 3, 11, 88

Ji, M., Gall, J., Zheng, H., Liu, Y., and Fang, L. [2017]. SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis. In *ICCV*. 30

Ji, X., Vedaldi, A., and Henriques, J. [2019]. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *ICCV*. 23, 41, 43, 45

Jiang, H. and Xiao, J. [2013]. A Linear Approach to Matching Cuboids in RGBD Images. In *CVPR*. 32

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. [2017]. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *IJCAI*. 22, 43, 45

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. [2017]. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*. 11, 49, 56, 59, 105, 108

Jojic, N. and Frey, B. J. [2001]. Learning Flexible Sprites in Video Layers. In *CVPR*. 26, 27, 50, 51, 67, 115

Joulin, A., Bach, F., and Ponce, J. [2010]. Discriminative clustering for image co-segmentation. In *CVPR*. 25, 60

Joulin, A., Bach, F., and Ponce, J. [2012]. Multi-Class Cosegmentation. In *CVPR*. 25

Kabra, R., Burgess, C., Matthey, L., Kaufman, R. L., Greff, K., Reynolds, M., and Lerchner, A. [2019]. Multi-object datasets. https://github.com/deepmind/multi_object_datasets/. 55, 56, 105, 108

Kaiser, A., Ybanez Zepeda, J. A., and Boubekeur, T. [2019]. A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. *Computer Graphics Forum*. 32

Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. [2018]. Learning Category-Specific Mesh Reconstruction from Image Collections. In *ECCV*. 29, 66, 70, 75, 76, 77

Kar, A., Tulsiani, S., Carreira, J., and Malik, J. [2015]. Category-Specific Object Reconstruction from a Single Image. In *CVPR*. 29, 66

Karayev, S., Trentacoste, M., Han, H., Agarwala, A., Darrell, T., Hertzmann, A., and Winnemoeller, H. [2014]. Recognizing Image Style. In *BMVC*. 46, 47

Karras, T., Laine, S., and Aila, T. [2019]. A Style-Based Generator Architecture for Generative Adversarial Networks. In *CVPR*. 28

Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., and Gaidon, A. [2020]. Differentiable Rendering: A Survey. *arXiv:2006.12057 [cs.CV]*. 20

Kato, H. and Harada, T. [2019]. Learning View Priors for Single-view 3D Reconstruction. In *CVPR*. 29

Kato, H., Ushiku, Y., and Harada, T. [2018]. Neural 3D Mesh Renderer. In *CVPR*. 20, 28, 66, 74

Kazhdan, M. and Hoppe, H. [2013]. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*. 30

Kerbl, B., Kopanas, G., Leimkuehler, T., and Drettakis, G. [2023]. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*. 4, 5, 98

Kingma, D. P. and Ba, J. [2015]. Adam: A Method for Stochastic Optimization. In *ICLR*. 16, 42, 108, 119, 121, 127

Kingma, D. P. and Welling, M. [2014]. Auto-Encoding Variational Bayes. In *ICLR*. 7, 19

Kluger, F., Ackermann, H., Brachmann, E., Yang, M. Y., and Rosenhahn, B. [2021]. Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images. In *CVPR*. 32, 82

Kolev, K., Klodt, M., Brox, T., and Cremers, D. [2009]. Continuous Global Optimization in Multiview 3D Reconstruction. *IJCV*. 30

Kosiorek, A., Sabour, S., Teh, Y. W., and Hinton, G. E. [2019]. Stacked Capsule Autoencoders. In *NeurIPS*. 22, 41, 43, 55, 59, 60, 109

Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. [2018]. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. In *NIPS*. 27

Kramer, M. A. [1991]. Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. *AIChE Journal*. 7

Krizhevsky, A., Sutskever, I., and Hinton, G. E. [2012]. ImageNet classification with deep convolutional neural networks. In *NIPS*. 15, 19, 21

Kuhn, H. W. and Yaw, B. [1955]. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*. 44, 60, 105

Kulkarni, N., Gupta, A., and Tulsiani, S. [2019]. Canonical Surface Mapping via Geometric Cycle Consistency. In *ICCV*. 68, 76

Kutulakos, K. and Seitz, S. [1999]. A Theory of Shape by Space Carving. In *ICCV*. 30

Labatut, P., Pons, J.-P., and Keriven, R. [2007]. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In *ICCV*. 30

Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., and Aila, T. [2020]. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics*. 20

Lattari, L., Montenegro, A., and Vasconcelos, C. [2015]. Unsupervised Cosegmentation Based on Global Clustering and Saliency. In *ICIP*. 60

Lazebnik, S., Schmid, C., and Ponce, J. [2006]. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*. 21

Le, E.-T., Sung, M., Ceylan, D., Mech, R., Boubekeur, T., and Mitra, N. J. [2021]. CPFN: Cascaded Primitive Fitting Networks for High-Resolution Point Clouds. In *ICCV*. 32

Learned-Miller, E. G. [2005]. Data Driven Image Models through Continuous Joint Alignment. *TPAMI*. 23

Lecun, Y. [1988]. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*. 15

LeCun, Y., Bottou, L., Bengio, Y., and Ha, P. [1998]. Gradient-Based Learning Applied to Document Recognition. In *Proc. of the IEEE*. 10, 44, 99

Lee, A. B., Mumford, D., and Huang, J. [2001]. Occlusion Models for Natural Images: A Statistical Study of a Scale-Invariant Dead Leaves Model. *International Journal of Computer Vision*. 26

Lee, D. C., Gupta, A., Hebert, M., and Kanade, T. [2010]. Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces. In *NIPS*. 28, 32

Lee, Y. J., Zitnick, C. L., and Cohen, M. F. [2011]. ShadowDraw: Real-Time User Guidance for Freehand Drawing. In *SIGGRAPH*. 37

Lenc, K. and Vedaldi, A. [2015]. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*. 22

Li, B., Sun, Z., Li, Q., Wu, Y., and Anqi, H. [2019a]. Group-Wise Deep Object Co-Segmentation With Co-Attention Recurrent Neural Network. In *ICCV*. 25

Li, L., Sung, M., Dubrovina, A., Yi, L., and Guibas, L. J. [2019b]. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *CVPR*. 32, 82

Li, Q., Sun, Z., Lin, Z., He, R., and Tan, T. [2016]. Transformation invariant subspace clustering. *Pattern Recognition*. 23

Li, W., Hosseini Jafari, O., and Rother, C. [2018]. Deep Object Co-segmentation. In *ACCV*. 25

Li, X., Liu, S., Kim, K., De Mello, S., Jampani, V., Yang, M.-H., and Kautz, J. [2020]. Self-supervised Single-view 3D Reconstruction via Semantic Consistency. In *ECCV*. 29, 66, 77

Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. [2011]. GlobFit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*. 32

Li, Z. and Snavely, N. [2018]. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*. 10, 46, 47, 104

Lin, C.-H., Wang, C., and Lucey, S. [2020a]. SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images. In *NeurIPS*. 29, 66, 75

Lin, C.-H., Yumer, E., Wang, O., Shechtman, E., and Lucey, S. [2018]. ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing. In *CVPR*. 26

Lin, D., Fidler, S., and Urtasun, R. [2013]. Holistic Scene Understanding for 3D Object Detection with RGBD Cameras. In *ICCV*. 32

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. [2020b]. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. In *ICLR*. 27, 49

Liu, S., Li, T., Chen, W., and Li, H. [2019]. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In *ICCV*. 8, 9, 20, 28, 31, 66, 67, 71, 72, 74, 76, 85, 86, 114, 119, 122

Liu, W., Wu, Y., Ruan, S., and Chirikjian, G. S. [2022]. Robust and Accurate Superquadric Recovery: a Probabilistic Approach. In *CVPR*. 4, 32, 82, 88

Liu, X., Tong, Y., and Wheeler, F. W. [2009]. Simultaneous Alignment and Clustering for an Image Ensemble. In *ICCV*. 23

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. [2020]. Object-Centric Learning with Slot Attention. In *NeurIPS*. 27, 49, 52, 55, 56, 57, 106

Loiseau, R., Bouvier, B., Teytaut, Y., Vincent, E., Aubry, M., and Landrieu, L. [2022a]. A Model You Can Hear: Audio Identification with Playable Prototypes. In *ISMIR*. 3

Loiseau, R., Bouvier, B., Teytaut, Y., Vincent, E., Aubry, M., and Landrieu, L. [2022b]. A Model You Can Hear: Audio Identification with Playable Prototypes. In *ISMIR*. 96

Loiseau, R., Monnier, T., Aubry, M., and Landrieu, L. [2021]. Representing Shape Collections with Alignment-Aware Linear Models. In *3DV*. 3, 12, 96

Loiseau, R., Vincent, E., Aubry, M., and Landrieu, L. [2023]. Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans. *arXiv:2304.09704 [cs.CV]*. 32, 82

Loper, M. M. and Black, M. J. [2014]. OpenDR: An Approximate Differentiable Renderer. In *ECCV*. 20

Loshchilov, I. and Hutter, F. [2019]. Decoupled Weight Decay Regularization. In *ICLR*. 16

Lowe, D. G. [2004]. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*. 18

Lucas, B. D., Kanade, T., et al. [1981]. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*. 23

MacQueen, J. [1967]. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*. 7, 22, 23, 36, 37, 39, 43, 53

Maddison, C. J., Mnih, A., and Teh, Y. W. [2017]. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*. 109

Matheron, G. [1968]. Modèle Séquentiel de Partition Aléatoire. Technical report, CMM. 26, 27, 51

Mattar, M. A., Hanson, A. R., and Learned-Miller, E. G. [2012]. Unsupervised Joint Alignment and Clustering using Bayesian Nonparametrics. In *UAI*. 23

McConnell, R. K. [1982]. Method of and apparatus for pattern recognition. Patent application, United States Patent No. 4567610. 18

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. [2019]. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *CVPR*. 8, 28, 66, 75, 120

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. [2020]. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*. 5, 7, 8, 9, 31, 32, 81, 83, 86

Miller, E. G., Matsakis, N. E., and Viola, P. A. [2000]. Learning from One Example Through Shared Densities on Transforms. In *CVPR*. 23

Monnier, T. and Aubry, M. [2020]. docExtractor: An off-the-shelf historical document element extraction. In *ICFHR*. 12

Monnier, T., Austin, J., Kanazawa, A., Efros, A. A., and Aubry, M. [2023]. Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives. In *NeurIPS*. 12

Monnier, T., Fisher, M., Efros, A. A., and Aubry, M. [2022]. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency. In *ECCV*. 12, 86

Monnier, T., Groueix, T., and Aubry, M. [2020]. Deep Transformation-Invariant Clustering. In *NeurIPS*. 12, 50, 52, 53, 55, 59, 60, 61, 68, 72, 107, 108, 109, 121

Monnier, T., Vincent, E., Ponce, J., and Aubry, M. [2021]. Unsupervised Layered Image Decomposition Into Object Prototypes. In *ICCV*. 12, 67, 84, 115

Mukherjee, S., Asnani, H., Lin, E., and Kannan, S. [2019]. ClusterGAN : Latent Space Clustering in Generative Adversarial Networks. In *AAAI*. 22, 43

Müller, T., Evans, A., Schied, C., and Keller, A. [2022]. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*. 5

Munkberg, J., Chen, W., Hasselgren, J., Evans, A., Shen, T., Muller, T., Gao, J., and Fidler, S. [2022]. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*. 31

Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. [2020]. Atlas: End-to-End 3D Scene Reconstruction from Posed Images. In *ECCV*. 30

Navaneet, K. L., Mathew, A., Kashyap, S., Hung, W.-C., Jampani, V., and Babu, R. V. [2020]. From Image Collections to Point Clouds with Self-supervised Shape and Pose Networks. In *CVPR*. 68

Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M. [2006]. Laplacian mesh optimization. In *GRAPHITE*. 72

Nesterov, Y. E. [1983]. A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*. 16

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. [2011]. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshops*. 10, 44, 50, 59, 99, 108

Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., and Yang, Y.-L. [2019]. HoloGAN: Unsupervised learning of 3D representations from natural images. In *ICCV*. 77

Niemeyer, M. and Geiger, A. [2021]. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In *CVPR*. 77, 120

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. [2020]. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*. 3, 8, 28, 31, 66, 74, 75, 76, 83, 119, 120, 122

Oechsle, M., Peng, S., and Geiger, A. [2021]. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *ICCV*. 31

Oliva, A. and Torralba, A. [2001]. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*. 18

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. [2014]. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. In *CVPR*. 19

Paschalidou, D., Gool, L. V., and Geiger, A. [2020]. Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image. In *CVPR*. 32

Paschalidou, D., Katharopoulos, A., Geiger, A., and Fidler, S. [2021]. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *CVPR*. 32, 87, 127

Paschalidou, D., Ulusoy, A. O., and Geiger, A. [2019]. Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. In *CVPR*. 8, 32, 84

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. [2016]. Context Encoders: Feature Learning by Inpainting. In *CVPR*. 19

Pavllo, D., Spinks, G., Hofmann, T., Moens, M.-F., and Lucchi, A. [2020]. Convolutional Generation of Textured 3D Meshes. In *NeurIPS*. 29

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Jin Chang, Hoffman, K., Marques, J., Jaesik Min, and Worek, W. [2005]. Overview of the Face Recognition Grand Challenge. In *CVPR*. 44, 99

Pons, J.-P., Keriven, R., and Faugeras, O. [2007]. Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score. *IJCV*. 30

Porter, T. and Duff, T. [1984]. Compositing Digital Images. In *SIGGRAPH*. 86

Prabhudesai, M., Goyal, A., Paul, S., van Steenkiste, S., Sajjadi, M. S. M., Aggarwal, G., Kipf, T., Pathak, D., and Fragkiadaki, K. [2023]. Test-time Adaptation with Slot-Centric Models. In *ICML*. 32

Quan, R., Han, J., Zhang, D., and Nie, F. [2016]. Object Co-segmentation via Graph Optimized-Flexible Manifold Ranking. In *CVPR*. 25

Radford, A., Metz, L., and Chintala, S. [2016]. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*. 19

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. [2018]. Improving Language Understanding by Generative Pre-Training. 7

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. [2019]. Language models are unsupervised multitask learners. 7

Ramamonjisoa, M., Stekovic, S., and Lepetit, V. [2022]. MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud. In *ECCV*. 4, 32, 88

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. [2021]. Zero-Shot Text-to-Image Generation. In *ICML*. 7

Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. [2007]. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In *CVPR*. 19, 22

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., and Gkioxari, G. [2020]. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501 [cs.CV]*. 20, 31, 71, 85, 86, 127

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. [2014]. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In *CVPR Workshops*. 19

Reddy, P., Guerrero, P., Fisher, M., Li, W., and Mitra, N. J. [2020]. Discovering Pattern Structure Using Differentiable Compositing. *ACM Transactions on Graphics*. 26

Reddy, P., Guerrero, P., and Mitra, N. J. [2022]. Search for Concepts: Discovering Visual Concepts Using Direct Optimization. In *BMVC*. 27

Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., and Novotny, D. [2021]. Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction. In *ICCV*. 3

Roberts, L. G. [1963]. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology. 8, 27, 31, 82

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. [2021]. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV) 2021*. 3

Romaszko, L., Williams, C. K. I., Moreno, P., and Kohli, P. [2017]. Vision-as-Inverse-Graphics: Obtaining a Rich 3D Explanation of a Scene from a Single Image. In *ICCV Workshops*. 49

Rosenblatt, F. [1958]. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*. 15

Rother, C., Minka, T., Blake, A., and Kolmogorov, V. [2006]. Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs. In *CVPR*. 25

Rubinstein, M., Joulin, A., Kopf, J., and Liu, C. [2013]. Unsupervised Joint Object Discovery and Segmentation in Internet Images. In *CVPR*. 25

Rubio, J., Serrat, J., Lopez, A., and Paragios, N. [2012]. Unsupervised co-segmentation through region matching. In *CVPR*. 25, 60

Rudin, L. I. and Oshe, S. [1994]. Total variation based image restoration with free local constraints. In *ICIP*. 87

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. [1985]. Learning Internal Representations by Error Propagation. Technical report, Institute for Cognitive Science, University of California, San Diego. 15

Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., and Zisserman, A. [2006]. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *CVPR*. 25

Saxena, A., Min Sun, and Ng, A. [2009]. Make3D: Learning 3D Scene Structure from a Single Still Image. *TPAMI*. 28, 66

Schnabel, R., Degener, P., and Klein, R. [2009]. Completion and reconstruction with primitive shapes. *Computer Graphics Forum*. 32

Schnabel, R., Wahl, R., and Klein, R. [2007]. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*. 32

Schönberger, J. L., Zheng, E., Frahm, J.-M., and Pollefeys, M. [2016]. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*. 30

Schroff, F., Kalenichenko, D., and Philbin, J. [2015a]. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. 22

Schroff, F., Kalenichenko, D., and Philbin, J. [2015b]. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. 68

Seitz, S. M. and Dyer, C. R. [1997]. Photorealistic Scene Reconstruction by Voxel Coloring. In *CVPR*. 30

Shaham, U., Stanton, K., Li, H., Nadler, B., Basri, R., and Kluger, Y. [2018]. SpectralNet: Spectral Clustering Using Deep Neural Networks. In *ICLR*. 22, 43

Sharma, P., Tewari, A., Du, Y., Zakharov, S., Ambrus, R. A., Gaidon, A., Freeman, W. T., Durand, F., Tenenbaum, J. B., and Sitzmann, V. [2023]. Neural groundplans: Persistent neural scene representations from a single image. In *ICLR*. 97

Shen, X., Efros, A. A., and Aubry, M. [2019]. Discovering Visual Patterns in Art Collections With Spatially-Consistent Feature Learning. In *CVPR*. 25

Siglidis, I., Gonthier, N., Gaubil, J., Monnier, T., and Aubry, M. [2023]. The Learnable Typewriter: A Generative Approach to Text Analysis. *arXiv:2302.01660 [cs.CV]*. 4, 5, 12

Siméoni, O., Puy, G., Vo, H. V., Roburin, S., Gidaris, S., Bursuc, A., Pérez, P., Marlet, R., and Ponce, J. [2021]. Localizing Objects with Self-Supervised Transformers and no Labels. In *BMVC*. 25

Simonyan, K. and Zisserman, A. [2015]. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*. 15, 72

Singh, K. K., Ojha, U., and Lee, Y. J. [2019]. FineGAN: Unsupervised Hierarchical Disentanglement for Fine-Grained Object Generation and Discovery. In *CVPR*. 26

Sinha, A., Murez, Z., Bartolozzi, J., Badrinarayanan, V., and Rabinovich, A. [2020]. DELTAS: Depth Estimation by Learning Triangulation and Densification of Sparse Points. In *ECCV*. 30

Sitzmann, V., Rezchikov, S., Freeman, W. T., Tenenbaum, J. B., and Durand, F. [2021]. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. In *NeurIPS*. 28

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. [2019]. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *NeurIPS*. 28

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. [2005]. Discovering objects and their location in images. In *ICCV*. 21, 25

Sivic, J., Russell, B. C., Zisserman, A., Freeman, W. T., and Efros, A. A. [2008]. Unsupervised Discovery of Visual Object Class Hierarchies. In *CVPR*. 21, 25

Sivic, J. and Zisserman, A. [2003]. Video Google: A text retrieval approach to object matching in videos. In *ICCV*. 21

Smirnov, D., Gharbi, M., Fisher, M., Guizilini, V., Efros, A. A., and Solomon, J. [2021]. MarioNette: Self-Supervised Sprite Learning. In *NeurIPS*. 27

Smith, C., Yu, H.-X., Zakharov, S., Durand, F., Tenenbaum, J. B., Wu, J., and Sitzmann, V. [2023]. Unsupervised Discovery and Composition of Object Light Fields. *Transactions on Machine Learning Research*. 27

Smith, L. N. [2017]. Cyclical Learning Rates for Training Neural Networks. In *WACV*. 16

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. [2014]. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*. 16

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. [2012]. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*. 49, 60, 108

Szeliski, R. [2010]. *Computer Vision: Algorithms and Applications*. Springer-Verlag. 14

Tai, K. S., Bailis, P., and Valiant, G. [2019]. Equivariant Transformer Networks. In *ICML*. 22

Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., and Kanazawa, A. [2023]. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *SIGGRAPH*. 89, 90, 91, 124

Tertikas, K., Paschalidou, D., Pan, B., Park, J. J., Uy, M. A., Emiris, I., Avrithis, Y., and Guibas, L. [2023]. PartNeRF: Generating Part-Aware Editable 3D Shapes without 3D Supervision. In *CVPR*. 31

Tieleman, T. [2014]. *Optimizing Neural Networks That Generate Images*. PhD Thesis, University of Toronto. 55, 109

Tieleman, T. and Hinton, G. E. [2012]. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*. 16

Tulsiani, S., Efros, A. A., and Malik, J. [2018]. Multi-view Consistency as Supervisory Signal for Learning Shape and Pose Prediction. In *CVPR*. 28, 66

Tulsiani, S., Kulkarni, N., and Gupta, A. [2020]. Implicit Mesh Reconstruction from Unannotated Image Collections. *arXiv:2007.08504 [cs]*. 29, 66, 70, 73, 75, 76, 77, 117

Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. [2017a]. Learning Shape Abstractions by Assembling Volumetric Primitives. In *CVPR*. 8, 32, 82, 84

Tulsiani, S., Zhou, T., Efros, A. A., and Malik, J. [2017b]. Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. In *CVPR*. 28, 29, 66, 76

Ulyanov, D., Vedaldi, A., and Lempitsky, V. [2016]. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv:1607.08022 [cs.CV]*. 16

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. [2020]. SCAN: Learning to Classify Images without Labels. In *ECCV*. 60

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. [2018]. Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. In *ICLR*. 27

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. [2017]. Attention is All you Need. In *NIPS*. 27

Vicente, S., Carreira, J., Agapito, L., and Batista, J. [2014]. Reconstructing PASCAL VOC. In *CVPR*. 29, 66

Vicente, S., Rother, C., and Kolmogorov, V. [2011]. Object Cosegmentation. In *CVPR*. 25

Vincent, E., Ponce, J., and Aubry, M. [2023]. Pixel-wise agricultural image time series classification: Comparisons and a deformable prototype-based approach. *arXiv:2303.12533 [cs.CV]*. 4, 6, 96

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. [2008]. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*. 19, 22

Vo, H. V., Bach, F., Cho, M., Han, K., LeCun, Y., Pérez, P., and Ponce, J. [2019]. Unsupervised Image Matching and Object Discovery as Optimization. In *CVPR*. 25

Vo, H. V., Pérez, P., and Ponce, J. [2020]. Toward Unsupervised, Multi-Object Discovery in Large-Scale Image Collections. In *ECCV*. 25

von Kügelgen, J., Ustyuzhaninov, I., Gehler, P., Bethge, M., and Schölkopf, B. [2020]. Towards causal generative scene models via competition of experts. In *ICLR Workshops*. 27

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. [2018]. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *ECCV*. 5, 28, 66, 68, 72

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. [2021]. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *NeurIPS*. 31

Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. [2019]. Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs. In *ICLR Workshops*. 26

Weber, M., Welling, M., and Perona, P. [2000a]. Towards automatic discovery of object categories. In *CVPR*. 22

Weber, M., Welling, M., and Perona, P. [2000b]. Unsupervised Learning of Models for Recognition. In *ECCV*. 22

Weber, R. A. S., Eyal, M., Skafte, N., Shriki, O., and Freifeld, O. [2019]. Diffeomorphic Temporal Alignment Nets. In *NeurIPS*. 24

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. [2010]. Caltech-UCSD Birds 200. Technical report, California Institute of Technology. 67, 76

Winn, J. and Jojic, N. [2005]. LOCUS: Learning Object Classes with Unsupervised Segmentation. In *ICCV*. 26

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. [2017a]. Learning to See Physics via Visual De-animation. In *NIPS*. 49

Wu, J., Tenenbaum, J. B., and Kohli, P. [2017b]. Neural Scene De-rendering. In *CVPR*. 49

Wu, S., Makadia, A., Wu, J., Snavely, N., Tucker, R., and Kanazawa, A. [2021]. De-rendering the World's Revolutionary Artefacts. In *CVPR*. 29, 66

Wu, S., Rupprecht, C., and Vedaldi, A. [2020]. Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Images in the Wild. In *CVPR*. 29, 66, 72

Wu, Y. and He, K. [2019]. Group Normalization. *IJCV*. 16

Wu, Y., Liu, W., Ruan, S., and Chirikjian, G. S. [2022]. Primitive-based Shape Abstraction via Nonparametric Bayesian Inference. In *ECCV*. 32, 82

Wysoczańska, M., Monnier, T., Trzciński, T., and Picard, D. [2022]. Towards Unsupervised Visual Reasoning: Do Off-The-Shelf Features Know How to Reason? In *NeurIPS Workshops*. 12

Xiang, Y., Mottaghi, R., and Savarese, S. [2014]. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *WACV*. 67, 76

Xiao, H., Rasul, K., and Vollgraf, R. [2017]. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747*. 10, 44, 99

Xiao, J., Russell, B., and Torralba, A. [2012]. Localizing 3D cuboids in single-view images. In *NIPS*. 82

Xie, J., Girshick, R., and Farhadi, A. [2016]. Unsupervised Deep Embedding for Clustering Analysis. In *ICML*. 22, 43

Xu, Q., Wang, W., Ceylan, D., Mech, R., and Neumann, U. [2019]. DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction. In *NeurIPS*. 28

Yan, X., Yang, J., Yumer, E., Guo, Y., and Lee, H. [2016]. Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision. In *NeurIPS*. 28, 66

Yang, J., Kannan, A., Batra, D., and Parikh, D. [2017]. LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation. In *ICLR*. 26

Yang, J., Parikh, D., and Batra, D. [2016]. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *CVPR*. 22, 43, 99

Yang, L., Luo, P., Loy, C. C., and Tang, X. [2015]. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*. 11, 67, 77, 78, 115

Yang, X., Deng, C., Zheng, F., Yan, J., and Liu, W. [2019]. Deep Spectral Clustering Using Dual Autoencoder Network. In *CVPR*. 43

Yang, Y., Chen, Y., and Soatto, S. [2020]. Learning to Manipulate Individual Objects in an Image. In *CVPR*. 27

Yao, C.-H., Hung, W.-C., Jampani, V., and Yang, M.-H. [2021]. Discovering 3D Parts from Image Collections. In *ICCV*. 68

Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. [2018]. MVSNet: Depth Inference for Unstructured Multi-view Stereo. In *ECCV*. 30

Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., and Quan, L. [2019]. Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference. In *CVPR*. 30

Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., and Quan, L. [2020]. Blended-MVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. In *CVPR*. 11, 89, 91

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. [2021]. Volume Rendering of Neural Implicit Surfaces. In *NeurIPS*. 31, 88, 89

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., and Lipman, Y. [2020]. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *NeurIPS*. 31, 88

Ye, Y., Tulsiani, S., and Gupta, A. [2021]. Shelf-Supervised Mesh Prediction in the Wild. *arXiv:2102.06195 [cs]*. 29

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. [2021]. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*. 83

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. [2016]. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv:1506.03365 [cs]*. 67, 77, 78

Yu, H., Xian, M., and Qi, X. [2014]. Unsupervised co-segmentation based on a new global GMM constraint in MRF. In *ICIP*. 60

Zeiler, M. D. and Fergus, R. [2014]. Visualizing and Understanding Convolutional Networks. In *ECCV*. 19

Zhang, J. [2020]. Visibility-aware Multi-view Stereo Network. In *BMVC*. 30

Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. [2007]. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*. 21

Zhang, J. Y., Yang, G., Tulsiani, S., and Ramanan, D. [2021a]. NeRS: Neural Reflectance Surfaces for Sparse-view 3D Reconstruction in the Wild. In *NeurIPS*. 31, 72

Zhang, R., Isola, P., and Efros, A. A. [2016]. Colorful Image Colorization. In *ECCV*. 19

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. [2018]. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*. 72, 86

Zhang, Y., Chen, W., Ling, H., Gao, J., Zhang, Y., Torralba, A., and Fidler, S. [2021b]. Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering. In *ICLR*. 28, 66

Zheng, E., Dunn, E., Jojic, V., and Frahm, J.-M. [2014]. PatchMatch Based Joint View Selection and Depthmap Estimation. In *CVPR*. 30

Zhou, P., Hou, Y., and Feng, J. [2018]. Deep Adversarial Subspace Clustering. In *CVPR*. 22

Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H. [2019]. On the Continuity of Rotation Representations in Neural Networks. In *CVPR*. 84, 118

Zhu, J.-Y., Lee, Y. J., and Efros, A. A. [2014]. AverageExplorer: Interactive Exploration and Alignment of Visual Data Collections. In *SIGGRAPH*. 37, 47