Vision as Inverse Graphics for Detailed Scene Understanding

Pol Moreno Comellas

Doctor of Philosophy

Institute for Adaptive and Neural Computation

School of Informatics

University of Edinburgh

2019

Abstract

An image of a scene can be described by the shape, pose and appearance of the objects within it, as well as the illumination, and the camera that captured it. A fundamental goal in computer vision is to recover such descriptions from an image. Such representations can be useful for tasks such as autonomous robotic interaction with an environment, but obtaining them can be very challenging due the large variability of objects present in natural scenes.

A long-standing approach in computer vision is to use generative models of images in order to infer the descriptions that generated the image. These methods are referred to as “vision as inverse graphics” or “inverse graphics”. We propose using this approach to scene understanding by making use of a generative model (GM) in the form of a graphics renderer. Since searching over scene factors to obtain the best match for an image is very inefficient, we make use of convolutional neural networks, which we refer to as the *recognition models* (RM), trained on synthetic data to initialize the search.

First we address the effect that occlusions on objects have on the performance of predictive models of images. We propose an inverse graphics approach to predicting the shape, pose, appearance and illumination with a GM which includes an outlier model to account for occlusions. We study how the inferences are affected by the degree of occlusion of the foreground object, and show that a robust GM which includes an outlier model to account for occlusions works significantly better than a non-robust model. We then characterize the performance of the RM and the gains that can be made by refining the search using the robust GM, using a new synthetic dataset that includes background clutter and occlusions. We find that pose and shape are predicted very well by the RM, but appearance and especially illumination less so. However, accuracy on these latter two factors can be clearly improved with the generative model.

Next we apply our inverse graphics approach to scenes with multiple objects. We propose using a method to efficiently and differentiably model self shadowing which improves the realism of the GM renders. We also propose a way to render object occlusion boundaries which results in more accurate gradients of the rendering function. We evaluate these improvements using a dataset with multiple objects and show that the refinement step of the GM clearly improves on the predictions of the RM for the latent variables of shape, pose, appearance and illumination.

Finally we tackle the task of learning generative models of 3D objects from a collection of meshes. We present a latent variable architecture that learns to separately capture the underlying factors of shape and appearance from the meshes. To do so we first transform the meshes of a given class to a data representation that sidesteps the need for landmark correspondences across meshes when learning the GM. The ability and usefulness of learning a disentangled latent representation of objects is demonstrated via an experiment where the appearance of one object is transferred onto the shape of another.

Lay summary

A fundamental problem in computer vision is to infer detailed descriptions of the objects in an image, such as their shape, pose, and appearance. This is important for instance in robot navigation tasks. However, due to the high complexity of objects and scenes, this is very challenging. The main approach we follow to solve this task is to use our knowledge of the physical process of how an image is generated from objects of the scene (generative model). This is typically done by searching over many possible object and scene configurations and choosing the one that results in a generated image that best describes the image of interest. Since this search process is often very slow, our framework combines it with a faster, but more inaccurate estimation method (recognition model), to speed up this search.

First we explore the benefits of using our framework when objects are modelled to have heavy occlusions. We show how we are able to better recover the appearance and illumination of an object in a cluttered scene when using the generative model. We also show we can also precisely infer which parts of the objects are occluded by other objects in the image.

We then explore the problem when images have multiple objects of interest. First we propose a more realistic shadowing model and show that this improves the inference of objects appearance and illumination of the scene. Second, we explore better ways of rendering the objects (around their edges) using the generative model and show in the experiments this improves the estimation of their shape and pose.

Finally, we focus on generative models of complex objects (such as faces and cars). We propose a method that, given a collection of existing examples of an object type, it can learn to generate new configurations of shape and appearances (such as a new car that was not in the collection), and to also swap the colours and shapes between two different objects (such as a blue pick-up car and a red sports car).

Acknowledgements

I would first like to thank my supervisor Chris for his guidance throughout these years. I've learned a tremendous amount from our research discussions, about machine learning and computer vision, and just as importantly about how to keep improving as a researcher. His deep and genuine interest in the research he does is a great inspiration for me going forward.

I would like to thank the School of Informatics and its staff for providing an excellent research environment and giving me the opportunity to carry out this research. I am also grateful to La Caixa Foundation for providing funding for part of my postgraduate studies.

I would also like to deeply thank all the people who have given me their kind support. There are too many to name all, but I would like to specially mention my parents Josep Maria and Mercè, and my brother Rubèn for always being there whatever the circumstance.

I would like to thank my friends and colleagues Konstantinos, Benigno and Sohan, for their support and great advice; as well as Katie, Hannah, Carmen, Marcel, and many more friends who have all made these past years a much better experience.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pol Moreno Comellas*)

Table of Contents

Chapter 1

Introduction

1.1 Detailed scene understanding

Humans are able to easily understand and reason about the objects in a scene (e.g. a kitchen, a living room, etc.) at a very high level thanks to our ability to represent the world in 3D. Understanding a scene involves having the knowledge of the 3D structures and objects in the scene, as well as the way they are positioned with respect to each other. But a complete description is not only about the 3D shapes, it also relates to the ability to describe the appearance of the objects and how the light sources illuminate them. Equipping a computer system with such representations opens a vast number of possibilities for interaction. Such a system can plan a path through the scene, interact with the objects, and even infer higher level functional descriptions from them. Indeed, for autonomous robotic tasks, having these descriptions is a crucial element to their success. It is not surprising that since the early days of computer vision, recovering the detailed 3D information of a scene from a single image has been one of its fundamental and most challenging goals.

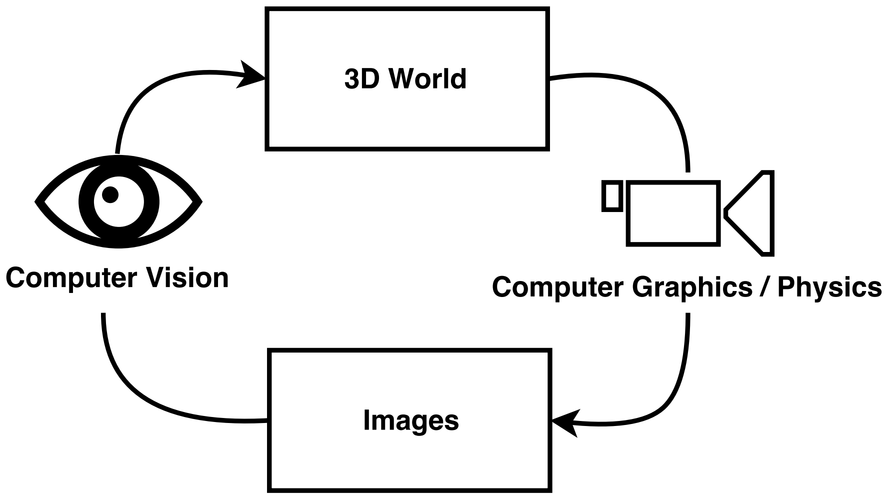


Figure 1.1: The relationship between the world and images.

The representational gap from image pixels to 3D objects is very large. As illustrated in Fig. 1.1, our goal is ultimately to invert the process that generated an image. This is an extremely ill-posed problem since many possible configurations can give rise to a single image. Even the human visual system can make mistakes due to inference ambiguities as is shown in the Ames Room illusion (Ittelson, 1952). Another example in computer vision is the Bas-Relief Ambiguity (Belhumeur et al., 1999) wherein more than one 3D structure explains an image taken under certain lighting and camera assumptions. To make matters more difficult, objects can show a very rich variability in classes of shapes and appearances.

A long-standing approach to inferring the 3D world from an image is precisely to use our prior knowledge of objects and the physical process that generates the image. The idea is to use this in order to constrain and cope with the ill-posed nature of vision. In machine learning this falls under the category of inference using probabilistic *generative models* (GM). In computer vision these are usually referred to as “vision as inverse graphics”, or “inverse graphics” for short (Baumgart, 1974),

Perhaps it is also interesting to consider that inverse graphics may be at the core of the best visual system we know of. Clark (2013) gives a review of the compelling amount of evidence that suggests that the forward and backwards flow of information in neural representations of biological visual systems may correspond to a Bayesian inference framework. The key idea is that the brain learns neural representations by predicting sensory signals, and that action, recognition and inference are driven by prediction error minimization. Yuille and Kersten (2006) refer to this theoretical framework as “analysis-by-synthesis”. The authors make references to recent examples from the computer vision literature that suggest that probability distributions on structured representations are capable of modelling the complexity of natural images and, as such, encourage further experimental studies with regards to the predictive capability of such models in the human visual system.

It is clear that we have all the parts of a complete generative model at our disposal: the physical process of image formation is very well defined and there exist efficient libraries that implement it in various ways. We also have access to large datasets of 3D models of objects, see e.g. Chang et al. (2015). Thus, the aim of this thesis is to explore the use of the generative model in the form of a graphics renderer, in order to infer the shape and appearance of objects, illumination and camera parameters. Inference under these generative models, however, typically involves a hard search problem over an intractably large number of scene configurations, which is why generative modelling is often considered to be only one part of the full picture. A general purpose vision solution must take into account many different aspects. To give some context, let us first define the standard tasks in computer vision according to the Pascal VOC 2010 Challenge (Everingham et al., 2010):

- *Object classification*, or object class recognition, is the task of classifying an object instance of an image according to its class (e.g. classify a picture of a cat as “cat”).

- *Object detection* is the task of inferring if a particular class of object is present in an image and localizing it. This is done by estimating a bounding box that includes all the pixels related to each object in the class.
- *Object segmentation* is the task of assigning regions of pixels to the class of objects in the image. That is, one labels pixels according to their class, or as part of the background.

Thanks to a number of research breakthroughs, the advent of deep learning and availability of large scale labelled datasets being among the most prominent ones, we have seen great advances towards solving these tasks and shrinking the gap with human level performance (Russakovsky et al., 2015). These improvements allow researchers to tackle increasingly more complex problems regarding richer scene descriptions. It is clear that a robust object classification and detection will be a pillar to any full-fledged computer vision system. As such we make use of these advances as part of our proposed methods. We will refer to the methods that are learned discriminatively, i.e. which take as input an image and produce different estimates of interest (e.g. detection of objects, estimates of their shape, pose, etc.) as the *recognition models* (RM).

This thesis is focused on the combination of generative and recognition models of images as illustrated in Fig. 1.2. We explore different architectures when combining them and address various obstacles in the process. To sum up, *in this thesis we focus on the problem of obtaining a detailed description of a scene from a single image using the vision as inverse graphics framework.* We will refer to detailed, in-depth, or fine-grained[1] scene descriptions interchangeably.

We can summarize some of the high-level insights regarding the relative merits of generative and discriminative models in the framework of probabilistic modelling:

1. Encoding our prior knowledge of how the images are formed is done naturally in generative models, also allowing us to generate samples in order to get a sense of how the model describes the images. In addition, these models usually require less training data as more knowledge is incorporated.
2. Generative models are often computationally slower during inference due to the

[1]Note that in computer vision literature, fine-grained scene understanding can have a different meaning, which consists of the task of intra-class classification of closely-related object categories, e.g. recognition of different classes of flowers (Nilsback and Zisserman, 2008).

complexity of the search problem in the latent space.

3. Discriminative models often perform better for specific tasks (e.g. object classification) since the features learned by these models are usually highly informative and specialized for the task at hand. A crucial element to their performance is the availability of a large and sufficiently varied training datasets to avoid over or under-fitting.

4. An important advantage of generative models is that they can deal with weak and no labelling (unsupervised learning), so a large amount of training resources are more readily available.

We consider that a recognition model plays an important role for an efficient solution to inverse graphics, as noted e.g. by Williams et al. (1997). Indeed, finding ways to combine these two approaches offers interesting possibilities and has been a topic of research up to this date.

An important benefit of generative models is that they have not only been devised as inference models, but also as a way to synthesize data in order to train computer vision models with the goal of coping with the common lack of sufficient real world fully labelled data (Hejrati and Ramanan, 2014; Zia et al., 2013; Shotton et al., 2013). In this thesis we also make use of synthetic image data in order to train our recognition models.

Another interesting application of generative models has also been to generate novel scenes and object configurations for different applications. Examples of these include stochastic arrangements of indoor (Fisher et al., 2012; Liu et al., 2014) and outdoor (Yeh et al., 2012) scenes, as well as creating novel object shapes from the parts of objects in a dataset (Kalogerakis et al., 2012; Averkiou et al., 2014).

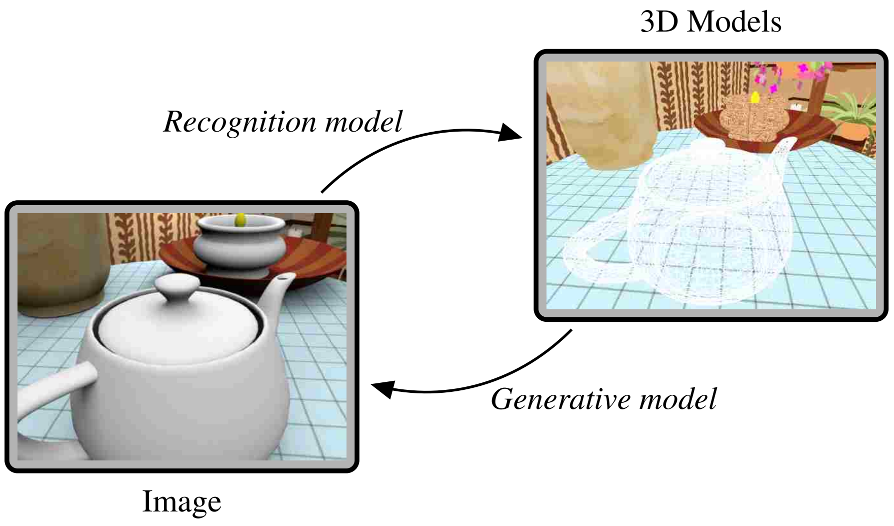


Figure 1.2: Vision as inverse graphics example in which images are interpreted by means of a recognition model, and a generative model is used to synthesize new images. In this case, the object being analysed is the teapot.

1.2 Thesis outline

This thesis focuses on using the inverse graphics approach for detailed scene understanding tasks. The remainder of this document has the following structure:

In **Chapter 2** we motivate our approach and give a brief overview of the scene reconstruction literature and particularly the existing work that makes use of generative models. Finally we describe the overall architecture of the generative models and the recognition models used throughout this thesis.

In **Chapter 3** we analyse the benefits of using inverse graphics when applied to the case of images where objects are partially occluded. We quantify the gains in refinement using a GM which has been designed to be robust to outliers. The experiments, which are carried out using a new image dataset based on our synthetic indoor scene generator, show that while the recognition models can accurately predict pose and shape, they struggle with appearance and illumination. Furthermore, using the robust GM significantly improves the appearance and illumination prediction across all levels of occlusion. This work is published in the following paper:

- Moreno, P., L., Williams, C. K. I., Nash C. and Kohli, P. (2016). Overcoming Occlusion with Inverse Graphics. In *Computer Vision-ECCV 2016 Workshops Proceedings Part III, eds. H. Gang and H. Jegou, Springer LNCS 9915 pp 170-185*

In **Chapter 4** we turn our attention to scenes with multiple objects. We describe a novel architecture of the recognition models used to perform robust predictions of object and global latent variables when an image presents multiple objects. In this chapter the generative model is improved in two ways: (i) we propose the inclusion of a method for efficient modelling of self shadows that results in more realistic renders, (ii) the rendering implementation is improved to address the occlusion boundaries of objects in order to get more accurate derivatives of the rendering function. Thanks to inclusion of (i), experiments show that illumination and appearances are significantly improved. Results also show that the changes in (ii) lead to improved predictions on the variables of shape and pose. Finally we do a qualitative experiment using real images that suggests that our inverse graphics models generalize to the real domain. This work is published in the following paper:

- 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 *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017, pp 940948*

In **Chapter 5** we address the aspect of inverse graphics related to modelling the shape and appearance of 3D objects. We are interested in learning a GM from a collection of meshes of a given object class. To do so we use a representation of the meshes that sidesteps the need of finding corresponding landmark points across meshes. We propose a latent variable architecture that separately captures the underlying factors of shape and appearance from these representations. The ability and usefulness of learning a disentangled latent representation of objects is shown via an experiment where the appearance of one object is transferred onto the shape of another.

Finally, in **Chapter 6** we summarize the contributions and results of this thesis and give a final overview of the state of art in scene understanding. We conclude with a number of interesting directions for future work.

Chapter 2

Background

Scene reconstruction is closely related to our goal of detailed scene understanding in that one of its main aims (and challenges) is the estimation of the 3D geometry of a scene. In Sec. 2.1 we give a brief overview of the different scene reconstruction paradigms in the literature. Using a generative model to reconstruct a scene is the approach we investigate in this thesis. Sec. 2.2 introduces the idea of a probabilistic generative model for vision and defines the key components used throughout this thesis. In Sec. 2.3 we motivate the use of a recognition model, based on a deep learning architecture, as an essential element to our inverse graphics solution for the task of detailed scene understanding.

2.1 Scene reconstruction

Since the early days of computer vision, the problem of general purpose vision has been acknowledged as being ill-posed: multiple scene configurations can give rise to the same image, and there is no straightforward mechanism to invert the physical generative process. Furthermore, visual data has a high number of factors of variability (viewpoint, lighting, shape and appearance and so on) and often has missing data due to occlusion and measurement noise.

As in many artificial intelligence domains, prior knowledge tends to be useful, if not essential, to being able to find good predictive models. As expected, there are different and sometimes opposing views of how to specify our prior knowledge.

One line of research considers using dense pointwise (or pixelwise) models of how images are formed based on physics and optics, and attempt to recover surface properties of the objects by assuming certain smoothness and isotropy regularities. Examples of this include Marr's 2.5D sketch (Marr and Nishihara, 1978) and Tenenbaum's intrinsic images (Barrow and Tenenbaum, 1981). These intermediate representations, such as surface depth, orientation and material properties, are subsequently used to interpret higher level scene configurations. An important drawback of this approach is that the assumed regularities might not always hold, and hence lead to incorrect interpretations (Pentland, 1986).

A different line of research uses 3D geometry of the object models instead. Different ways of representing the 3D geometry include CAD models (Lowe, 1985; Grimson and Lozano-Perez, 1984), and parametric parts-based representations such as generalized cylinders (Binford, 1971) and superquadrics (Pentland, 1986). Brooks' ACRONYM (Brooks, 1983) and Dickinson's OPTICA (Dickinson et al., 1992) are examples of vision systems that use these generic part-based representations. The approach of using models of the objects' shape is commonly known as model-based vision. The long-standing goal of recovering the 3D structure of a scene is known as scene reconstruction.

The mapping from images to models is known as the representational gap and is one of the central issues in computer vision (Keselman and Dickinson, 2001). We can group most of the strategies that attempt to narrow this gap as coming from either a bottom-up or a top-down approach, or a combination of both. On one hand, bottom-up approaches rely on features (i.e. functions that map image pixels to structures useful for different vision tasks) that are predictive of the variables of interest while remaining invariant to nuisance variables as much as possible. In the context of inverse graphics we refer to these as the recognition models. The top-down view uses generative models of images or image features, where inference consists of a search problem over the unknown and high dimensional parameter space for hypotheses that match the image according to the generative model.

Inspired by theories of human vision (Koffka, 2013), some researchers advocated for features based on non-accidental groupings of the image pixels such as collinearity, endpoint proximity, parallelism, and symmetry (Lowe, 1985). Unfortunately, these grouping strategies, also known as perceptual organization, have been shown not to be a reliable basis for recognition due to the issue of not detecting them with enough frequency or stability (Lowe, 1999). Instead, better success has been achieved by extracting collections of appearance-based local features.

Combining both approaches in order to take advantage of their relative strengths is the central theme of this thesis. These hybrid architecture have a long history in the research community. For instance, in the SUCCESSOR system (Binford et al., 1987) scene description hypotheses are voted or selected according to the correspondences between generative model observables and image features (bottom-up). In general, a stage of verification or refinement is performed to improve the scene description hypotheses. This is done by generating or synthesizing images (or observables) using the model, and maximizing the match between these and the real image.

We can see how scene reconstruction and vision as inverse graphics are closely related. While in scene reconstruction the goal is specifically to recover the 3D shape of the objects in a scene, an inverse graphics task might not solely consist of recovering the latent variables of shape, but also in recovering scene descriptions such as the appearances or various attributes of the objects, the illumination, etc.

The difficulty of dealing with the representational gap between the image features and a 3D model of the world led to a shift in computer vision research towards models that focus on lower levels of image representation (i.e. models that represent objects as description of their appearance and 2D shape) which in turn has led to a great success in many vision tasks. However, some researchers believe this trend has begun to swing back to some of these model-based ideas, due in part to having cheaper and faster computers, but also due to having a better understanding and improved techniques in computer vision, computer graphics and machine learning. See (Dickinson, 2009) for a comprehensive historical account of the different approaches in the computer vision community to bridge such representational gap.

The bottom-up and top-down views can be cast into the commonly-known discriminative and generative machine learning frameworks respectively, see Bishop (2006, Chapter 1.5.4) for an overview of the relative merits of each approach. Most of the research in recent years has focused on the discriminative approach due to its ability to efficiently specialize its learned features for a desired task. In this framework, deep learning techniques have lead to significant progress in solving many vision tasks as further described in Sec. 2.3. However, we notice that there has been a rise in research interest in generative models as well as hybrid architectures. In the next sections we describe our choices of generative and recognition models, and give an overview of related work.

2.2 Generative models for vision

From a Bayesian perspective, a generative model is modelled by prior probabilities over latent variables and a likelihood function that determines the probability densities of the observed variables given the latent variables. Inference then consists of estimating the posterior probabilities of the latent variables. In the context of computer vision, the latent variables represent the scene configuration such as the objects' shape and appearance, the illumination and the camera parameters, as illustrated in Fig. 2.1.

Some research has focused on using generative models to infer pixel-wise latent variables related to the object's surface (e.g. depth, albedo and orientation) and illumination of a scene. Above we mentioned a number of early examples that follow this 2.5D approach (Marr and Nishihara, 1978; Barrow and Tenenbaum, 1981). A more recent example is Tang et al. (2012), in which they model pixel-wise reflectances under a Lambertian reflectance assumption and use such a model to recover the illumination and shape (in the form of surface normal vectors per pixel) of the image. A similar approach is to carry out probability density estimation to capture the statistical regularities of specific classes of objects (Tang et al., 2013) without assuming a specific reflectance model. While these are not full generative models of images (i.e. they do not encode the generative process starting from high level 3D representations of the

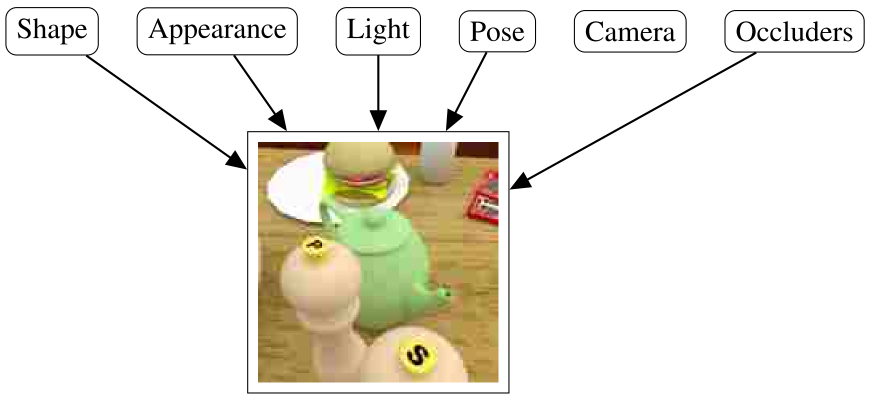


Figure 2.1: Example of a generative model of an indoor image in which the object of interest is a teapot.

objects shapes and appearances), they might be useful for certain applications. While it is not clear whether obtaining these intermediate representations is essential for 3D scene understanding (that is, if we are able to infer the full 3D structure of a scene, having these 2.5D representations might not provide additional useful information), these techniques can be used as part of the pipeline to infer the 3D structures, as done for instance in the recent work of Wu et al. (2017).

Since we are interested in the 3D information of the scene, the focus of this thesis is to model the scene in 3D instead of 2.5.D However, the difficulty of training and doing inference with these models have sometimes limited their success. Hence, these models have usually been constrained to a low dimensionality on the latent variable space or used "poor" models of 3D shape and appearance (Zia et al., 2013; Fidler et al., 2012; Gupta et al., 2010; Hejrati and Ramanan, 2012; Mansinghka et al., 2013). Alternatively, they have relied on extra cues such as depth maps (Shotton et al., 2013).

Recently, there has been a surge of interest in using graphics engines as part of the generative model. This view consists of treating the rendering process as a black box (e.g. a graphics renderer) and subsequently perform sampling techniques to recover the most likely scene parameters (Gupta et al., 2010; Mansinghka et al., 2013; Kulkarni et al., 2014; Jampani et al., 2015). We believe this to be a flexible framework as it makes no assumptions about the types of scenes it can render. Our generative method of choice is very similar in that we also use a graphics renderer. The main difference is that we don't treat the model just as a black-box since we are able to obtain derivatives from it. In Sec. 2.2.1 we explain the rendering process in detail and how these derivatives are computed.

One of the key aspects of the above pieces of work is the use of data-driven Monte Carlo Markov Chain (DD-MCMC) proposals (Tu and Zhu, 2002) in order to more efficiently explore the space of parameters. For instance (Jampani et al., 2015) cluster Histogram of Gradients (HOG) features to estimate the density over the parameter space using kernel density estimation. Kulkarni et al. (2014) propose a similar technique but instead use Deformable Parts Models (DPM) as the data-driven features. These "informed" proposals can also be seen as recognition models, which in general have an important role in this thesis.

Given that a practical graphics renderer cannot produce fully realistic images, often the likelihood function is replaced by a score function that operates on an image feature space of the real and rendered images (Kulkarni et al., 2014; Zia et al., 2013; Mansinghka et al., 2013). It can also be the case that instead of producing pixel intensities, a generative model directly predicts a set of features (i.e. predicted observables). Then, the likelihood function is defined in terms of a score (e.g. distance function) between these generated features and the features of the image. While these types of models avoid having to do complete rendering of images, see e.g. Hoiem et al. (2008), they do not define a proper likelihood of the image data and hence cannot be considered to be principled generative models of images.

2.2.1 The rendering pipeline

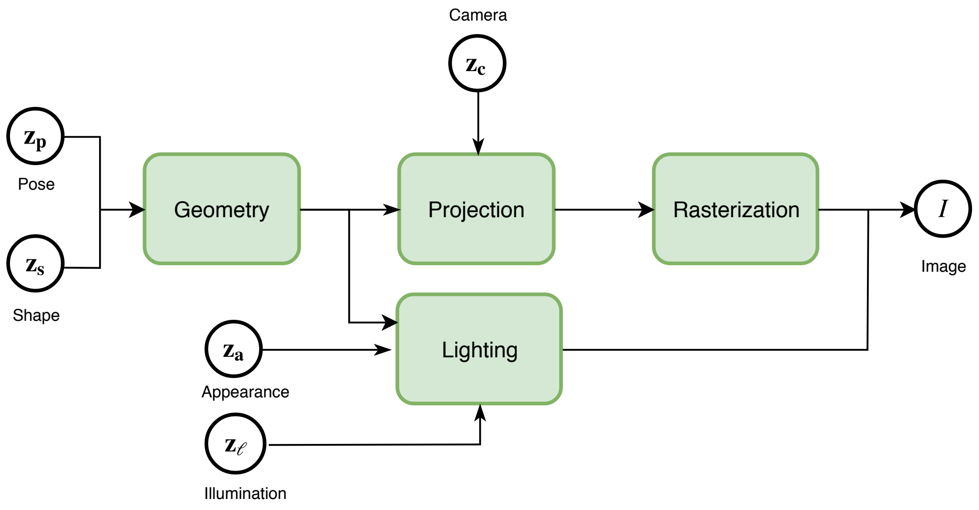


Figure 2.2: The main steps of the rendering pipeline of our generative model.

The main component of our generative model is the graphics renderer. In Fig. 2.2 we show the steps of the rendering pipeline (Akenine-Moller et al., 2008) that are relevant to our research project. The following subsections describe how these different steps are implemented in our inverse graphics solution.

2.2.1.1 Modelling the geometry

As with most graphics engines, the geometry of a scene is modelled using triangular meshes. Throughout this thesis, we represent an object i in the scene by a list of triangular faces $\mathbf{f}_i$, vertices $\mathbf{v}_i$, vertex normals $\mathbf{n}_i$ and vertex colours $\mathbf{vc}_i$. There are different ways we can obtain and generate meshes.

One approach is to simply have a large library of rigid CAD models for each class along with their appearance descriptions, see e.g. Tan et al. (1998) in the case of vehicle recognition and localization. Then, reconstruction consists of finding the model parameters that maximize some similarity measure with the object in the image. However, this simple strategy might most often fail to work well with novel shape and appearance configurations of objects, unless the class of object has a low degree of variability.

A further step has been to hand-craft a parametrization for a specific class of object (e.g. vehicle), which can then be instantiated to match the object in an image (Koller et al., 1994; Haag and Nagel, 1999). In a similar fashion, Jampani et al. (2015) use a rich, learned model of the articulated human body for their shape and pose parameters. These are examples of using domain specific knowledge to represent the models of objects. The problem with these strategies is precisely their reliance on engineered models which makes it time-consuming to models new classes.

A third alternative is to learn a deformable geometry representation from a set of corresponding mesh landmark points, e.g. using Principal Component Analysis (PCA). That is, one can obtain the main modes of shape variation on the vectors of 3D coordinates. Novel shapes can then be obtained as a linear combination of these components plus a mean wireframe (Blanz and Vetter, 1999; Zia et al., 2013). In order to achieve further expressiveness, Blanz and Vetter (1999) compute these modes of variation independently for different regions (parts) of the meshes. In order to do so, the meshes need to be segmented previously according to the different parts. Sometimes there is a further step required to blend the resulting meshes in order to form the complete object (Choi et al., 1991). More recent examples of parts-based shape models are e.g. Nash and Williams (2017); Huang et al. (2015a).

The shape models we use throughout this thesis are based on meshes obtained from CAD datasets. In Chapter 3 we use the third method, a deformable model of a teapot trained from a set of teapot meshes. In Chapter 4 we use a collection of different mug objects instead of learning a parametrized shape model. We refer to the shape latent variables by $\mathbf{z}_s$. The orientation and position of the objects are referred to as the pose latent variables $\mathbf{z}_p$. In this thesis we model the appearances of each object by an RGB colour $\mathbf{z}_a$. Examples of richer descriptions of appearance useful for inverse graphics models include using different colours per vertex (e.g. via a PCA representation of colours such as the one used for faces (Blanz and Vetter, 1999)), or by using textures mapped to the rendered object meshes. In Chapter 5 we propose learning a generative model from a collection of meshes that captures the factors of variation in its latent representation.

2.2.1.2 Illumination model

In computer graphics, the rendering equation (Kajiya, 1986) is the mathematical formula that describes the reflected radiance on the surface of an object as a function of the incoming illumination, the surface properties, and the reflectance model. In its general form we can write the reflected light on a surface point $\mathbf{x}$ as an integral over the whole sphere S:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_S r(\mathbf{x}, \omega_i, \omega_o) L(\mathbf{x}, \omega_i) \cos(\theta_i) d\omega_i, \tag{2.1}$$

where ω_o is the direction of output reflectance, $L_e(\mathbf{x}, \omega_o)$ is the light that the object intrinsically emits, $r(\mathbf{x}, \omega_i, \omega_o)$ is the bidirectional reflectance distribution function (BRDF), and $L(\mathbf{x}, \omega_i)$ defines the light arriving into $\mathbf{x}$ from direction $\omega_i = (\theta_i, \phi_i)$. Fig. 2.3 illustrates such illumination model on a surface.

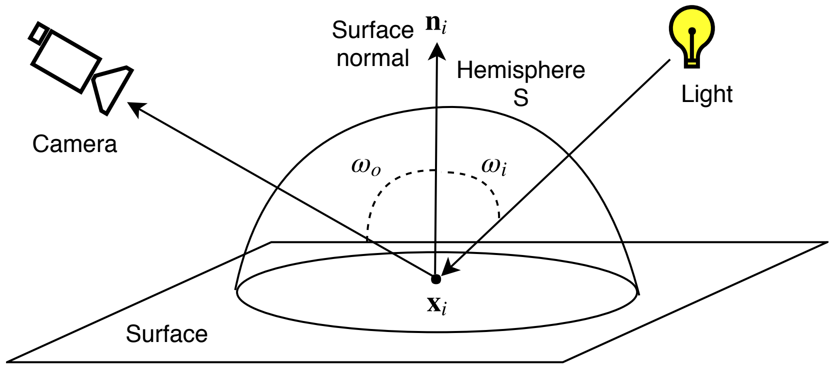


Figure 2.3: Illumination model of a light source on a surface point.

In this thesis we make some simplifying assumptions. First, we assume objects do not have intrinsic illumination and drop $L_e(\mathbf{x}, \omega_o)$. The BRDF we use is the Lambertian reflectance which models the diffuse aspect of objects. Thus, the illumination does not depend on ω_o. Lambertian reflectance is a diffuse surface property where the resulting RGB reflectance $r(\mathbf{x}_i)$ of an incoming light to a point $\mathbf{x}_i$ on the surface is given by

$$\mathbf{r}(\mathbf{x}_i) = \mathbf{z}_a \max(\mathbf{n}_i \cdot \omega_\mathbf{i}, 0), \tag{2.2}$$

where $\mathbf{z}_a$ is the RGB albedo, ω_i is the incoming light direction and $\mathbf{n}_i$ is the vertex normal at point $\mathbf{x}_i$. As is typically in computer graphics, we compute the reflectance at each vertex of our mesh. In order to compute the reflectance in any point within a triangle we interpolate the reflectances at each of its vertices 0, 1 and 2:

$$\mathbf{r}(\mathbf{x}_i) = b_0(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_0) + b_1(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_1) + b_2(\mathbf{x}_i)\mathbf{r}(\mathbf{x}_2), \tag{2.3}$$

where $b_0(\mathbf{x}_i)$, $b_1(\mathbf{x}_i)$ and $b_2(\mathbf{x}_i)$ are the barycentric coordinates of each triangle vertex relative to the sampled point and $b_0(\mathbf{x}_i) + b_1(\mathbf{x}_i) + b_2(\mathbf{x}_i) = 1$. Barycentric coordinates can be computed from the vertex positions thus:

$$\begin{aligned} b_0(\mathbf{x}_i) &= \frac{1}{2|T|}\mathbf{n}_T \times (\mathbf{x}_2 - \mathbf{x}_1)(\mathbf{x}_i - \mathbf{x}_1), \\ b_1(\mathbf{x}_i) &= \frac{1}{2|T|}\mathbf{n}_T \times (\mathbf{x}_0 - \mathbf{x}_2)(\mathbf{x}_i - \mathbf{x}_2), \\ b_2(\mathbf{x}_i) &= \frac{1}{2|T|}\mathbf{n}_T \times (\mathbf{x}_1 - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0), \end{aligned} \tag{2.4}$$

where $|T|$ is the area of the triangle and $\mathbf{n}_T$ the triangle normal vector.

Photorealistic or global illumination methods such as path tracing approximate the integral in Eq. 2.1 via Monte Carlo which generally is very expensive to compute. If we assume a finite set of light sources (e.g. a single directional light source), and assume that light rays do not bounce across objects, then the integral turns into a discrete sum over these sources and illumination becomes more simple to compute. Similarly, some representations of illumination such as Spherical Harmonics (Ramamoorthi, 2006, pp. 385–424) can be used efficiently to model complex scene lighting as we use them in Chapter 3. Whether we model illumination using spherical harmonic coefficients (as in Chapter 3), or a directional light source plus uniform ambient illumination (as in Chapter 4), we denote the light latent variables by $\mathbf{z}_\ell$. For example, for a single light source assumed to be at an infinite distance, $\mathbf{z}_\ell = (\omega_i, l)$ where ω is the light incidence angle and l is its intensity. In such case, the integral of Eq. 2.1 becomes $\int_S r(\mathbf{x}, \omega_i, \omega_o) L(\mathbf{x}, \omega_i) \cos(\theta_i) d\omega_i = r(\mathbf{x}, \omega_i, \omega_o) \cdot l$.

2.2.1.3 Camera projection

The camera of a scene defines the mapping between the object points from 3D to 2D that subsequently forms the image. The positions of the mesh vertices are ini-

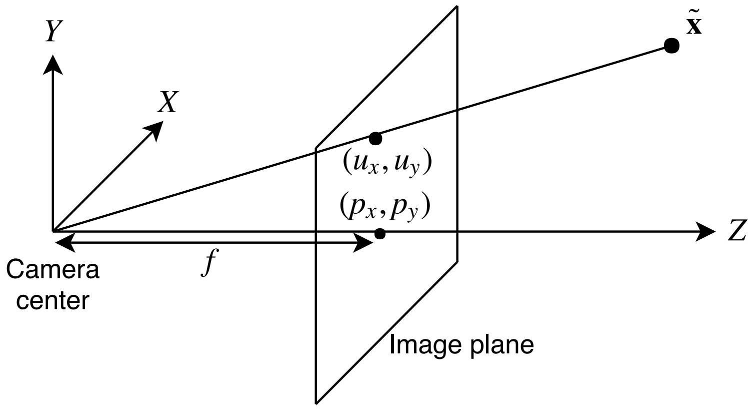


Figure 2.4: Projection of a point $\tilde{\mathbf{x}}$ onto the image plane using a pinhole camera.

tially defined in the *world coordinate frame*. The camera extrinsic parameters define the transformation of the point from the world coordinate frame to the camera space. This can be written as $\tilde{\mathbf{x}} = R(\mathbf{z}_r)(\mathbf{x} - \mathbf{z}_t)$, where $\mathbf{z}_t$ defines the center position of the camera and $R(\mathbf{z}_r)$ is a rotation matrix as a function of the camera latent variables $\mathbf{z}_r = (z_\theta, z_\phi)$. That is, we assume the camera cannot roll (in the pitch, roll, yaw convention), we can parametrize R using the azimuth z_θ and elevation z_ϕ angular variables as $R = R_Z(z_\theta)R_X(z_\phi)$, where

$$R_z(z_\theta) = \begin{bmatrix} \cos(z_\theta) & -\sin(z_\theta) & 0 \\ \sin(z_\theta) & \cos(z_\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_x(z_\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(z_\phi) & -\sin(z_\phi) \\ 0 & \sin(z_\phi) & \cos(z_\phi) \end{bmatrix}. \tag{2.5}$$

Throughout this thesis we assume a basic pinhole model for the camera as shown in Fig. 2.4. Once the objects have been transformed to the camera (or view) space, i.e. $\tilde{\mathbf{x}} = R(\mathbf{z}_r)(\mathbf{x} - \mathbf{z}_t)$, the final projection is given by the matrix of intrinsic parameters, also known as the camera calibration matrix:

$$\begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}, \text{ where} \tag{2.6}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{x}}, \tag{2.7}$$

where f is the focal length, u_x and u_y are the image coordinates of the point $\mathbf{x}$. Instead of the focal length, the field-of-view ω can be parametrized using the relationship

$$f = \frac{a_0}{2}\tan(\frac{\omega}{2}), \tag{2.8}$$

where a_0 is the sensor size which is assumed to be known. We denote the camera latent variables as $\mathbf{z}_c = \{\mathbf{z}_t, \mathbf{z}_r\}$, although alternative parametrizations can be used. For a more detailed description of the pinhole and other types of camera models we refer the reader to Hartley and Zisserman (2004, Ch. 6).

2.2.1.4 Rasterization

The final step in the rendering pipeline is to produce an image out of the projected scene. Rasterization is one of the most common methods for discretizing the continuous representation of the 3D scene. The core algorithm involves looping over each triangle (or more generally referred to as polygon or primitive) in the scene and figuring out which pixels are covered by it. Different surfaces can be projected such that they overlap the area of a pixel, so the triangles are ordered by their distance to the camera and the closest surface point is used to compute the resulting pixel colour as illustrated in Fig. 2.5.

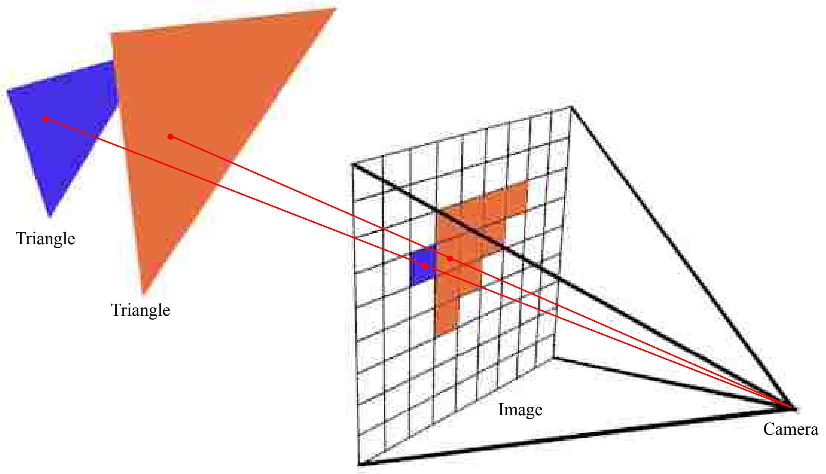


Figure 2.5: Process of rasterization of a triangular surfaces. Image squares represent the pixels of the rasterized image.

Very efficient parallel implementations exist to compute such pixel coverage (Pineda, 1988). When multiple surfaces (or triangle primitives in our case) intersect with the area of a pixel, a choice must be made regarding how this is handled. For instance, the simplest method would be to take the colour of the surface point at the center of the pixel. This strategy can lead to aliasing effects and can have negative results in the final render. More importantly, a small change in the geometry can also produce a sudden change in the rasterized image. For instance, if the orange triangle in Fig. 2.5 shifts to the right at some point the orange pixels become blue due to them becoming only covered by the blue triangle. In the next section we describe the rendering function and how derivatives are obtained from this function. It is clear that special care needs to be taken to handle derivatives of the image with respect to the rasterization step.

To sum all the steps up, the final image I is formed by taking the image point coordinates U that resulted from the rasterization step, and assigning to them the corresponding color intensities L that result from interpolating the surface reflectances as per Eq. 2.3.

2.2.2 Model refinement via gradient-based search

A key idea in this thesis is that of gradient based local search to find the latent variables that maximize the likelihood of an image under the GM. As we will see throughout the experiments in this thesis, having derivatives plays a key role in the optimization process. For this we need gradients of the rendering function with respect to these latent variables. Let us denote by I the image as a matrix $D \times 3$ of pixel colours; V is an $N \times 3$ matrix of the N scene vertices, U is the $N \times 2$ matrix of 2D coordinates of the projected vertices (as per Eq. 2.6), and finally L is the $N \times 3$ matrix of colour reflectances. The computational graph of the rendering function and its derivatives is shown in Fig. 2.6.

Every step of this graph, with the exception of rasterization, is defined analytically as shown in the subsections above and is differentiable. The use of an auto-differentiation framework[1] allows us to code the forward rendering process starting from the latent variables, and then derivatives are numerically evaluated by applying the chain rule to each of the steps of our rendering pipeline. This is a convenient setup as it gives us the flexibility to model each node of our computational graph differently without having to re-derive the gradient functions each time. For instance, to compute $\frac{\partial L}{\partial \mathbf{z}_\ell}$ one needs to compute the derivative of the chosen reflectance function (e.g. Lambertian

[1] `https://github.com/mattloper/chumpy`

$$\begin{tikzcd}
 & & I & & \\
 & U \arrow[ur, "\frac{\partial I}{\partial U}"] & & L \arrow[ul, "\frac{\partial I}{\partial L}"'] & \\
\mathbf{z}_c \arrow[ur, "\frac{\partial U}{\partial \mathbf{z}_c}"] & & V \arrow[ul, "\frac{\partial U}{\partial V}"'] \arrow[ur, "\frac{\partial L}{\partial V}"] & \mathbf{z}_\ell \arrow[u, "\frac{\partial L}{\partial \mathbf{z}_\ell}"'] & \mathbf{z}_a \arrow[ul, "\frac{\partial L}{\partial z_a}"'] \\
 & \mathbf{z}_p \arrow[ur, "\frac{\partial V}{\partial \mathbf{z}_p}"] & & \mathbf{z}_s \arrow[ul, "\frac{\partial V}{\partial \mathbf{z}_s}"'] &
\end{tikzcd}$$

Figure 2.6: Computational graph of the rendering function and the partial derivatives of its different steps.

function) with respect to the illumination latent variables (e.g. intensity and angle of incidence for a single light source at an infinite distance). This derivative can be easily analytically computed for each of the $N \times 3$ color reflectance in L.

The derivatives $\frac{\partial I}{\partial U}$ relate to how the pixel colours change as vertices shift in the image, and as noted by Loper and Black (2014) this is non-differentiable due to the rasterization step. The closest covering triangle to the camera is sampled to obtain the final colour of a pixel. When the meshes are transformed (e.g. scaled, shifted, etc.), the coverage of the triangles on each pixel can vary resulting in sharp changes in the pixel colours. This issue arises predominantly on object boundary pixels that are covered by non-contiguous triangles.

Loper and Black (2014) propose Open Differentiable Renderer (OpenDR), a renderer that uses approximate derivatives of $\frac{\partial I}{\partial U}$. The essential idea is to replace $\frac{\partial I}{\partial U_x}$ and $\frac{\partial I}{\partial U_y}$ by finite-differences using a Sobel filter $\frac{1}{2}[-1,0,1]$ and $\frac{1}{2}[-1,0,1]^T$. This is the framework for gradient based search we use in chapters 3 and 4. Approximating the step $\frac{\partial I}{\partial U}$ with finite differences has the advantage that it needs to be computed once and can be re-used when the relevant image derivatives are computed (e.g. $\frac{\partial I}{\partial \mathbf{z}_s}$, $\frac{\partial I}{\partial \mathbf{z}_c}$, etc.). In contrast, an end-to-end finite difference approximation of $\frac{\partial I}{\partial \mathbf{z}}$ requires evaluating the finite difference for every variable of interest in $\mathbf{z} = \{\mathbf{z}_c, \mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_p, \mathbf{z}_\ell\}$, which becomes more computationally inefficient as the number of variables grow, as Loper and Black (2014) empirically show. In Sec. 4.3.4 we analyze the consequences of different rasterization strategies have on the way gradients are approximated.

In general we are interested in minimizing an error function $E(I_{GT}, I_R)$ of a ground truth image I_{GT} and the GM output I_R, or equivalently, maximizing a likelihood function:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} p(I_{GT}|I_R(\mathbf{z})). \tag{2.9}$$

Now that we have $\frac{\partial I_R(\mathbf{z})}{\partial \mathbf{z}}$, we can differentiate the error function and perform gradient based search using any gradient based optimizer. In this thesis we explore different likelihood functions. Note that if we assume a prior over the latent variables $p(\mathbf{z})$ then we can estimate the maximum a posteriori probability (MAP), i.e.

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} p(\mathbf{z}|I_{GT}), \tag{2.10}$$

where $p(\mathbf{z}|I_{GT}) \propto p(I_{GT}|I_R(\mathbf{z}))p(\mathbf{z})$.

Alternative strategies to gradient-based search are possible. As mentioned above, one way is by finite differences through the renderer, e.g. Eslami et al. (2016) use them to obtain gradients used in an end-to-end unsupervised auto-encoder in which the decoder is a renderer. A different approach is to define an error function $E(I_{GT}, I_R)$ and analytically obtain the gradients while taking special care on the object boundaries (de La Gorce et al., 2008). The advantage of the OpenDR (and our) approach is that by directly differentiating the rendering function, one has the flexibility to define any desired error or likelihood function for different models or tasks. Rhodin et al. (2015) propose a completely different image formation model that directly addresses the differentiability of the visibility function using a translucent medium modelled by Gaussian distributions. While their solution handles occlusion in an analytical and smooth way, their image representation cannot accurately model detailed shape and textures of objects. Finally, a recent piece of work on 3D face reconstruction uses a photometric loss function defined per projected vertex (instead of per pixel) which sidesteps the issue of surface occlusion. However, such an approach does not model the whole image since the error is only defined on those pixels near which a vertex of the face is projected.

2.3 Discriminative models for vision

Discriminative strategies use features of the images that rely on being informative for a given task, yet invariant to nuisance factors such as the object viewpoint, intra-class variability, and the illumination of a scene.

Well known examples of these features are the Scale-invariant feature transform (SIFT) (Lowe, 1999) and Histogram of Oriented Gradients (HOG) descriptors (Dalal and Triggs, 2005). The key idea is that these features capture the gradients of pixel intensities in a discrete set of orientations, which are then pooled (local neighbourhoods of features are grouped together) into histograms which are further normalized in order to achieve invariance to local lighting, viewpoint and translation, as well as improving the robustness to noise. In order to achieve scale invariance, these features are usually extracted at multiple scales in what is called a pyramid representation.

Inspired by techniques in the text domain, one way to use these features is by grouping them into orderless vectors known as bag-of-features. These feature vectors are usually quantized and converted into codebooks (e.g. using clustering techniques) in order to achieve an efficient representation (Zhang et al., 2007). When analysing an image, it is typical to only represent an image as a sparse set of these codewords, also called keypoints. Finally a statistical classification technique, such as Support Vector Machines (SVM), is used to determine the object category based on the extracted image features (Zhang et al., 2007). It is important to note that every step of the classification procedure is important to achieve a good performance. For instance, using more powerful encoding techniques (which lose less information than standard histograms of quantized local features) such as Fisher vectors (Perronnin et al., 2010) allows these techniques to significantly improve their object detection performance (Cinbis et al., 2013).

Instead of using bag-of-features, an alternative approach is to use descriptors that are spatially aligned in the image. That is, image predictions are scored by computing linear functions of the features extracted from the image (Dalal and Triggs, 2005). These type of techniques are commonly referred to as template-based. In the context of template-based techniques, the best recent results for object detection come from heavily engineered methods known as Deformable Parts Model (Felzenszwalb et al., 2010). These models use a coarse global template and a set of fine-grained templates (which represent different object parts) that can lead to efficient object recognition systems. One drawback of these systems is their use of exhaustive scanning over the image in order to match the templates. This procedure is also known as sliding windows, see e.g. (Rowley et al., 1998), and usually limits the type of features to those that can be efficiently computed.

The techniques described above do not explicitly exploit the 3D structure of objects (e.g. the fact that the appearance and shape remain similar under small viewpoint changes). Instead of using many view-point dependent classifiers, variants of such methods have been devised that efficiently integrate this multi-view property (Thomas et al., 2006). Hence, learned object representations can be transferred across different viewpoints requiring less training data and improving detection on novel viewpoints. These discriminative techniques are also used for more fine-grained image interpretation which is the main task of interest in this thesis. For instance, Lopez-Sastre et al. (2011) use a mixture of DPM components to represent discrete set of viewpoints and Pepik et al. (2012) use a 3D parts displacement model (instead of 2D) for pose estimation. The use of Computer-aided Design (CAD) models can provide geometric information to better initialize and train these models (Glasner et al., 2011; Hejrati and Ramanan, 2014; Lim et al., 2014; Zia et al., 2013). In this thesis we also make use of CAD models (and synthetic data in general) as an effective strategy when there is insufficient labelled real data.

Finally, there are alternative ways to the template based approach when combining image features in order to improve the robustness of the final prediction. Random sample consensus (RANSAC, (Fischler and Bolles, 1981)) iteratively uses subsets of the features to generate and test prediction hypotheses in a robust way. For instance, Lim et al. (2013) and Brachmann et al. (2014) use RANSAC for pose estimation. Alternatively, features can cast votes directly on a discretized (binned) space of the parameters and then the hypotheses for which enough evidence has been accumulated are selected. This technique is known as Hough transform (Duda and Hart, 1972) and different variants are used also for pose estimation (Glasner et al., 2011) and reconstruction (Hejrati and Ramanan, 2014). An idea inspired by the Hough transform is used and described in more detail in the recognition models of Chapter 4.

2.3.1 Deep learning in computer vision

Recently, state-of-the-art performance has been achieved by deep learning methods that learn image features from large amount of training data. In contrast to most handcrafted or linear features explained above, these neural network models are trained in a supervised fashion to minimize the error of an objective function. While neural networks have been used for much longer, it has mostly been in the last five years, thanks to the increasing abundance of data and processing power, that these deep representations have been very successful for many vision tasks including object recognition (Krizhevsky et al., 2012; Szegedy et al., 2015), detection and localization (Sermanet et al., 2013; Girshick et al., 2014), attribute modelling (Zhang et al., 2014) and pose estimation (Tompson et al., 2014; Toshev and Szegedy, 2014). The architectures of these networks usually require a large number of parameters, but techniques have been designed to exploit the spatial structure of images in order to make more efficient use of the parameters (namely convolutional neural networks or CNN).

An important idea in these models is that of searching in the right regions of images. Class-agnostic selective search techniques (van de Sande et al., 2011) that output region proposals is usually seen as an essential step when the memory or computational requirements would otherwise be prohibitive. Other examples use these region proposals before feeding them into convolutional neural networks (CNNs) (Girshick et al., 2014), or apply CNNs in consecutive stages to refine the estimation of recognition and localization (Sermanet et al., 2013; Toshev and Szegedy, 2014). Some solutions also jointly learn not only what to extract but also where to extract. Examples of this approach are Fixation Neural Autoregressive Distribution Estimator (Zheng et al., 2014), and Auto-masking Neural Networks (Yang et al., 2014).

Due to their powerful ability to learn complicated mappings, the architectures of our recognition models throughout this thesis are based on variants of CNNs.

Chapter 3

Overcoming Occlusion with Inverse Graphics

3.1 Introduction

In this chapter we propose an inverse graphics solution to extract detailed descriptions of an object in an indoor scene. These descriptions (or factors) are the shape, appearance and pose of the object, and the illumination of the scene.

One of the factors that negatively affects the performance of object detectors is the occlusion of the objects of interest in images, see e.g. (Hoiem et al., 2012). Other factors include the size and localization of the object, as well as novel (unseen) variations in the shape and appearance of objects. It is clear that these factors can have an even greater impact on more detailed scene understanding tasks than on object detection. Here we study the effect that occlusions have on the performance of scene understanding and propose a combination of RM and GMs to overcome them. Our approach is illustrated in Fig. 3.1 and explained in detail in Sec. 3.3. In general, we refer to the process of optimizing some parameters or latent variables of a generative model as the *fitting* of the model (i.e. maximization of the probability of the data given those parameters or latent variables); and we will refer to the process of fitting the model's latent variables, with those latent variables initialized with a point estimate (e.g. given by the recognition models), as the *refinement* process.

Recent work has made some exploration of the combination of generative and recognition models for scene understanding, see e.g. Yildirim et al. (2015). However, the experiments to date have typically been made on "clean" scenes which do not contain occluding objects and background clutter. From a high level, the motivation to combine recognition and generative models is to leverage the strengths of each of these approaches jointly. Some of most important properties are discussed in Sec. 1.1 of the introduction chapter. In this chapter we explore the benefits of combining a recognition model with a generative model when inferring scene descriptions with images that have occlusions. Our contributions are:

- We study how the inferences are affected by the degree of occlusion of the foreground object, and show that a robust generative model which includes an outlier model to account for occlusions works significantly better than a non-robust Gaussian model.

- We characterize the performance of the recognition model and the gains that can be made by refining the search using a generative model. We find that pose and shape are predicted very well by the RM, but appearance and especially illumination less so. However, accuracy on these latter two factors can be clearly improved with the generative model.

- Production of a new synthetic dataset with which one can evaluate the performance of the models with variation across pose, shape, appearance, complex illumination (extracted from a collection of indoor environment maps), a diverse set of indoor scene backgrounds, and with the foreground object being partially occluded. This goes beyond prior work (Yildirim et al., 2015) which only explores shape, appearance and limited lighting variation.

The structure of the chapter is as follows: we first provide an overview of related work on the issue of occlusions, scene understanding and recent work on inverse graphics (see Sec. 3.2). In Sec. 3.3 we describe the **generative model** as a differentiable renderer. The idea is to make use of approximate gradients that describe how the image intensities change with respect to the scene factors, leading to efficient search over these. The robust and Gaussian likelihood models are also explained here. The **recognition model** (see section 3.4) is trained discriminatively to predict the scene factors of interest. In order to do this we have created a **synthetic dataset** of indoor scenes in which we can generate novel instantiations of object classes of interest, as explained in Sec. 3.5. The experimental setup and results are given in Sec. 3.6.

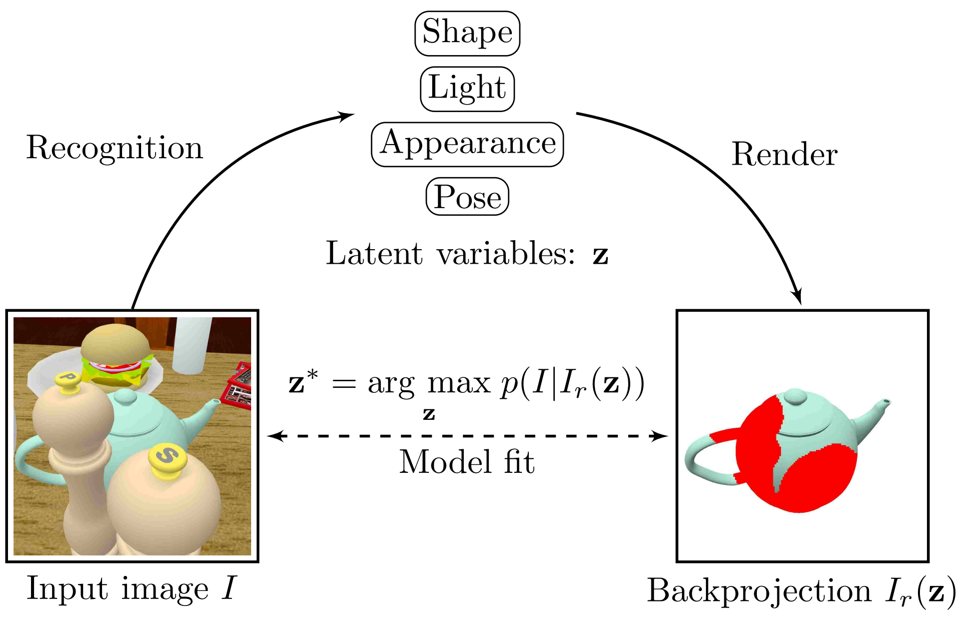

Shape
Light
Appearance
Pose
Latent variables: $\mathbf{z}$
Recognition
Render
$\mathbf{z}^* = \underset{\mathbf{z}}{\arg\max}\, p(I|I_r(\mathbf{z}))$
Model fit
Input image I
Backprojection $I_r(\mathbf{z})$

Figure 3.1: Illustration of our inverse graphics solution. Given an image, the recognition models of shape, lighting, appearance and pose initialize the latent variables. Then, the differentiable renderer generates an image which is refined by optimizing over the latent variables $\mathbf{z}$ that best match the input image, i.e. $\mathbf{z}^* = \underset{\mathbf{z}}{\arg\max}\, p(I_{GT}|I_r(\mathbf{z}))$. The detected occlusion mask is illustrated with red pixels.

3.2 Related work

3.2.0.1 Effect of occlusions in computer vision

The occlusion of objects in images is a factor that is often overlooked when designing computer vision methods. Hoiem et al. (2012) analyze the performance of different object detectors in the PASCAL VOC 2007 (Everingham et al., 2007) and find that misdetection often happens for occluded or small objects. Unfortunately, the way standard computer vision benchmarks are set up does not encourage improvement on some of these aspects (such as robustness to occlusion) as these may have little impact on the overall average precision. Thus they conclude that specific analysis is essential for progress in overcoming such factors. The PASCAL VOC 2008 (Everingham et al., 2008) included an “occluded” flag to the labels of objects with significant occlusion. While this is a good step in the direction of more detailed analyses regarding occlusion, it still lacks a more fine-grained quantification of occlusion, thus limiting the extent to which one can analyze such factor.

A number of works that address occlusion do so by proposing part-based representations as features for the discriminative predictors. This way, if a part of the object is occluded, the model can still produce robust predictions, see e.g. Wang et al. (2009); Wu and Nevatia (2007). Instead of hand-crafting the features one can also design a dataset that includes occluded images. This approach is well suited for deep learning techniques that learn the features from scratch. For instance, Chandler and Mingolla (2016) show improved object recognition performance compared to identical networks that have been trained without occlusion in the training set. Sez-Trigueros et al. (2017) analyze the parts of faces that are highly discriminative and design a training set such that the network evenly learns discriminative features from all regions of the face.

These pieces of work, however, rely on augmenting the dataset with low-level artificial occlusions (e.g. corrupting or adding noise to pixels of the image) which follow a very different pixel distribution compared to real occlusions. Our work differs from the above methods in two ways: first we provide a dataset in which occlusions are exactly quantified (due to having a synthetic scene generator) and are generated in a way one would find with objects embedded in indoor scenes (instead of artificial noise). Second, we study the benefit of using an inverse graphics approach to handling occlusions, as opposed to modifying a RM for this purpose.

3.2.1 Hybrid generative-discriminative architectures.

The problem of inferring the physical world that gave rise to a given image is a long-standing and fundamental problem in computer vision. One recent example of this reconstructive paradigm is the work of Barron and Malik (2015) where depth maps, reflectance maps, and illumination models are recovered from an input image by optimizing a data fit criterion and making use of prior distributions. Another nice example is the work of Loper and Black (2014) as applied to the problem of fitting an articulated human body model to Kinect data.

In the above papers the search over the latent factors of variation is made directly using gradient-based optimization methods. However, it is also possible to make use of a recognition model to predict these factors bottom-up. One example is the algorithm behind Kinect (Shotton et al., 2013), which is shown to be a key component of efficiently estimating human pose.

The work of Jampani et al. (2015) and Kulkarni et al. (2014) makes use of bottom-up sampling probability distribution proposals to get an approximation of the posterior probability of the latent variables more efficiently for black-box renderers. Both papers show improved sampling performance using these data-driven proposals.

Krull et al. (2015) work on estimating 6D pose from RGB-D images. Rather than explicitly modelling occlusions they learn an energy function that compares the rendered model and the input image (which may contain occlusions) using a convolutional neural network. At inference time a search is carried out in 6D pose space to minimize the energy function. Note that this work is restricted to inference of pose, while we also address object shape, appearance and illumination factors.

Perhaps the most closely related work to ours is that of Yildirim et al. (2015), which makes use of both a recognition model and a generative model. They use the morphable face model (Blanz and Vetter, 1999) and explore variation in shape, texture, pose, and lighting direction. In our work we make use of more complex illumination (extracted from a collection of indoor environment maps), and also include scene backgrounds and foreground occlusions. We also analyze the performance of the recognition model with respect to the different factors. A further difference is that Yildirim et al. (2015) make use of MCMC methods for inference, while we use local search using a differentiable renderer.

The Picture framework of Kulkarni et al. (2015a) provides a general probabilistic programming framework for specifying scene understanding problems and general-purpose inference machinery, and applies it to a number of example problems. Rather than generating pixels it uses a “representation layer” of features to assess the match of an input image and the rendered scene, using a likelihood-free method. Note, however, that this means there is a choice of representation that needs to be made (by hand) in order to use the framework. Also, Picture does not explicitly handle occlusions, nor complex lighting.

3.2.2 Learning the generative model.

Another piece of related work is to use a deep learning auto-encoder model for both the recognition model (encoder) and generative model (decoder), as in Kulkarni et al. (2015b). This Deep Convolutional Inverse Graphics Network (DC-IGN) model does not make use of an explicit 3D graphics model, but rather learns a mapping from the graphics code layer (hidden units) to images in the decoder. This gives a lot of freedom to the model as to how it wishes to represent variation in shape, illumination etc. However, it also means that the effect of e.g. pose variation has to be learned specifically for an object class, rather than being generic geometric knowledge. The ability of DC-IGN to generalize to novel images (e.g. under different rotations) seems to be limited if one looks at the visual quality of its renders. Experiments are again conducted with respect to shape, texture, pose, and simple lighting direction variation, without scene backgrounds and foreground occlusions. Note also that the auto-encoder architecture means that there is no scope for refinement of the code predicted by the recognition model, in contrast to our work and that of Yildirim et al. (2015).

The work of Dosovitskiy et al. (2015) is somewhat similar to the DC-IGN model in that it learns a "decoder" network from variables representing shape class, pose and other transformations (e.g. in-plane rotation, colour transformations), although there is no corresponding encoder network. Their network shows impressive generalization over the chair class, but is not used to address issues of scene understanding.

3.3 Generative model

We first describe how the foreground object is modelled, and then discuss the rendering process and our likelihood models.

3.3.1 Object model

In this work, the factors that characterize a visual object are the pose, shape, appearance, and the incident illumination. The choice parametrization of a scene is a trade-off between its ability to model complex scenes and having a compact representation for efficient search over the latent variables. In this work we use the following parametrization:

Shape (10D): We model the shape of a given object class using a deformable mesh characterized by a linear latent variable model, similar to the work of Blanz and Vetter (1999) for modelling faces. The training of the teapot shape model was carried out by my colleague Charlie Nash. More specifically, a Probabilistic Principal Component

Analysis (PPCA) model was trained on landmark point positions for a dataset of teapot CAD meshes. The corresponding landmark point positions were obtained by deforming a set of template points using a variant of non-rigid iterative closest point (ICP) (Amberg et al., 2007). We use 10 latent dimensions so as to capture 90% of the variability in the dataset. Landmark points can be generated from the model by sampling the latent variables, then applying the learned linear transformation and adding noise. Given a set of sampled landmark points, we compute regularized affine transformations that map the template points to the sampled points. The template mesh vertices are deformed by a weighted sum of the nearest-neighbour landmark point transformations. This can be thought of as an embedded deformation (Sumner et al., 2007) in which the nodes of the deformation graph are the landmark points. Mathematically the generated object vertices X are given by:

$$X = T(W_s^T \mathbf{z}_s + \mathbf{b}_s), \tag{3.1}$$

where W_s are the principal components and $\mathbf{b}_s$ are the mean landmark positions and T is the $3N \times D$ "nearest-neighbour" transformation matrix that maps the D landmark points to the $N \times 3$ vertex coordinates. The shape latent variables are thus the 10 PPCA components that explain the most variance.

Pose (2D): We assume the camera viewing direction is centered at the foreground object and positioned at a fixed distance with variable azimuth and elevation angles. While having a camera centered at a fixed distance may seem to be a restrictive setup if we consider how objects are usually seen in real images, it allows us to focus on the issues of interest of this chapter. In Chapter 4 we address the more general case of having multiple objects in scenes in which the camera is sampled at varying distance and focal lengths. The pose is represented by $\mathbf{z}_p = (z_\theta, z_\phi)$ as per the equations 2.5 and 2.6.

Appearance (3D): A global RGB color for all object vertices. We will also refer to this as the albedo as defined in Eq. 2.2, with the reflectance function assumed to be Lambertian.

Illumination (9D): Inverse illumination is known to be an ill-posed problem (Ramamoorthi and Hanrahan, 2001b, Ch. 6), with the additional issue that real illumination can be very complex (thus modelled poorly by single directional light sources). We are interested in representations that are practical for our inverse graphics framework.

One of the most natural representations is to project lighting to basis functions such as Spherical Harmonics, see Ramamoorthi (2006), a technique which is also widely used in computer graphics for efficient approximate rendering of lighting. Other representations include Haar wavelets (Reinhard et al., 2010), as well as basis images of the object class illuminated from different light conditions (Belhumeur and Kriegman, 1998). We use Spherical Harmonics (SH) due to its compact representation and the efficiency at which re-lighting is computed when rotating the light or changing the shape and pose of the modelled objects. Following the notation of (Ramamoorthi, 2006), the Lambertian reflection in the SH basis as a function of the illumination latent variables $\mathbf{z}_\ell^{lm}$ (where ℓ denotes lighting) is given by

$$r_{lm} = \Lambda_l A_l \mathbf{z}_\ell^{lm}, \tag{3.2}$$

where $\Lambda_l = \sqrt{4\pi/(2l+1)}$ and A_l is the Lambertian transfer function in the SH representation:

$$A_l = 2\pi \int_0^{\frac{\pi}{2}} Y_{l0}(\theta_i') \cos\theta_i' \sin\theta_i' d\theta_i'. \tag{3.3}$$

Here Y_{lm} are the different SH basis functions based on the associated Legrende polynomials where $-l \geq m \geq l$ for different orders l (for our 9 coefficients we set $l = 3$). Note that A_l has a closed form solution (Ramamoorthi, 2006). Fig. 3.2 illustrates the nine SH basis functions used. We can see for instance how the first SH basis function only captures a constant function, i.e. uniform lighting in the scene, and subsequent basis functions capture more detailed directional illumination intensities over the whole sphere. Thus, for instance, the first illumination latent variable $z_\ell^{l=0m=0}$ models the intensity of uniform lighting, and e.g. $z_\ell^{l=1m=0}$ models the intensity of lighting exactly coming from the object. Finally, we can compute the reflectance by

$$r(\alpha,\beta) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} r_{lm} Y_{lm}(\alpha,\beta). \tag{3.4}$$

Note how $r(\alpha,\beta)$ is a function of α and β, which are the angles of rotation of the normal vector $\mathbf{n}$ at the surface point of which we are evaluating the irradiance. This follows the relationship $\mathbf{n} = (\sin\alpha\cos\beta, \sin\alpha\sin\beta, \cos\alpha)$. Importantly, any vertex normal is also a function of the shape latent variable $\mathbf{z}_s$ since the matrix of vertex normals N of a mesh is computed from the matrix of vertices X defined in Eq. 3.1.

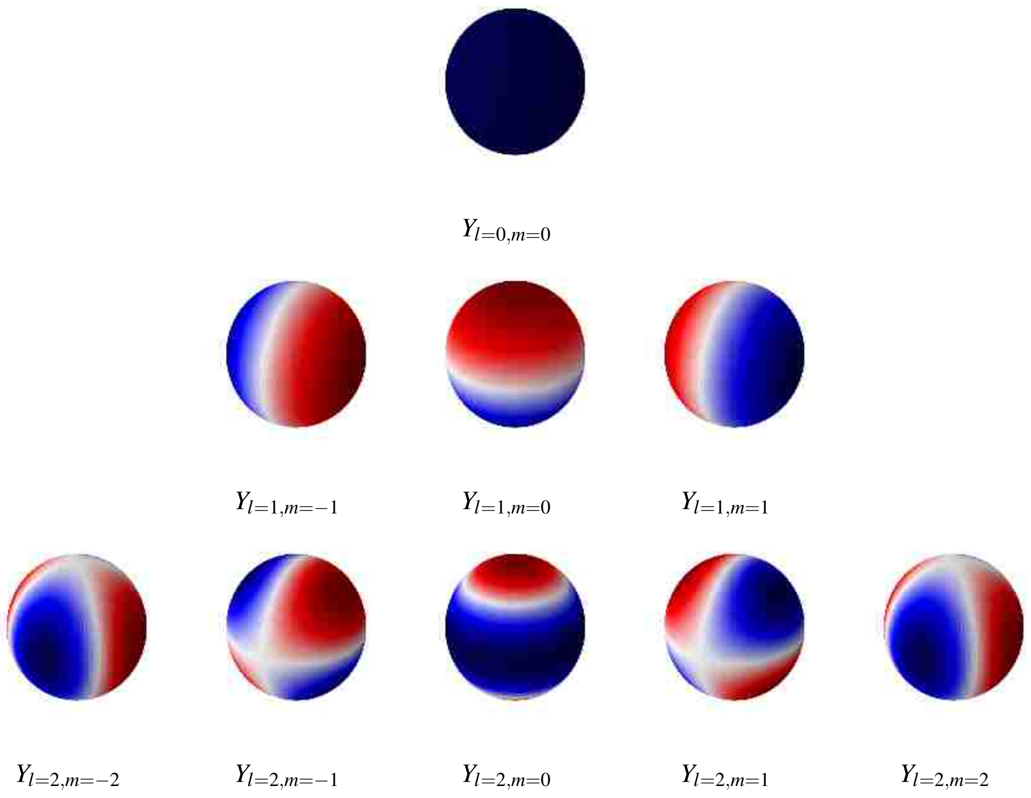


Figure 3.2: Illustration of the SH basis functions defined over the full sphere. Here, red colours indicate positive values and blue colours negative values (and white colour values closer to zero).

Spherical Harmonics form an orthogonal basis that can be used to approximate complex illumination. For Lambertian reflectance, it can be shown that only 9 components ($l = 3$) can approximate images of a convex object by at least 99.22% (Basri and Jacobs, 2003). Even though real world objects have cast shadows, specularities, and other non-Lambertian factors, our assumption is that a Lambertian approximation is sufficient for a large variety of tasks in inverse graphics. While in this work we do not explicitly use a prior probability distribution over $\mathbf{z}_\ell$ when refining the GM to an image, one way to model a prior would be to capture the distribution of SH coefficients found in real scenes[1].

To sum up, our object representation has a total of 24 latent variables: 10 for shape, 2 for pose, 3 for colour and 9 for illumination. Fig. 3.3 shows a few examples of samples and variabilty in the teapot model.

[1]For instance, we could use the SH coefficients dataset from real environment maps, described in Sec. 3.5, that are used to generate our synthetic dataset.

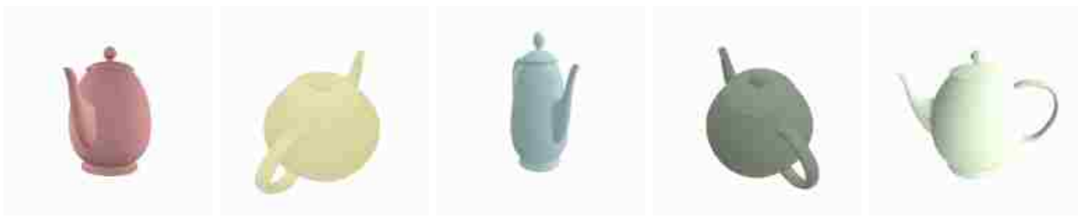


Figure 3.3: Samples from the generative model.

3.3.2 Renderer

As we described in Sec. 2.2.1, we use a graphics renderer as a generative model that takes as input a set of object meshes (which include information of their vertex coordinates, normal vectors, textures and colors) and then generates the renders using OpenGL. Our renderer implementation is based on OpenDR: Differentiable Renderer (Loper and Black, 2014). However, we have completely re-implemented the renderer to make it capable of rendering multiple objects and textures in a simple way (while still remaining differentiable). It still uses the Chumpy[2] auto-differentiation framework as it allows to seamlessly define the complete differentiable scene model from the latent variables to the generated pixels. Our implementation uses the modern version of OpenGL back-end (i.e. OpenGL v3.0 or greater), as opposed to the older fixed function pipeline[3], in order to support modern graphics features and hardware-accelerated rendering. One of the modern OpenGL features we take advantage of in Chapter 4 is the ability to use shaders as programmable software.

The main advantage of the OpenDR framework is that it provides approximate derivatives of the image with respect to the latent variables or any variable of interest. The approximation stems from the fact that the rendering function is non-differentiable due to self-occlusion and occlusion across objects. Having derivatives plays a key role in the efficiency of the optimization process as the dimensionality of our latent space increases.

We can think of the cost function of our generative process (i.e. the reconstruction error) in terms of a likelihood function. In this work we explore two possible models over the pixel intensities. Since we are modelling one foreground object, all the rendered pixels that lie outside the object mask are considered as background and modelled by a uniform distribution. Given an image I, the likelihood of the foreground pixel (fg)

[2]https://github.com/mattloper/chumpy

[3]https://www.khronos.org/opengl/wiki/Fixed_Function_Pipeline

intensities are modelled as follows:

- **Gaussian model:** The simplest case is a Gaussian distribution on the pixel intensities:

  $$p(\mathbf{c}_i|\mathbf{z},fg) = \mathcal{N}(\mathbf{c}_i;\mu_i(\mathbf{z}),\sigma^2 I) \tag{3.5}$$

  where $\mathbf{c}_i$ is the RGB color at pixel i, $\mu_i(\mathbf{z})$ is the RGB color output of the renderer given scene latent variables $\mathbf{z}$, σ^2 is the spherical variance (assumed to be the same for all pixels), and $\mathcal{N}$ denotes the Gaussian distribution. In terms of the optimization landscape, this is equivalent to using a squared error cost function.

- **Robust model:** We want our model to tolerate foreground occlusions by using outlier statistics as in Williams and Titsias (2004)

  $$p(\mathbf{c}_i|\mathbf{z},fg) = \alpha\mathcal{N}(\mathbf{c}_i;\mu_i(\mathbf{z}),\sigma^2 I) + (1-\alpha)\mathcal{U}(\mathbf{c}_i)), \tag{3.6}$$

  where α is the mixing probability of a pixel being unoccluded, and $\mathcal{U}$ is the uniform distribution. We can learn α from our training set by e.g. taking the average of the proportion of unoccluded pixels. We can compute this quantity using the fully labelled training set we also use to train the recognition models. Note that the label of whether a pixel is fg or bg is given by the generative model, thus depends on $\mathbf{z}$ (i.e. we implement it in the generative model by rendering a pixel mask of presence or absence of objects). For simplicity we have dropped the dependence on $\mathbf{z}$ in the notation.

Thus, the overall log-likelihood of an image given the scene latent variables $\mathbf{z}$ is given by

$$L = \sum_{i\in fg} \log p(\mathbf{c}_i|\mathbf{z},fg) + \sum_{j\in bg} \log \mathcal{U}(\mathbf{c}_j). \tag{3.7}$$

The pixel-wise outlier model as used above does not impose spatial priors on regions of occlusion. One could enhance this e.g. using Markov random field models on the occlusion labels, see e.g. (Fransens et al., 2006), at the cost of greater complexity in inference. One could also consider more complex occlusion models that learn the structure of occlusions from data, see e.g. Gao et al. (2011). While Black and Rangarajan (1996) propose robust statistical techniques for a number of vision tasks, the advantage of the latent variable formulation of our robust model is that it allows us to explain occlusions by using the posterior probabilities of the foreground/outlier pixel segmentation.

Estimating the latent variables of interest is obtained by initializing the OpenDR input with the estimates of the latent variables given by the recognition model, followed by locally improving its fit (i.e. refinement) using the likelihood function, Eq. 3.7, subject to the constraint that the PCA coefficients of the shape model lie within ± 3 standard deviations of the mean. This is why it is essential that the predictions of the recognition model should lie within the basin of attraction of the generative model. We explain further details of the optimization procedure in Sec. 3.6.1.2.

3.4 Recognition model

The choice of architecture of the recognition models depends on the task domain and the scene factors we want to infer. For instance, the model can output a single point estimate or take a probabilistic approach; in this work we focus on the former case. Predicting a point estimate is easier in the sense that one does not have to define a probability distribution; however a point estimate comes with no information regarding the uncertainty of the prediction (which we would have if we predicted a probability distribution). Having the latent variables probability distribution from the recognition model could for instance be used to obtain multiple samples, and subsequently perform the fitting process to each of them, potentially leading to a better final fitted prediction of the latent variables. Again, we want the estimates of the bottom-up predictions to lie within the basin of attraction of the generative model.

Using the right feature representation of the images is also key to good discriminative performance. After experimenting with a diverse set of features for each type of parameter (e.g. Histogram of Gradients for pose prediction, and different basis expansions for illumination) we found that learning the features from raw images gave the best prediction performance. Therefore, we make use of convolutional neural networks (CNNs), implemented using the Theano library (Theano Development Team, 2016), to predict each of the different groups of latent variables (shape, appearance, pose, colour and illumination). CNNs have been key to the success for many computer vision tasks in recent years, see e.g. Krizhevsky et al. (2012). The loss function used for networks is the mean squared error (MSE) of the ground truth and predicted values of the latent variables. To train the angular variables we first convert the angles to their sine and cosine representation (and predict both values) to avoid the wrap-around effect. The architecture of choice is almost the same in all cases as it showed a good generalization capability (in the sense that the validation errors where very similar to the training errors, with validation error being only slightly worse): three convolutional layers (64 5x5 filters) with max-pooling and two dense hidden layer (with 256 and 32 hidden units respectively). The input to the networks are the RGB images, except in the case of pose and shape CNNs, in which the input is assumed to be gray-scale. A dropout rate of 0.5 was used, and the Nesterov Accelerated Gradient descent with 0.9 momentum was the optimizer used.

The CNNs were trained on 90% of the 10,000 images of our synthetic dataset which we describe in the following section. From the training set, 20% of the images were used as a validation set to choose the hyperparameters. These hyperparameters include the number of convolutional and fully connected layers, and their corresponding number of channels or hidden units.

3.5 Synthetic ground truth dataset

Synthetic datasets have been used for a variety of computer vision tasks. For instance, Shotton et al. (2013) generate hundreds of thousands synthetic depth images to train randomized decision forests. Zia et al. (2013) synthetize views of 3D CAD models of chairs to discriminatively learn the 2D to 3D mapping from rendered images.

In the inverse graphics literature, Jampani et al. (2015) perform different synthetic experiments which include human body shape reconstruction; Kulkarni et al. (2015b) propose an encoder/decoder architecture and is evaluated on synthetic face and chair data sets; Dosovitskiy et al. (2015) use synthetic data for both training and generation of chair and car models.

In our work we use fully rendered synthetic images in which the object of interest is instantiated in randomly generated and plausible scenes. The purpose of our dataset is to train the RMs and to carry out the quantitative evaluation.

In order to generate synthetic data we use an indoor scene generator based on the dataset of Fisher et al. (2012). This 3D dataset consists of roughly 10,000 CAD models they collected and stochastically arranged together to form different plausible indoor scenes. Here, we use over 80 different indoor scenes generated from their dataset. Fig. 3.4 shows two samples of the indoor scenes used as part of the dataset generation.

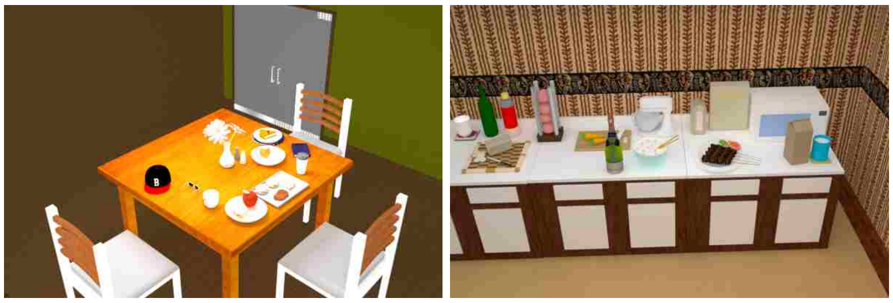


Figure 3.4: Example of 3D arrangement of scenes from Fisher et al. (2012) used in our dataset image generation.

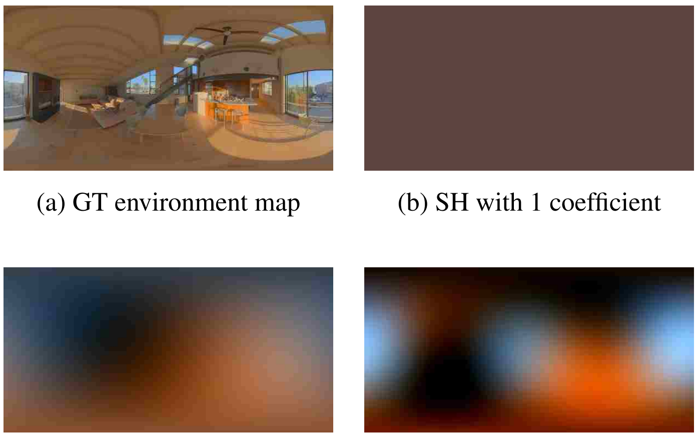
In this chapter we focus on the teapot object class. The reason for choosing teapots is that it is a common type of object in indoor scenes and has a good degree of variability. However, our stochastic scene generation is not limited to only one type of object, and many different object models can be collected from sources such as the Princeton ModelNet (Wu et al., 2015) and easily embedded into the scenes.

We use the deformable teapot model explained in Sec. 3.3.1 to generate random instantiations of the teapot in terms of shape and colour.

To generate the scenes with complex and varied illumination settings, we collected over 70 complex indoor environment maps (also known as light probes) from different sources, see e.g. Debevec (1998). An environment map is the projection of the incident illumination on a point of a scene into an equirrectangular image. Fig. 3.5a shows one such environment maps projected into an image.

The ground truth of each sample in the dataset is generated by instantiating the teapot object in one of the indoor scenes. We first manually tagged a large number of possible surface points in all the scenes where a teapot can be instantiated such that they are placed plausibly on a surface such as a table, etc. The position of a teapot is thus chosen uniformly at random from all these surface points and all the scenes. This involves sampling the shape parameters from the shape prior, as well as a uniformly sampled object rotation around the Z-axis (up axis). The camera's angles of azimuth and elevation are randomly sampled with a uniform distribution on the upper hemisphere and it is placed at a fixed distance from the object. The scene is then illuminated by one of the environment maps which is rotated uniformly along the Z-axis[4]. In total, we produced 10,000 images, 10% of which are reserved as test images for evaluation purposes.

We distinguish two versions of the data set. The first is based on the scenes rendered using the photorealistic ray tracer Cycles[5], which makes use of the environment maps for illumination. The second data set is composed of renders generated using the same rendering pipeline as per the GM as described in Sec. 2.2.1. In this version, we use the collected environment maps to render illumination by using their Spherical Harmonics (SH) representation. Since the Lambertian reflectance function acts as a low-pass filter, it effectively removes high-frequency information (Basri and Jacobs, 2003), hence the resulting illuminated objects are perceptually well approximated compared to using the actual environment maps. In Fig. 3.5 we can see the resulting approximation of one of the environment maps (first image) using Spherical Harmonics of different orders. One can see for instance how the first coefficient captures the mean illumination. Since it is not a global illumination model, it does not render cast shadows across or within objects. We refer to the second one as the OpenGL data set as the rendering is implemented using OpenGL.


(a) GT environment map (b) SH with 1 coefficient

(c) SH with 4 coefficients (d) SH with 9 coefficients

Figure 3.5: Example of illumination modelled by an indoor environment map (first image). The remaining images represent the approximations with 1, 4 and 9 Spherical Harmonics coefficients.

[4]By rotating the environment maps along the up axis, we significantly augment the number of ground truth illuminations in a way that these still represent illuminations from plausible scenes (just with different orientations).

[5]http://www.blender.org/manual/render/cycles/introduction.html

Rendering with Cycles takes roughly three orders of magnitude of time longer compared to the OpenGL method. We use the first type as the out-of-sample test set in our experiments and the second type is used to train and validate our recognition models (which requires many more images). As explained in Sec. 3.6, we evaluate our models with both OpenGL and Cycles versions to determine the change in performance of our models when images have realistic illumination.

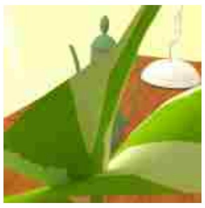
The width and height of images are set to 150 by 150 pixels. Since each image is composed of a teapot randomly instantiated in the scene, the object is often occluded by other objects that are present in the indoor scenes. We can quantify the level of occlusion of the foreground object by the percentage of pixels that are shown in the image relative to the full object mask. Fig. 3.6 shows a few examples rendered with Cycles. Fig. 3.6 (c) shows the histogram of occlusion levels in our dataset, note how there are occlusions of all levels up to 90% (it is not useful to have images with occlusions near 100% for the purpose of training or evaluation).

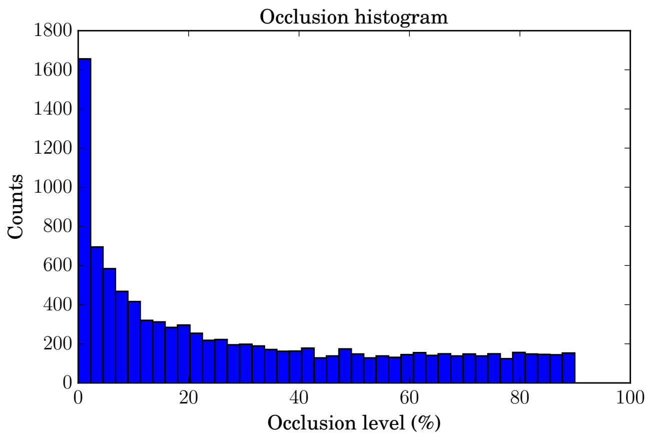
We believe the data set can be a useful resource to the community as a test bed for scene understanding tasks. Our scene generator and dataset are available online and can be accessed in the following project repository: `https://github.com/polmorenoc/inversegraphics`.

(a) 8% Occl.

(b) 21% Occl.

(d) 39% Occl.


(e) 70% Occl.



(c) Histogram of occlusions

Figure 3.6: Examples of our synthetic data set in which the object of interest is randomly instantiated. (c) shows the histogram of the levels of occlusion of all 10,000 images.

3.6 Experiments

3.6.1 Experimental setup

In the following experiment we use a test set which consists of 1000 test samples from our synthetic dataset. Using images from our synthetic dataset gives us the ground truth scene latent variables which are used to quantify the performance of our models. However, the images used for training the recognition models are rendered using the method explained in Sec. 3.5 which uses Lambertian reflectances and does not follow a photorealistic illumination method. In order to evaluate the generalization performance of our models we instead render these images using the an unbiased photorealistic ray-tracer Cycles. For reference, we provide results using both rendering methods.

We are interested in understanding the performance of our models (recognition, Gaussian and robustified) as the occlusion level is increased. As a metric of reference, we also provide a baseline prediction based on the mean values of the different factors on the training set (which we call *mean baseline*). The evaluation metrics and optimization procedure for this experiment are explained below.

3.6.1.1 Evaluation metrics.

Evaluating predictions of the fine-grained scene latent variables (e.g. pose, shape, appearance, illumination) is difficult due to a lack of labelled datasets, but here we have the advantage of having a synthetic dataset in which all these factors are designed to have a rich variability. One way to evaluate the performance of our method is the log-likelihood on the test set. However, that does not give us an easily interpretable quantity. Instead, evaluation is carried out with respect to the predictive performance of the latent variables we are modelling, which have a more intuitive and physical interpretation. We also evaluate how well the un-occluded foreground pixels are predicted.

Choosing the evaluation metric for each of these factors is not necessarily straightforward. For **pose** azimuth and elevation, we define the errors to be the absolute angular difference for each angle (azimuth and elevation variables range 360 and 90 degrees respectively).

Special care needs to be taken to measure the **appearance and illumination** attributes as they interact with each other multiplicatively, see equation 2.2. For the appearance error e_a^i, we convert the colour representation to Lab space[6] which is more perceptually uniform, and omit the luminance L to give

$$e_a^i = \sqrt{(a_{pred}^i - a_{gt}^i)^2 + (b_{pred}^i - b_{gt}^i)^2}, \tag{3.8}$$

where a_{pred}^i and b_{pred}^i correspond to the predicted color dimensions of the Lab representation, and a_{gt}^i and b_{gt}^i to the ground truth values for test image i.

For the illumination error we use the mean squared error (MSE) of the Spherical Harmonics coefficients. It is known that estimating the illumination from a single view can be potentially ill-conditioned (Ramamoorthi and Hanrahan, 2001a) so it is possible that the ground truth illumination cannot be recovered exactly. It is easy to see that our metric is equivalent to the MSE of the incident SH illumination projected on a sphere. Note that we use a scale-invariant version of this metric in order to account for the fact that appearance and illumination interact with each other multiplicatively. A similar evaluation metric is used in the work of Barron and Malik (2015). The scale-invariant sMSE of a predicted $\hat{\mathbf{x}}$ and ground truth $\mathbf{x}^\star$ has the form:

$$sMSE = \min_{\alpha} \|\alpha\hat{\mathbf{x}} - \mathbf{x}^\star\|_2^2, \tag{3.9}$$

where α has a closed-form solution:

$$\alpha = \frac{\hat{\mathbf{x}}^T \mathbf{x}^\star}{\hat{\mathbf{x}}^T \hat{\mathbf{x}}}. \tag{3.10}$$

Finally, we evaluate shape reconstruction by using the mean Euclidean distance between the vertices of the ground truth and predicted meshes, both aligned to a canonical pose.

The occlusion predictions from the robust model are evaluated thus: for the robustified GM, we use the posterior over the pixel-wise outlier (i.e. binary) latent variables to compute the mask. That is, the outlier latent variable of pixel i, $\mathbf{z}_o^i$ has a probability distribution given by

$$p(\mathbf{z}_o^i = 1|\mathbf{c}_i, \mathbf{z}, fg) = \frac{\mathcal{N}(\mathbf{c}_i; \mu_i(\mathbf{z}), \sigma^2 I)}{\alpha\mathcal{N}(\mu_i(\mathbf{z}), \sigma^2 I) + (1-\alpha)\,\mathcal{U}(\mathbf{c}_i))}. \tag{3.11}$$

We use a threshold of $p(\mathbf{z}_o^i = 1|\mathbf{c}_i, \mathbf{z}, fg) >= 0.5$ to define a pixel as belonging to the object.

[6] https://en.wikipedia.org/wiki/Lab_color_space

These are then compared to the ground truth un-occluded foreground pixels using the segmentation accuracy as defined e.g. in Everingham et al. (2010), where the number of true positives is divided by the sum of the true positives plus false positives plus false negatives. For the recognition model (without iterative refinement), we assume that all of the predicted foreground pixels are un-occluded.

3.6.1.2 Optimization procedure.

For the robust model we explored different optimization strategies since convergence is sensitive to the choice of the likelihood variance. If we use a pixel variance that is too large, then occlusions and background clutter will affect the optimization negatively. On the other hand, if the variance is too small, the robust model tends to ignore large parts of the image including those which are important for a correct optimization. We jointly optimize the latent variables of pose, shape, appearance and illumination using a standard deviation of $\sigma = 0.03$ of the pixel likelihood. Note that the pixel color for each RGB channel ranges between 0 and 1. Also, we clamp shape parameters to be less than three standard deviations from the prior mean: it is much more likely that the optimization has gone wrong than it is having an input image with an unreasonably large (or small) deformation of the mesh.

We explored different minimization methods including gradient descent (with and without momentum and decay), nonlinear conjugate gradient (CG)[7], dogleg CG (as used in the OpenDR work), and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (Fletcher, 1987). Overall, nonlinear CG consistently converged to better minima in our tests hence we use it in our experiments. It is not surprising that these methods often converge differently since the optimization landscape is non-convex and the derivatives are approximate.

3.6.2 Results

3.6.2.1 Effect of occlusion on accuracy.

Fig. 3.7 shows the median cumulative predictive performance as a function of the percentage occlusion when evaluating on the photorealistic test images. Note that in pan-

[7]http://learning.eng.cam.ac.uk/carl/code/minimize/

els (a)-(e) lower error is better, while in panel (f) higher scores are better. We also provide confidence intervals for the estimation of the cumulative median to have a notion of its variability, and the significance of this evaluation. These are computed by bootstrapping (re-sampling with replacement subsets of the measured errors) and using the percentiles of the bootstrap distribution, which is known as the percentile bootstrap method (Davison et al., 1997, p. 203). Comparing the recognition network to the recognition network plus robust refinement, we see very similar performance for azimuth, elevation and shape factors, but marked improvements with refinement for appearance, illumination and occlusion. Pose and shape are predicted very well by the recognition model, which suggests that it is good at inferring latent variables for which there are clear and localized cues (e.g. for pose, the locations of tip of the spout, the handle, etc.), even under high levels of occlusion. Subsequent refinement of azimuth and shape factors on average shows little to no improvement, but we do see an improvement for elevation. The small gains in refinement for pose are progressively diminished and we see for the very large occlusion levels that the predicted performance reverts back to the performance of the RM predictions. We also see how the recognition model has not learned to predict the illumination and that occlusion has a strong effect on the prediction. This agrees with the intuition that the other factors can be more easily inferred by having a smaller portion of the object be un-occluded. The recognition models generally either do very well at predicting some of the latent variables (e.g. pose, shape) as shown by the low errors or really badly (such as for illumination). For those variables that are well predicted, like pose, it seems that the RMs have learned to predict the pose from subset of image features reliably (this is the advantage of bottom-up methods), so it does fairly well across most occlusion levels. For illumination on the other hand, performance does degrade as occlusion increases but the predictions by the RM were generally bad even for low level of occlusions (performance is similar to that of the baseline), so there isn't that much room to degrade more from. The biggest effect of occlusion on performance is reflected in the segmentation accuracy, where we clearly see it worsen from 80 to 60% as we increase the occlusion level.

3.6.2.2 Refinement results.

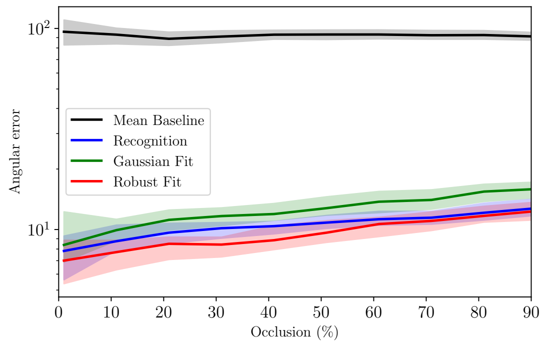
It is noticeable that the Gaussian model performs much worse than the robust model for elevation, appearance, and illumination. Indeed it sometimes performs worse after refinement than the recognition model itself, which is most likely explained by the fact that the Gaussian model does not handle occlusion and background clutter well. Plots of the mean performance rather than the median show similar trends to those in Fig. 3.7, except that the elevation angular error of the robust fit becomes worse than the recognition model over the occlusion range (increasing from 6 to 8 degrees). Analysis shows that this is due to some large errors that arise particularly at higher occlusion levels.

Table 3.1 summarizes the plots by showing the median prediction errors for the baseline, the recognition model prediction, and the Gaussian and robust fits for up to 75% occlusion. (So, for example the performance at 75% occlusion is obtained from all test cases with this much occlusion or less.) The table shows results for both the photorealistic images (left columns) and OpenGL images (right columns). The 90% confidence intervals of the median are also provided for the 75% cumulative occlusion. We notice how for the latent variable predictions for which we saw a clear gain after refinement (i.e. color and illumination errors, and segmentation accuracy), their median error estimates are indeed significantly different.

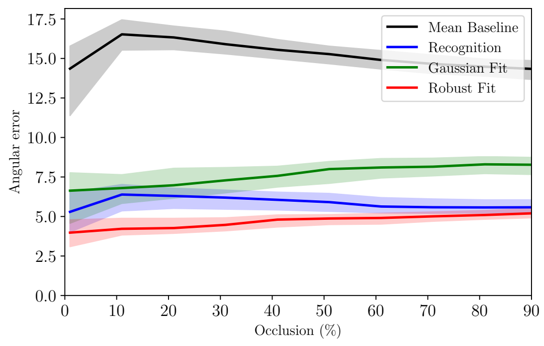
3.6.2.3 Effect of illumination model on accuracy.

Computer vision researchers ultimately wish to create models that can be applied to real images. Our recognition models are trained on synthetic dataset with non-photorealistic illumination. Similarly, our generative model uses the same rendering assumptions (non-global illumination). Thus, it is interesting to investigate the effect that real world illumination may have on the performance of the models. Fig. 3.8 shows the median plots in which we compare the errors of our models when tested on photorealistic (solid lines) vs OpenGL (dashed lines) images. We notice how the relative improvements after refinement are similar in both Cycles and OpenGL cases, but the OpenGL experiments have lower errors (for both RM and RM+fit), as is expected. Interestingly, the prediction of the appearance latent variable seems to be affected the most, while the illumination and shape prediction performance does not change regardless of the illumination model.

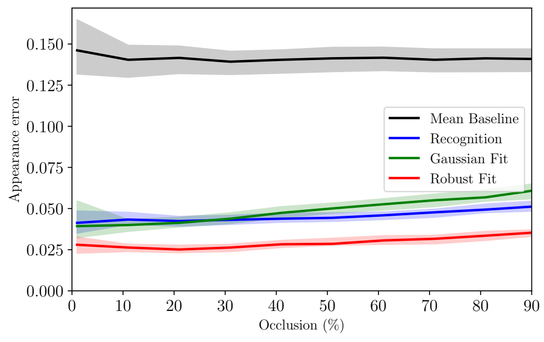
Table 3.1 shows that using the RM estimates followed by robustified model refinement is consistently the best method for all variables and illumination models.



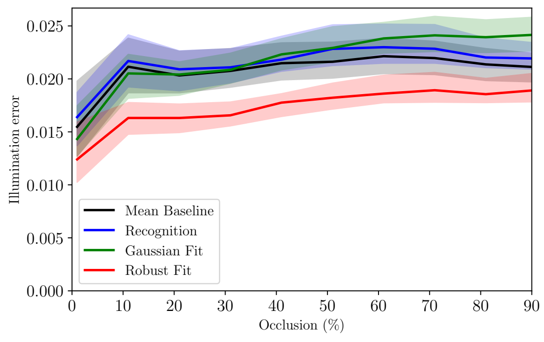
(a) Azimuth errors



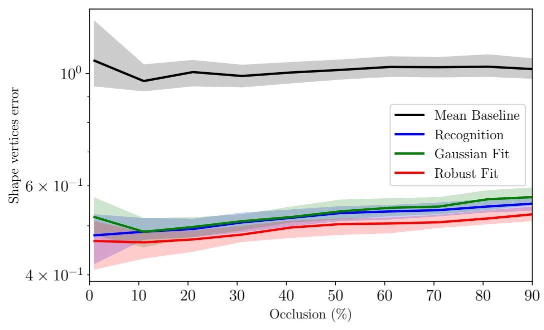
(b) Elevation errors



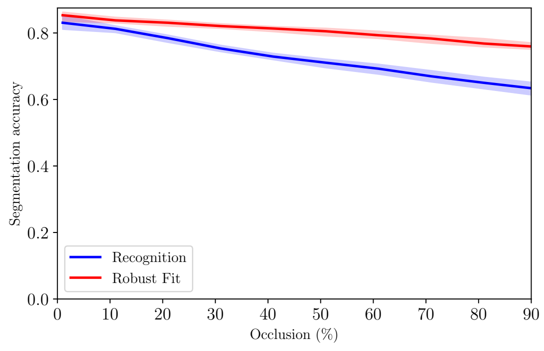
(c) Appearance errors



(d) Illumination errors

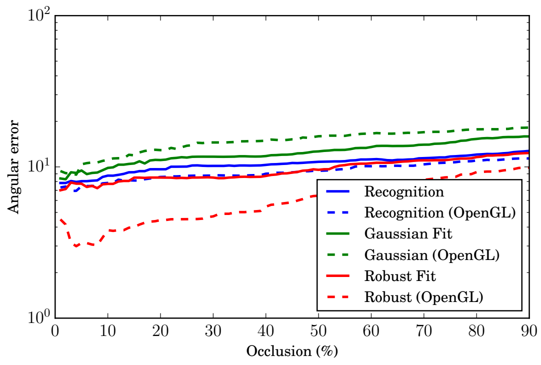


(e) Shape vertices errors

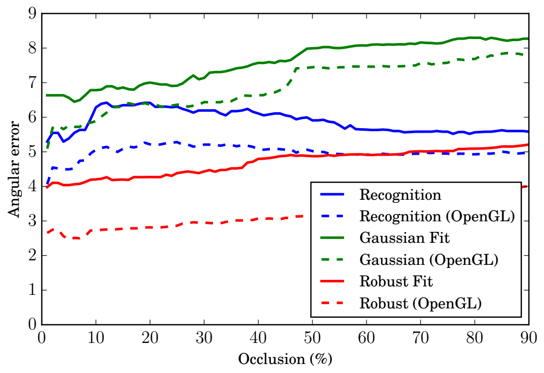


(f) Segmentation accuracy (higher is better)

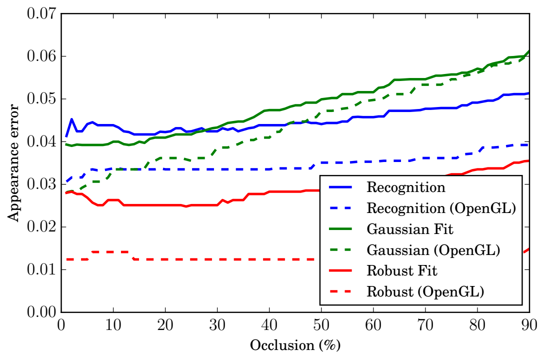
Figure 3.7: Median errors for azimuth, elevation, appearance, illumination, shape; figure (f) is the segmentation accuracy. Shaded areas indicate the lower and upper bounds of the 90% confidence intervals of the median.



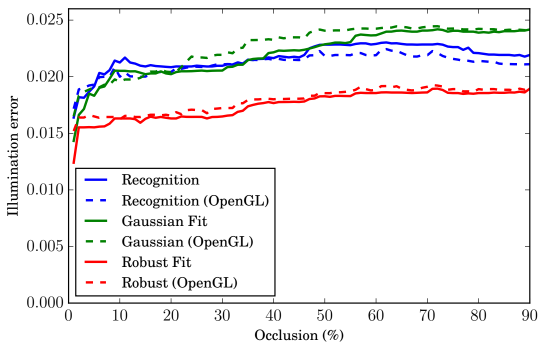
(a) Azimuth errors



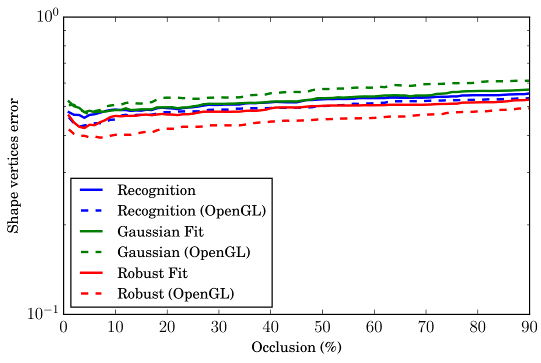
(b) Elevation errors



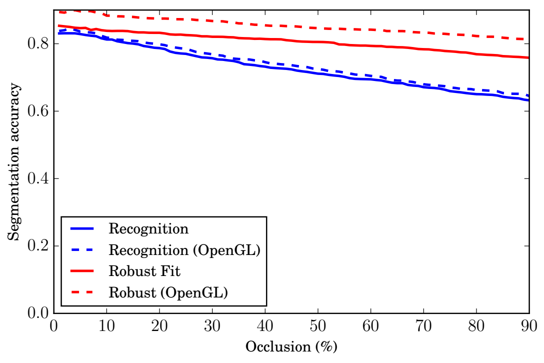
(c) Appearance errors



(d) Illumination errors



(e) Shape vertices errors



(f) Segmentation accuracy (higher is better)

Figure 3.8: Comparison of performance when evaluating on non-photorealistic vs Cycles rendered test images. Median errors for azimuth, elevation, appearance, illumination, shape; Fig. e) is the segmentation accuracy.

		Photorealistic			OpenGL		
	Baseline	RM	Gaussian	Robust	RM	Gaussian	Robust
Azimuth°	92.14	11.71	14.52	**11.17**	10.68	17.18	**8.55**
	88.62,98.11	10.80,12.66	13.18,16.14	10.30,12.37	9.72,11.40	15.76,18.64	7.39,9.66
Elevation°	14.44	5.58	8.19	**5.02**	4.94	7.57	**3.67**
	14.03,15.15	5.20,6.11	7.62,8.69	4.71,5.32	4.58,5.36	6.75,8.14	3.27,4.05
Appearance	0.140	0.048	0.056	**0.032**	0.036	0.055	**0.014**
	0.133,0.147	0.045,0.051	0.052,0.060	0.023,0.035	0.035,0.039	0.050,0.059	0.012,0.014
Illumination	0.021	0.023	0.024	**0.019**	0.022	0.024	**0.019**
	0.020,0.023	0.021,0.024	0.022,0.026	0.018,0.020	0.020,0.023	0.022,0.026	0.018,0.021
Shape	1.005	0.541	0.560	**0.511**	0.525	0.599	**0.479**
	0.985,1.075	0.527,0.555	0.534,0.576	0.501,0.527	0.501,0.540	0.575,0.622	0.462,0.494
Segmentation		0.659		**0.778**	0.670		**0.828**
		0.645,0.678		0.764,0.790	0.658,0.688		0.819,0.837

Table 3.1: Median prediction errors for the latent variables for levels of occlusion up to 75%. Higher is better for segmentation evaluation. Lower and upper bounds of the 90% confidence intervals of the median are given in the rows below the median values.

3.6.2.4 Qualitative evaluation.

Given a point estimate of the latent variables we can compute the segmentation mask of the teapot projected on the image as per Eq. 3.11.

Fig. 3.9 shows examples of how the robust model explains away the occlusion rather impressively for different levels of occlusion, which is something difficult to achieve with only bottom-up techniques.

In Fig. 3.10 we visually compare changes in the prediction of the RM and RM plus refinement with the robustified GM. These examples illustrate how the illumination (here represented as an environment map) is refined to capture the main directional sources of illumination in complex indoor illumination settings, even with multiple light sources. Note that all the scene latent variables were predicted and refined in this experiment. In general, the robust model seems much more capable of capturing fine-grained descriptions such as illumination, appearance and occlusions than the recognition model. The last two examples show failure cases when the object is occluded to the extent that the resulting fit becomes completely wrong.

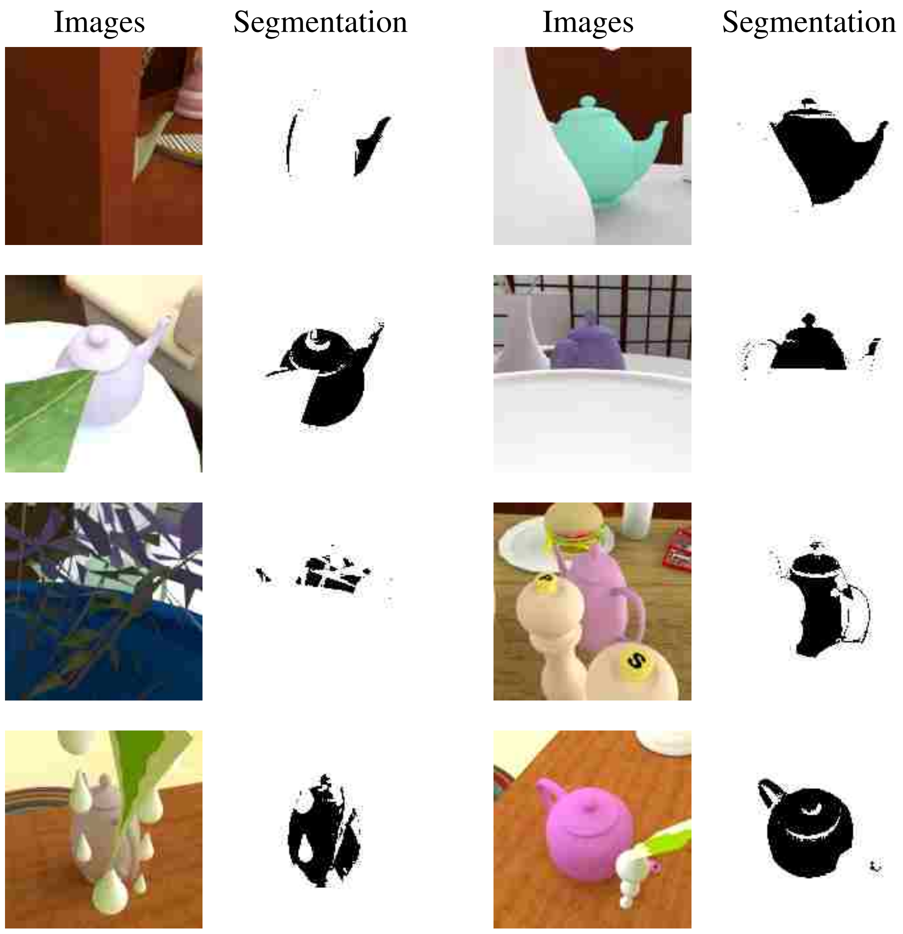


Figure 3.9: Examples of occlusion inference using the posterior of the robust pixel probabilities

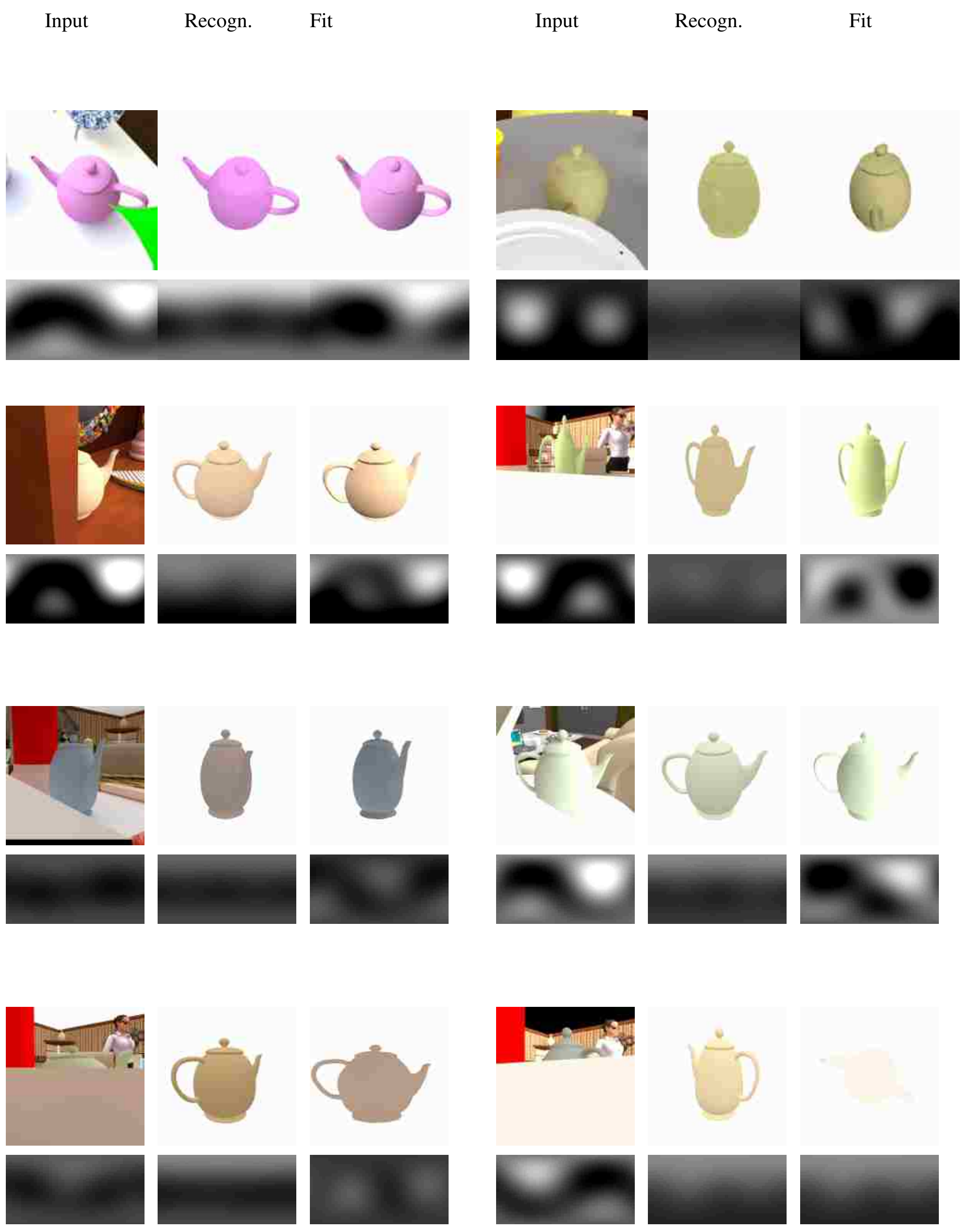


Figure 3.10: Examples of how refinement with the robust model captures the correct illumination whereas RM is unable to do so. For each example image we show at the bottom the corresponding illumination map (in terms of the projection of the 9 Spherical Harmonics coefficients to an equirrectangular image).

3.7 Discussion

Above we have investigated the combined use of vision as inverse graphics with recognition models for a target object embedded in background clutter and subject to occlusions. A great advantage of using data generated by computer graphics is that it allows us complete access to the underlying scene parameters, and hence the ability to explore these systematically. Our synthetic data set includes a large number of factors of variation including pose, shape, appearance, illumination and occlusion, and can complement datasets of real images for scene understanding tasks.

Our results show that some of the latent variables like pose and shape are well-predicted by the recognition network, while others such as illumination, appearance and occlusion benefit from subsequent refinement by fitting a robust generative model. Our results also show that the robustified model of Eq. 3.6 clearly outperforms a Gaussian likelihood, and provides a way to detect occlusions even in complicated cases.

Explicitly modelling occlusion in the GM allows us to infer the object segmentation of an image in the presence of complex occlusion as we saw in the some examples of our test set.

We believe the insights gained in our systematic study of RM vs RM+GM when predicting different factors of variability (i.e. RM works well with pose and shape, and GM significantly improves appearance and illumination) can help the research community design methods tailored to the relevant factors specific to many scene understanding tasks.

Future directions for research include investigating: detecting if the fitting process has fallen into improbable basins of attraction, and the use of multi-modal predictions in the recognition network (as per Jacobs et al. (1991) or Guzman-Rivera et al. (2014)). We believe that improving the realism of the synthetic dataset as well as GM with richer reflection variations (see e.g. Barron and Malik (2015); Hara et al. (2005)) is also an interesting direction for future work.

Chapter 4

Inverse graphics for scenes with multiple objects

4.1 Introduction

In the previous chapter we studied the effects of occlusion when modelling a single centered object. A more general purpose solution cannot assume that a single object of interest has been detected and positioned exactly at the center of the field of view of the camera. In this chapter we apply our inverse graphics solution in scenes where multiple objects are not centered. While the effect of occlusion is the main focus of Chapter 3, here we are interested in addressing the challenges of modelling and fitting (i.e. refining) a generative model to scenes with multiple objects. Intuitively, the predictive performance of a generative model is improved when more elements of the underlying process are accounted for, both in terms of the objects in the scene as well as the realism of the rendering (e.g. presence of shadows and different types of reflectance functions). This is joint work with my PhD colleague Lukasz Romaszko and the details can be found in the conference article *Vision-as-inverse-graphics: Obtaining a rich 3d explanation of a scene from a single image* (Romaszko et al., 2017). To clarify the extent of our contributions:

- I provided the initial script for generating synthetic images in which multiple objects are instantiated randomly in a scene using the Blender (Blender Online Community, 2017) software. Similarly to our ground truth dataset in Chapter 4, these scenes serve as data to train the recognition models as well as a test set to quantitatively evaluate alternative solutions. Lukasz significantly improved the scene generation code to include randomized images of indoor scenes as background (actual 3D indoor scenes are not used here), a plane on which the objects lie, as well as improvements in the specification of the scene geometry and efficiency of the objects random instantiations.

- The main focus in Romaszko et al. (2017) is a novel method to aggregate predictions from object-dependent recognition models when predicting global variables such as the extrinsic parameters of the camera. This work was carried out by Lukasz. In Section 4.3, we explain the architecture of these models.

- The article shows qualitative results when refining both synthetic and real image scenes with our inverse graphics solution. This was my main contribution to the work and we have extended these experiments with a quantitative evaluation, which is the main focus of this chapter. A number of key changes in our generative model and refinement process are proposed in order to improve the estimation of the latent variables of these scenes. We explain them in detail in sections 4.3.4 and 4.3.3.

Specifying a scene with multiple objects involves a series of technical changes in terms of the geometry and layout of the objects with respect to the camera. In Sec. 4.2, we specify in detail the scene generator used to train our recognition models in a supervised learning framework. In Sec. 4.3, we describe the recognition and generative models used as well as the strategy for refining the latent variables of the scenes given the estimates of the recognition model. In Sec. 4.4, a test set of held-out images is used for quantitative evaluation of the performance of our proposed inverse graphics solution. We also provide a qualitative evaluation on real images with multiple objects.

4.2 Stochastic scene generation

The new scenes generated for this work are significantly different from Chapter 3. To begin with, the object class used here is *mug* and not *teapot*, partly because mugs are more often seen in varying number of quantity, and also because their reduced size makes it easier to instantiate a larger number of them in a single scene. We do not use a parametric shape model like previously, but instead a discrete representation of 15 different mugs obtained from ShapeNet5 (Chang et al., 2015).

Instead of using 3D indoor scenes as in the previous chapter for increased realism in background and clutter, real indoor scene images from NYU Depth V2 dataset are used as backgrounds. Mugs are randomly placed on top of a rectangular (ground or table) plane. The object and plane colours are sampled from a uniform distribution on each of the RGB channels. Random texture patterns are also used to increase the richness in the appearance of the objects. Thus, the object-dependant latent variables are the objects' position on the plane, size (mug's diameter), rotation around the up axis relative to the camera (i.e. object azimuthal rotation), and their RGB color values.

The camera parameters and illumination are referred to as the global latent variables of the scene. The camera is always placed at the origin of the XZ plane at a height $Y = h$. The camera's tilt α is the angle of elevation, and the focal length is denoted by f. Fig. 4.1 illustrates the camera setting relative to the objects in the scene. In Figure 4.2 we show examples of images generated. Finally, illumination is modelled as a uniform light and directional light source from an infinite distance, which is composed of the directional light intensity, and the azimuth and elevation angles of incidence into the scene. Scenes are rendered into 256x256 images using Bender, including Lambertian and specular shading, as well as cast shadows. Note that these scenes are not rendered with photorealistic illumination using Blender Cycles but with the approximate renderer of Blender Internal. The reason Blender Internal is used here is to produce more realistic scenes than our GM since it models self shadows and shadows cast across objects, as well as specular illumination. These scenes are not fully photorealistic in that it is not a path-tracing method like Cycles, thus it is faster to render a large quantity of images needed to train the recognition models.

The camera parameters are sampled within the following ranges: field-of-view $\omega \in [20,60]$ (which relates to the focal length as per Eq. 2.8), camera elevation $\alpha \in [0,90]$, and camera height $h \in [0,150]$ cm. The uniform light has a strength within the range of $[0,1]$ and the directional light intensity within $[0,3]$. Up to seven objects are sampled sequentially using one out of the 15 types of shapes, a diameter sampled uniformly within the range of $[8.0,10.4]$ cm, and RGB intensity values uniformly sampled between $[0,1]$. For each mug, a random texture is chosen from a collection of 200 grayscale texture patterns (e.g. stripes, marbled, etc.). This texture is multiplied with the object colours to produce the final colour of the mug. When a scene is sampled, we reject the sample if certain plausibility conditions are not met (i.e. if sampled mugs collide with each other, or if a mug is occluded or only partially visible by less than 50% of its projection on the image).

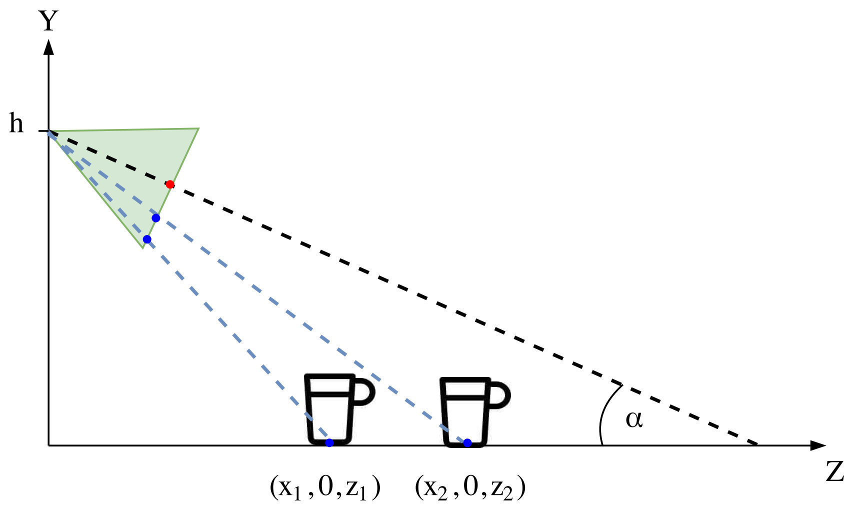


Figure 4.1: Illustration of our scene set up.

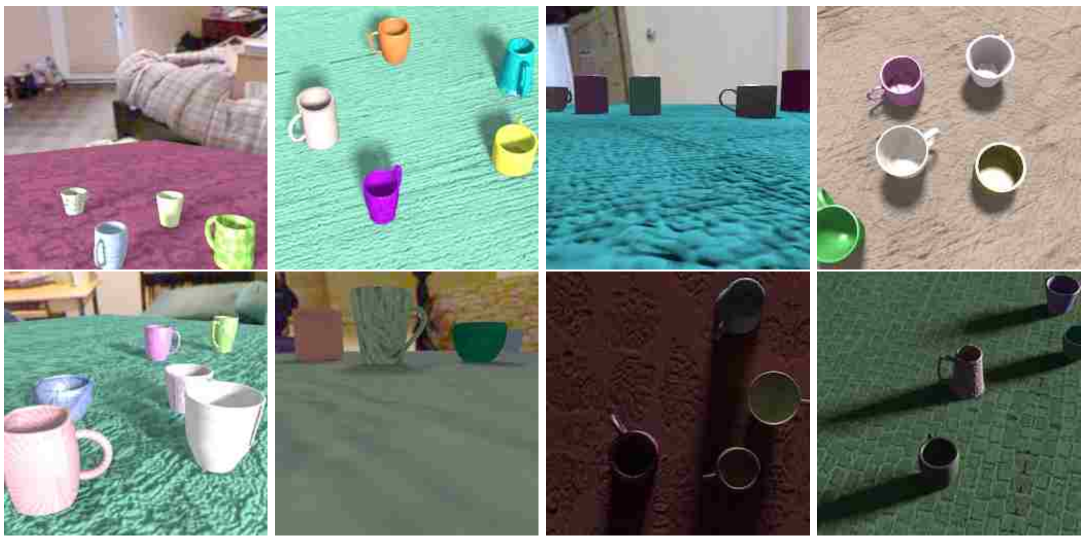


Figure 4.2: Examples of our dataset rendered with the scene generator.

4.3 Scene models

Here, we follow the same inverse graphics method of using a recognition model to give us a point estimate of the scene latent variables, followed by local search to improve the fit using the generative model as in Chapter 3. In the next section we briefly explain the recognition models used, which are an improvement on the purely Convolutional Neural Network (CNN) architecture in order to obtain more accurate estimates of the latent variables when multiple objects are present. Again, our modern implementation of OpenDR is used as a graphical renderer under the same assumption of a Lambertian reflectance function. Improvements in the rendering of the scene are proposed and we explain them in Section 4.3.2.

4.3.1 Recognition models

To train the recognition models explained below, a total of 7,000 images from our stochastic scene generator are used. The recognition model follows the pipeline of **object detection** to obtain confidence maps of the regions where objects are found, non-maximum suppression (NMS) to reduce the set of detection to a sparse set, followed by applying global and object **latent variable predictors** on the resulting patches of the sparse set. The details of these two stages are as follows:

Object detector. A per-class object detector is trained in a sliding-window fashion. This object detector is trained to predict the probability of the class of the object (in

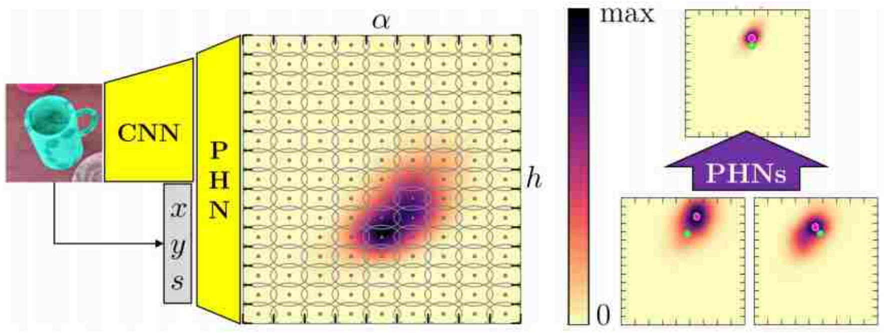


Figure 4.3: Probabilistic HoughNets framework. A patch with a detected object is fed to a CNN. The representation of its last layer, along with the image position (x,y) and size s of the detected patch is then used as the input to the PHN. Each PHN casts a vote on the camera *Hough space* (α,h) (conditioned on ω) via a mixture of Gaussians (dots represent their means and circles their standard deviation). Finally, when multiple objects are detected in an image, each of these PHN predictions are combined to produce a robust prediction of the scene camera parameters. Figure from Romaszko et al. (2017).

this case, mug) present in the window patch. To do so we divide the training set into half positive and half negative (random crops without the object class present) training examples. Furthermore the detector also predicts the *projection scale* (the pixel-wise extent of the object relative to the image). The resulting probability heat map is processed via NMS, and the sparse set of detected patches, along with their predicted projection scales are fed into the latent variable predictors as explained below.

Latent variable predictors. Here we distinguish two types of architectures: the Probabilistic HoughNets (PHN) used to predict camera latent variables of height, elevation angle, and focal length; and CNNs to predict the remaining latent variables, including the class of the 15 shapes of mugs. A novel architecture, referred to as the Probabilistic HoughNet, is proposed in order to estimate global variables of the camera. In general, predicting both the camera distance from an object and the focal length is a difficult problem.

The thin lens equation is $\frac{1}{o_d}+\frac{1}{i_d}=\frac{1}{f}$, where o_d is the object distance, and i_d is the image distance. Due to this focal length – camera distance ambiguity, any RM that predicts these variables are prone to produce inaccurate estimates. PHNs address this problem by robustly aggregating votes on camera parameter hypotheses from each of the objects detected in the image, as is typically done with the Hough transform. In the Hough transform, multiple voting elements cast a vote in a binned (i.e. discretized) space. Here the bins are represented probabilistically using a Gaussian mixture model (GMM), where a Gaussian probability distribution is fixed in the center of each bin. The output of the PHN is thus the mixture coefficients probabilities of the GMM. Figure 4.3 illustrates the PHN framework to produce robust camera predictions. The global variable predictions per object are combined by taking the product of the GMM coefficients along each component, and the component with a maximum density is chosen via hill-climbing search. The main advantage of producing independent predictions per object in the scene is two-fold. First, the final prediction is more robust to outliers (e.g. implausible camera estimation). Second, it is a natural way to design a recognition model architecture that can generalize to arbitrary number of objects in an image, since the voting procedure acts on a set of any size. For the remaining object latent variables as well as illumination variables, standard CNNs are used with the following architecture.

All of the CNN architectures are based on the VGG-16 (Simonyan and Zisserman, 2015). The first 13 convolutional layers are used and kept fixed without the last two max-pooling layers to preserve more spatial information. Three convolutional layers are learned on top of the 13th layer followed by a fully connected layer and the output layer. These layers are not shared for different object latent variable predictions. The prediction of a color of an object is done with a 3 layer CNN. The hyperbolic tangent function *tanh* is used as the nonlinearity of all the networks. Again, the dataset described in Sec. 4.2 was used for training all the networks (but for the latent variable predictors, we use the detected patches cropped at 4245 pixels pixels from the center of the detection).

4.3.2 Generative models

Given the recognition model estimates, we instantiate a scene using our generative model with the plane, illumination, camera and object latent variables. The purpose of this work is to enable generative models to be a practical inverse graphics tool for images with multiple objects. This involves having a generative model that matches the ground truth generative processes (e.g. real images or images generated with other graphics engines like Blender) as much as possible.

Our initial model does not account for cast shadows and specular illumination. Cast shadows can significantly affect the rendered pixel colors especially when objects are strongly concave, such as mugs. We propose a pre-processing step of the 3D meshes that captures the self shadows within the objects, and include it in the generative model. This method is described in Sec. 4.3.3.

As we saw in the previous chapter, derivatives with respect to the shape, pose and position of objects are not exact when using the OpenDR framework and often do not work very well. Since we perform gradient based local refinement, having accurate derivatives is a crucial step. We analyse the problem that arises with these derivatives and propose improvements in the rendering function that make them more accurate. These improvements are described Sec. 4.3.4.

4.3.3 Modelling object self shadows

A visual inspection of our generated scenes in Fig. 4.2 suggests that shadows are an important factor in the final appearance of the images as well as strong cues of the directional lights. Thus, a model that generates an image without any shadows can limit its ability to accurately fit to a scene.

Correct rendering of illumination and shadowing requires ray tracing algorithms that take into account the global scene geometry as light rays travel through the scene. These methods are rarely used when the speed of computation is a priority, such as in video-games as well as in the case of inverse graphics. For instance, proper rendering of a cast shadow depends on the position of the light source, the object onto which the shadow is cast, and the objects that lie in between. Instead, efficient but approximate shadowing techniques are widely used in computer graphics, such as shadow mapping (Williams, 1978) and shadow volume (Crow, 1977). However, these techniques require re-computing shadowing when the scene geometry/lighting changes, and are also non-differentiable.

We propose using an existing method, Precomputed Radiance Transfer (PRT Sloan et al., 2002), which does most of the illumination computation in a pre-processing step and provides approximately differentiable shadows when rendering. To our knowledge, PRT had not been used in the context of applying generative model of images for the task of inverse graphics in the literature except for the recent independent work of Schneider et al. (2017).

For each vertex i in the mesh, there is an implicit binary function of both the incoming light source angles of azimuth ϕ and elevation and θ, as well as the object shape z_s, i.e. $s_i = f_i(\phi, \theta, z_s)$. For a given shape, this function is 1 if the geometry is not in the way between the vertex and the light source and 0 otherwise. Fig. 4.4 illustrates this function where green rays are set to 1 and red set to 0. The idea is to project this spherical function s_i to a Spherical Harmonic (SH) basis giving us N coefficients (in our case $N = 9$) per vertex in the mesh. With these coefficients we can efficiently compute a continuous and differentiable shadow variable for any given light source direction. For each mesh with V vertices we then have a $V \times 9$ matrix of SH coefficients. Thus, this method can be seen as a special case of PRT in which we assume a directional light instead of also modelling the incoming illumination with a SH basis in combination with f_i (as in the general PRT method).

Schneider et al. (2017) take this idea a step further by learning a linear mapping from shape parameters to the pre-computed coefficients (also known as the transfer matrix T_i) which enables efficient computation of new PRT coefficients when the shape changes. Their work can thus be seen as an extension of the same idea, and can be included in our solution when modelling objects with deformable geometry (e.g. faces). Note that PRT is used here to model self shadowing, so it is not a complete general purpose shadow model as it does not compute shadows across different objects (e.g. the shadow of the mug onto the plane). That said, it is particularly well suited for heavily concave objects such as mugs.

Fig. 4.5 compares the images from our dataset with the corresponding renders with and without our SH model of self shadows. The shadows within the mugs are noticeably more similar to the ground truth images, even if they do not capture the exact sharp shadow transitions due to the SH representation with 9 coefficients. One could improve the accuracy of the representation by using a larger number of coefficients when precomputing the coefficients as well as rendering the shadows at inference time, however that would come at the expense of computation time.

4.3.4 Improving the render function gradients

Having approximate gradients can hinder the refinement process for variables that modify the position of the meshes in the 3D space. We saw this occur in Chapter 3 with the shape and pose latent variables of the teapot model. We could try using

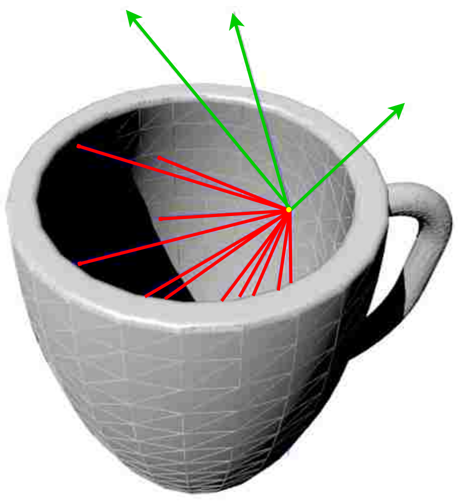
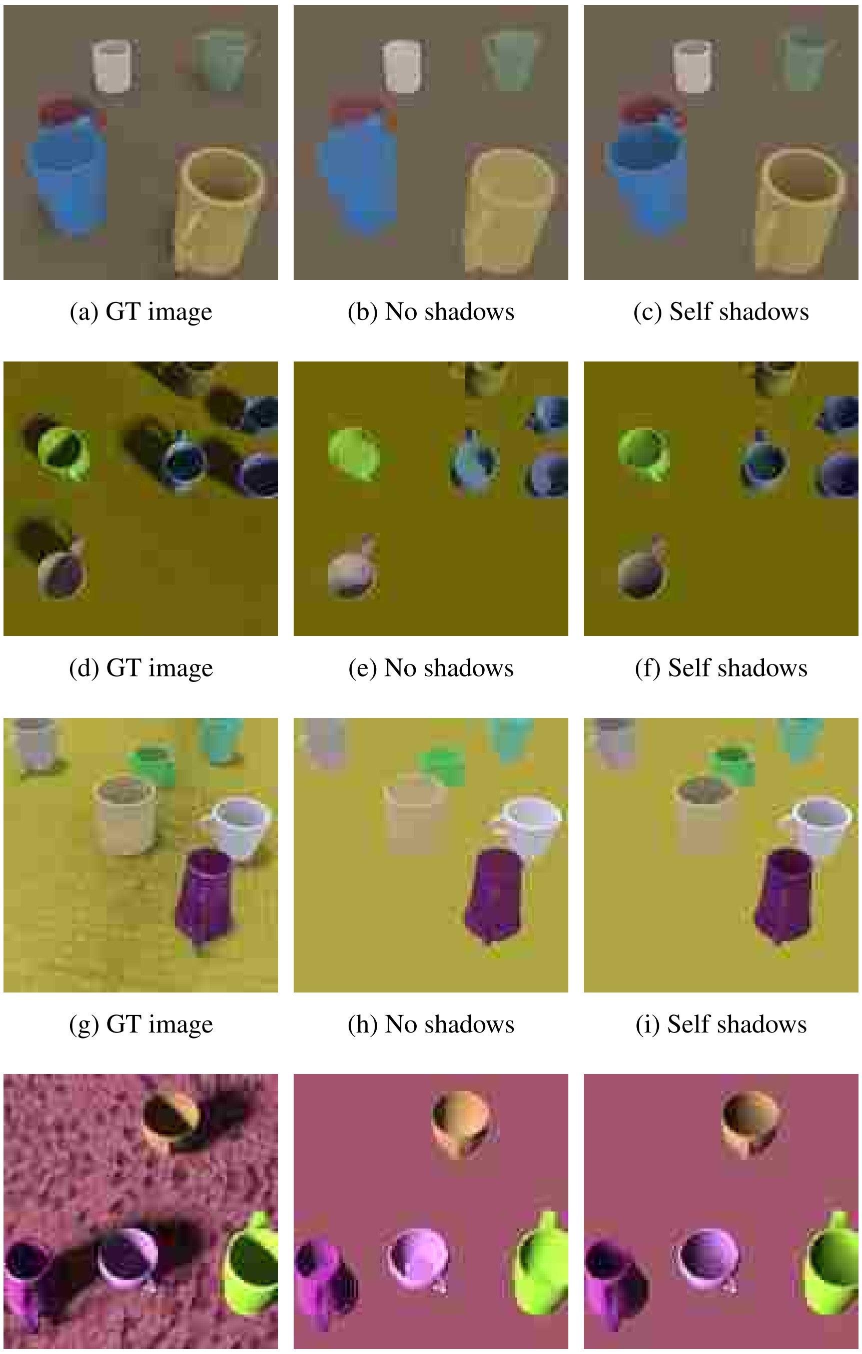


Figure 4.4: For each vertex (e.g. yellow dot) we uniformly evaluate spherical directions and store whether each direction intersects with a part of the mesh (red lines) or not (green lines). This binary function is projected to a Spherical Harmonics basis in a pre-processing step.

alternative methods instead of gradient-based local search. For example, we can use approximate Bayesian computation (ABC) to produce posterior samples of a scene as in Mansinghka et al. (2013), or use different combination of proposal distributions when searching the posterior space with Markov Chain Monte Carlo (Jampani et al., 2015). In the optimization literature there are also global optimization methods such as particle swarm optimization (Kennedy and Eberhart, 1995) and Bayesian Optimization (Mockus, 1974) that can be used. Most of these methods often do not scale well to scenes with a large numbers of latent variables (e.g. when there are many objects). In any case, we should note that these methods are not mutually exclusive with the local refinement method used here and can be included to strengthen the overall solution. Before we consider including these methods, however, we identify the issues of the approximate gradients and propose ways to overcome some of their limitations.

The main questions we want to answer are: how much do the approximate gradients of a cost function differ from the analytical gradients, and what causes such differences? Given that the cost function we use is defined in a pixel-wise basis, we can focus this analysis on the gradients of a single channel (e.g. red or RGB) of a given pixel of the image. As we expect, the main sources of errors happen in those pixels at the edges of objects (i.e. where objects start occluding other objects and sharp changes in image


(a) GT image (b) No shadows (c) Self shadows

(d) GT image (e) No shadows (f) Self shadows

(g) GT image (h) No shadows (i) Self shadows

(j) GT image (k) No shadows (l) Self shadows

Figure 4.5: Examples of images modelled with and without cast shadows.

intensities arise). The cost function at pixel i can be seen as the likelihood function of the ground truth image (I^{GT}) and the render (I^R). For the sake of simplicity we consider the squared error cost function as an example, i.e. $E_i = (I_i^R - I_i^{GT})^2$. The main quantity of interest is thus the residual $R_i = I_i^{GT} - I_i^R$. The gradient then becomes

$$\frac{\partial E_i}{\partial z} = 2(I_i^R - I_i^{GT}) \cdot \frac{\partial I_i^R}{\partial z}$$

The first source of error comes from the fact that the cost function is defined in the discretized image domain instead of the underlying continuous domain, i.e. only one intensity is chosen for the whole area of a given pixel, which happens during the rasterization step of the rendering process as we described in Sec. 2.2.1.4. If the residual is 0 (e.g. the foreground object in the render and GT image have the same intensity) then there will be no gradient signal for that pixel. Fig. 4.6 shows the default rasterization strategy of choosing the intensity of the sample in a triangle (which could be part of a larger mesh) that lies in the center of the pixel. Note that this would not happen in the continuous domain since the cost function would integrate over the whole pixel region. We propose and compare using two alternative rasterization methods of I^R to overcome this issue. The first is to average the intensities over multiple sampled positions of the pixels. This is similar to Monte Carlo estimate of integral of the overall pixel intensity of the pixel. Figure 4.6b illustrates this method. This method can be implemented very easily and efficiently as it is a well known anti-aliasing technique known as multisample antialiasing (MSAA)[1]; most modern graphics libraries such as OpenGL provide it.

The second source of inaccuracy is closely related and is due to the discrepancy between the way the rendered pixel intensity (and thus the residual) changes as a triangle shifts across the pixel, which could be the result of a change in some latent variable z. A small change of this variable Δz should also have a small change in the intensity of the pixel and ideally should correspond with $\frac{\partial I_i^R}{\partial z}$ in the limit where $\Delta z \rightarrow 0$. With the first rasterization method explained above, the intensity will have a discontinuous change when the triangle crosses the centre. With MSAA this effect is still discontinuous but has a set of smaller transitions (i.e. bumps) when the triangle intersects with each sampled position. Inspired by the technique proposed in (de La Gorce et al., 2011, sec. 4.7.3), we propose using an antialiasing technique in which sampled points are weighted by their projected distance to the 2D edge of the triangle that intersects the

[1] `https://www.khronos.org/opengl/wiki/Multisampling`

pixel. The main differences with their work is that we do not operate with analytical gradients of an error function (instead we apply this antialiasing only when rendering the pixel intensity), and we make use of the distance-to-edge weights in a simpler and more efficient way by implementing them with OpenGL GPU shaders. In other words, we retain the flexibility and efficiency of OpenDR while overcoming some of the main issues of approximate gradients. Samples closer to the edge will have progressively less weight which makes the transition completely smooth, as illustrated in Figure 4.7. We call this method Distance Sample antialiasing (DSAA). In the computer graphics domain, similar anti-aliasing techniques that use triangle edge intersection have been used, e.g. Persson (2011).

Figure 4.7 empirically shows the changes in pixel intensity as a triangle (with intensity of 0.3) is shifted until it fully overlaps with the pixel. There is no built-in function in OpenGL for this type of custom anti-aliasing, but we take advantage of our modern OpenGL implementation of OpenDR which allows us to write programmable shaders (which by default run in GPU) from which these sub-pixel samples are retrieved and the distances computed. To retrieve these values we use the special OpenGL function *texelFetch*[2].

To understand how these different rasterization techniques affect the gradients, we show in Fig. 4.8 how the negative log-likelihood (NLL) of an image of our dataset changes as the size of one mug is changed from its ground truth value (green line). Notice how without any edge antialiasing the NLL is noisier and the gradients (arrows) often point in the wrong direction. This last observed issue can severely affect the performance of the inverse graphics framework since incorrect image gradients makes the refinement process (which uses gradient-based local search) result in sub-optimal solutions.

A simple example of a failure case is shown in Fig. 4.9, in which a square fails to be fitted into its original size, (a), starting from (b), using our generative module due to incorrect gradients when not using antialiasing (c). Using either of the proposed antialising techniques correctly overcomes this issue as shown in (d) and (e). When using MSAA the gradients directions are generally correct and when using the DSAA method the function is even smoother due to having continuous pixel intensity transitions. In Sec. 4.4 we evaluate whether these improvements have a quantitative effect when refining the latent variables to the images of a test set.

[2] https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/texelFetch.xhtml

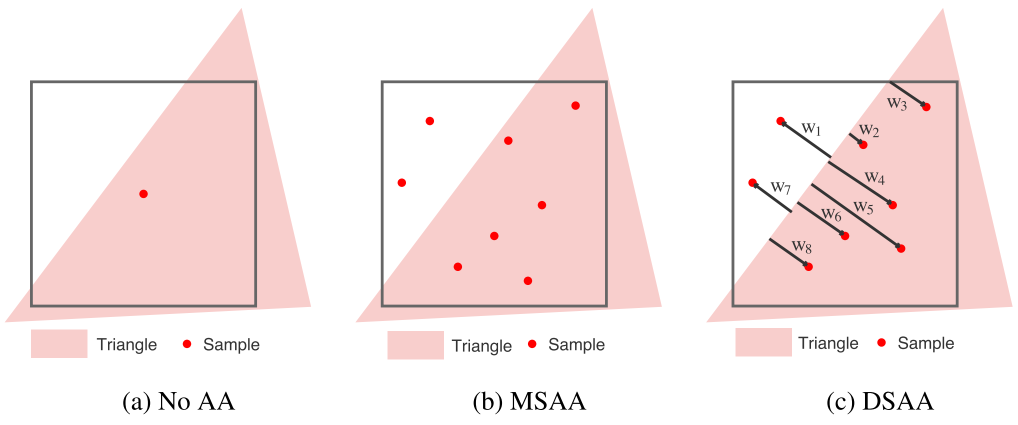


Figure 4.6: Antialiasing strategies when rendering the color of a pixel. Red dots indicate the position in the pixel where color is sampled with respect to the mesh triangle intersected. In (b), the colors at the sample points are averaged. In (c) the average is weighted according to the distances to the triangle edge. Throughout this work we chose to use 8 samples for the multi-sample strategies as it is well supported by modern GPUs and no further advantage was found when using more (e.g. 16 samples).

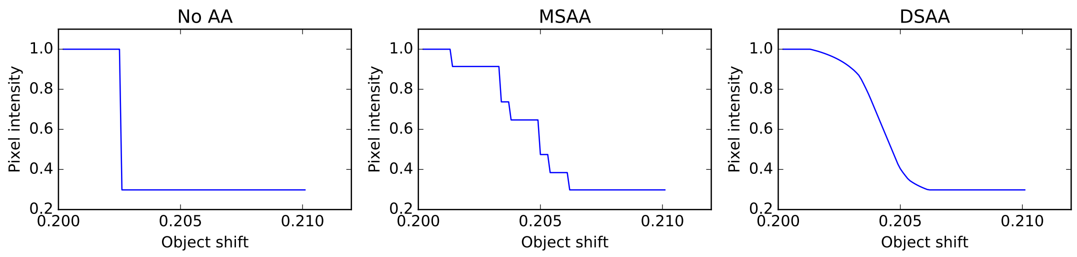


Figure 4.7: Pixel intensity changes as a foreground object (with intensity of 0.3) shifts into the pixel.

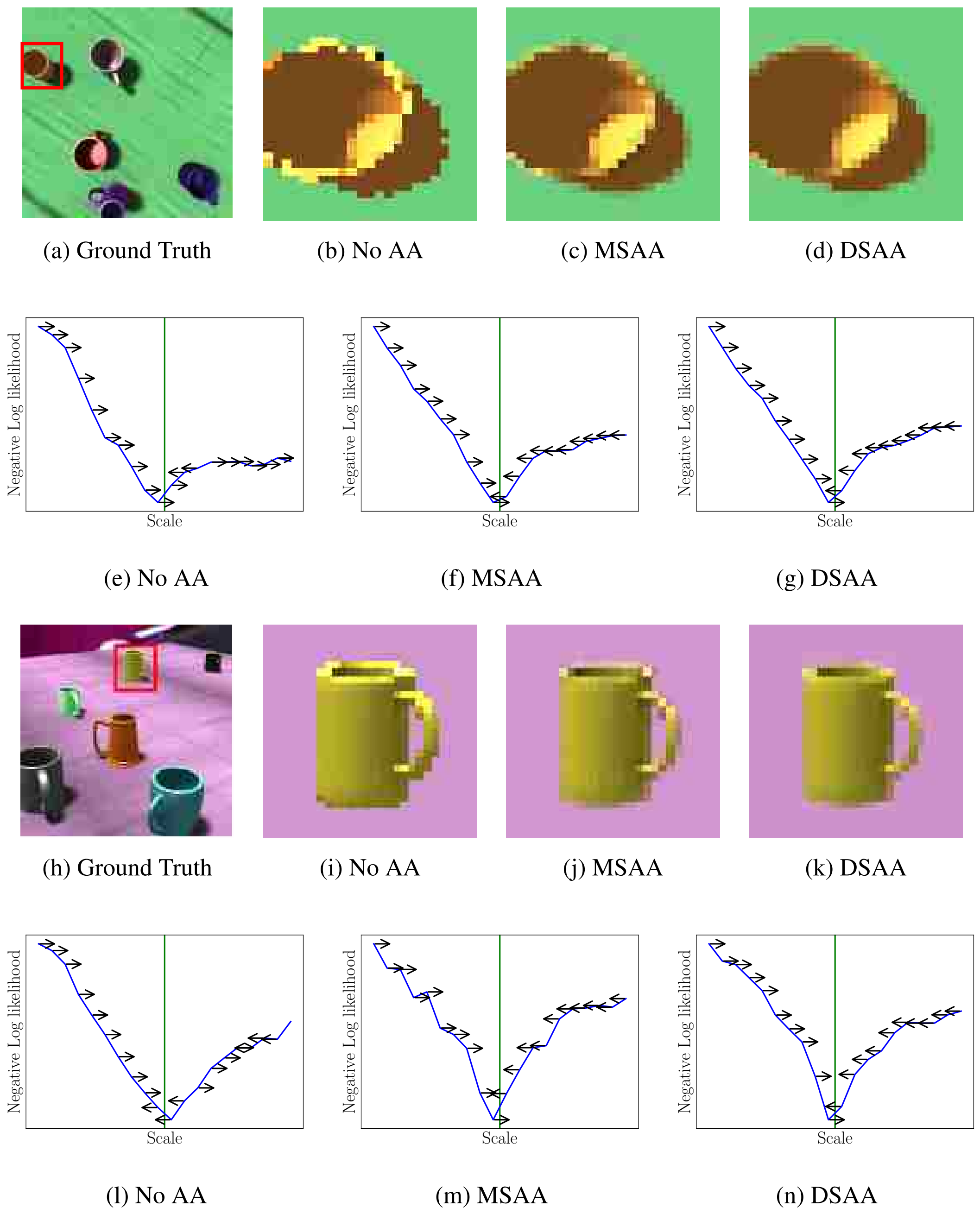


Figure 4.8: Evaluation of the whole image NLL as a function of the 3D scale of the object marked in red for OpenDR renders with different antialiasing methods. The ground truth value of the scale is the green vertical line. Arrows show the direction OpenDR gradients of the log-likelihood as per the different anti-aliasing methods (magnitudes of the gradients are normalized).

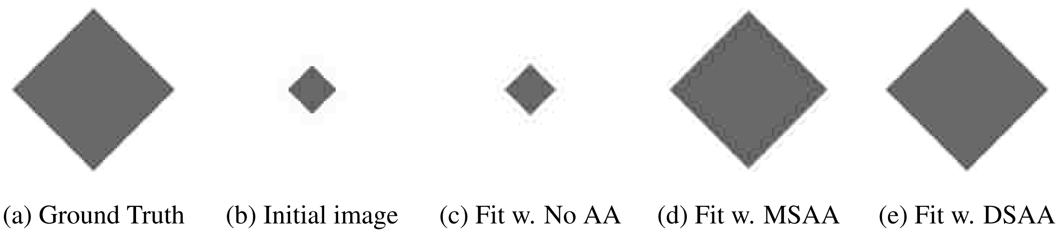


Figure 4.9: Simple failure case of refining without antialiasing. The GT cube is rendered to image (a), and we maximize a pixel-wise Gaussian likelihood function with respect to a rendered image initialized in (b). Final results after refinement using different methods as shown in (c), (d) and (e).

4.3.5 Optimization strategy

When fitting (i.e. refining) the GM to the images there are a number of design choices that need to be made as we established in Sec. 3.6. Most of the procedure we use here is the same as when fitting the scenes with a single object. Even though accounting for occlusion is not the focus here, we still compare the Gaussian with the robust likelihood functions. There are still good reasons to use the robust likelihood function. First, the robust function can help not only in the presence of unexplained occluding objects (as in Chapter 3), but also when certain aspects of the image generation process are not modelled systematically (e.g. cast shadows across objects, or textures).

We use the same setting of 0.9 mixture probability of the Gaussian and uniform components of the robust model. The standard deviation of the pixel-wise Gaussian is again set to 0.1 (pixel intensities range between $[0,1]$) throughout all the refinement procedure. The choice of these settings was made by evaluating the fit on a validation set.

Block coordinate descent also worked better than optimizing all the latent variables at once. In other words, different groups of object latent variables are fit in turn to minimize the negative log likelihood (NLL). The main reason why such an approach works well is that the latent variables across objects are to a certain extent independent (with the exception of cases in which some objects are occluding others). Furthermore, latent variables that relate to surface appearance (object colors and illumination) and those that affect vertex positions (i.e. size, rotation and position of objects as well as camera height and elevation) are also minimized in separate steps. Combining exact gradients (due to appearance and illumination) and approximate gradients (due to vertex changes) does not work well with the optimization methods we have tried.

For the minimization method of the NLL, we use the Truncated Netwon Method (Nash, 1984) as empirically it optimized the shape latent variables better on the validation set.

To sum up, the RM predictions plus GM refinement pipeline for one scene is the following:

1. Initialize scene with the object detection and recognition model predictions.
2. Refine global variables of uniform illumination and ground plane colour.
3. For each object:
    (a) Refine object RGB parameters
    (b) Refine object object size, pose and position.
4. Final refinement of all illumination variables inc. directional light.
5. Refine camera variables of height and elevation.

Note that focal length is not part of the optimization process as it is a difficult parameter to fit as its effect on the image is difficult to disentangle from the physical distance of the objects with respect to the camera. We thus fix same focal length as given by the prediction of the recognition model.

At the end of a refinement step, a scene configuration can be found to be implausible or simply farther from the ground truth than the initial estimates. Since we are not specifying a perfect generative model of images there may be some corner cases, such as in the presence of strong shadows or highly specular reflectance, in which worse values of latent variables can give better NLL. When such cases happen, it is desirable (if possible) to detect them and revert the latent variables back to the initial estimate (as given by the recognition models) and assume that to be the better hypothesis. We can use our knowledge of the latent 3D structure to detect some of these failure cases.

To do so we use various thresholds on the deviation from the initial predictions $\Delta z = z^{Rec} - z^{Fit}$ that we found to be suitable based on a validation set. These are obtained empirically and used to detect when refinement has most probably gone wrong. More specifically, for the refinement of objects we use a threshold value of 10 cm for 3D position (i.e. if the object has been moved for more than 10 cm in any axis), 0.5 of the scale in each axis, 25 degrees for azimuth rotation, and 0.5 for each RGB channel of the object color. For illumination we use a threshold of 15 degrees of azimuth and elevation, and 0.5 for directional intensity and uniform light. For camera variables we use thresholds of 10 degrees for elevation and 10 cm for height. Finally, having mugs be very close to each other so that they intersect is also detected and used to revert the changes in the latent variables of the object that intersected.

4.4 Experiments

We wish to quantify the gains in predictive performance when refining the models to the scenes. To do so we use 157 images of the test set for our quantitative evaluation. Both the robust and Gaussian likelihood models are tested as we did in the previous chapter but we also compare the effects of our proposed improvements to the generative modelling of the scenes. We will refer to Gaussian (DSAA) and Robust (DSAA) as the methods that use the pre-computed self shadows and the distance based antialiasing method. Robust (no self shadow) also uses DSAA but it is the only method in which self shadows are not modelled. Significance tests are also carried out in order to find the statistical significance of these results. Finally, we show qualitative results on real images to test our complete inverse graphics system.

4.4.1 Error metrics

The NLL alone does not give us a sense of how well our models infer the scene latent variables. The desired error metrics should be informative of how much the ground truth and predicted scenes match in latent variable space. Error metrics that directly operate on absolute 3D position and shape values of objects are difficult to use as these are highly entangled with camera position relative to the objects. Thus, the main metric we use to measure the 3D match is by computing the projected intersection over union (which we refer as Projected IoU, which is the sum of the pixels that belong to the predicted object, divided by the sum of the union of pixels that belong to the object and the pixels that belong to the ground truth object) of the 2D segmentations of the ground truth and rendered objects. Note that it is the only metric we report in which higher is better, where 1 indicates perfect overlap, and 0, none. This metric naturally accounts for position, scale and pose of the objects relative to the camera.

We also report a unified metric to measure illumination in a scene: a 3D sphere is sampled in the center of the scene and the scale-invariant MSE (see Eq. 3.9) of the incoming illumination (uniform plus directional) into all its vertices is computed. We refer to this metric as MSE of illumination sphere (MSE-IS). For the colors of the plane and objects we use the scale-invariant Euclidean norm of the RGB channels. As we noted in Chapter 3, the multiplicative interaction between illumination and appearance when modelling the reflectance of objects introduces a problem when evaluating them separately; using scale-invariant metrics is one way to overcome this issue. Note that the values of RGB range between $[0,1]$. For unintuitive physical quantities like the MSE-IS and colour errors defined above, we also provide **baselines** to be used as reference. For illumination, we compute the MSE-IS of the ground truth values against a sphere illuminated with (0.7 uniform intensity) and a single directional light source coming from above with 0.8 light intensity; the idea is to assume a fixed generic lighting condition. For the color of objects baseline we compare the ground truth with a fixed gray color (RGB values of 0.5).

We report the absolute value of the angular difference for the latent variables of object azimuthal rotation as well as the elevation of the camera (the units shown are in angles and not radians). The camera height error is simply the absolute error of predicted and ground truth height values. Note that false detections of the object detection stage are ignored (we don't use them in these experiments).

4.4.2 Results and discussion

Table 4.1 shows the median errors for the object latent variables, while the global latent variables errors are shown in Table 4.2.

The first noticeable result in Table 4.1 is that the there is a very clear improvement in the projected IoU of objects when using either antialiasing technique, going up from 0.65 to 0.76 when refining using the Robust (DSAA) method. This indicates that there is a clear improvement when refining the positions and scales of the objects. The appearances of objects also show a clear improvement with respect to the recognition model predictions. The only latent variable that seems to have no improvements regardless of the choice of the refinement method is the object azimuthal rotation. As we saw in the previous chapter, the azimuth rotation error is not easily captured when comparing pixel-wise likelihoods of the GT and predicted objects. Most of the azimuthal information on a mug is in its handle, which indicates that the basin of attraction is very tight.

For the global latent variables errors, we see in Table 4.2 an improvement when refining illumination and especially plane colour (which is one of the easiest variables to fit). However, similarly to what we see when refining the azimuthal rotation of objects, the camera extrinsic parameters of height and elevation seem to remain largely unchanged.

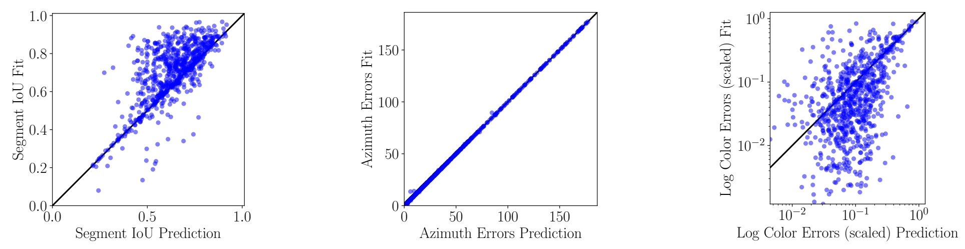
	Projected IoU (%)	Obj Azimuth°	Obj Color
Baseline	-	-	0.301
Recognition	0.65	21.52	0.092
Gaussian (DSAA)	0.77	21.57	0.090
Robust (DSAA)	0.76	21.41	0.057
Robust (DSAA) Simplex	0.76	22.68	0.060
Robust (MSAA)	0.74	22.35	0.060
Robust (no AA)	0.68	21.70	0.070
Robust (no self shadow)	0.75	22.19	0.056

Table 4.1: Object variables evaluation (median) with Truncated Newton optimizer

	Illumination	Plane Color	Cam El°	Cam Height (m)
Baseline	0.0458	0.248	-	-
Recognition	0.0096	0.042	3.11	0.158
Gaussian (DSAA)	0.0089	0.020	3.31	0.158
Robust (DSAA)	0.0074	0.008	3.22	0.158
Robust (DSAA) Simplex	0.0092	0.017	3.30	0.158
Robust (MSAA)	0.0083	0.007	3.34	0.158
Robust (no AA)	0.0084	0.007	3.42	0.152
Robust (no self shadow)	0.0091	0.008	3.26	0.158

Table 4.2: Global variables evaluation (median) with Truncated Newton optimizer

In Fig. 4.10 we see the gains per object. In Fig. 4.11 we can individually compare the gains per scene for the global variables. Points below the diagonal line indicate better performance after refinement (except for the Projected IoU metric). It is clear that illumination is a difficult variable to infer and optimize as seen by the variability of positive and negative gains with respect to the recognition model prediction. Fig. 4.11a shows that, while on average a large number of scenes show an improved illumination (points below the black line), there are also a number of scenes in which illumination is worse. Plane and object colours are clearly improved after refinement as seen in Fig. 4.11b and 4.10c. In terms of camera parameters, it is clear that these are rarely changed after the refinement step, which is a similar case with the azimuthal rotation of objects. However, the rest of the 3D object latent variables clearly show an improvement when refining as we can see from the Projected IoU scatter plot in Fig. 4.10a.

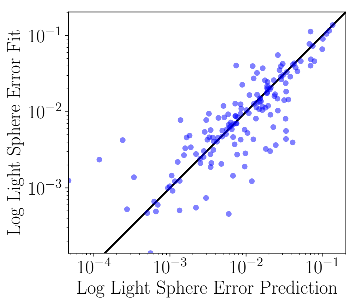


(a) Projected IoU (b) Azimuth (c) Color

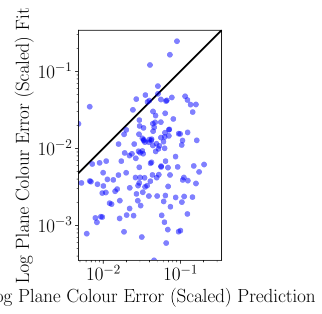
Figure 4.10: Comparison of the errors between recognition model prediction (X axis) vs. refined (Y axis) of the object latent variables. Each dot represents one object in one of the scenes. Points above the diagonal line indicate improvement after refining for the Projected IoU, whereas points below the line indicate improvements for the azimuthal rotation and color errors.

In order to establish the significance of the gains (i.e. decrease in error) after refinement we perform a statistical test on paired differences known as the Wilcoxon signed-rank test (Wilcoxon, 1945). It is a more robust version of the paired T-test, which is suitable when reporting the median average of errors. The idea is to compare the difference between the errors both before refinement (i.e. recognition model predictions) and after refinement to find if the null hypothesis that there is no difference in the results should be rejected. We report both the p-value and the effect size. The effect size s is computed as $s = z/\sqrt{N}$ where z is the z-score of the Wilcoxon test and N is the number of observations. We use the criteria of Cohen (1988) where the effect size is interpreted as follows: 0.1 = small effect, 0.3 = medium effect and 0.5 = large effect.

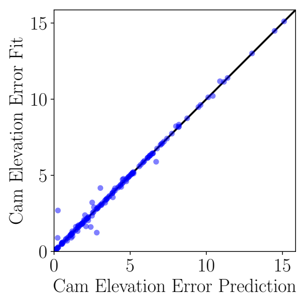
Table 4.3 shows the object latent variables significance tests. To indicate for which latent variables we see a significant gain, we use the following criteria: the columns in which (a) the error differences are statistically significant (i.e. p-value < 0.05), (b)



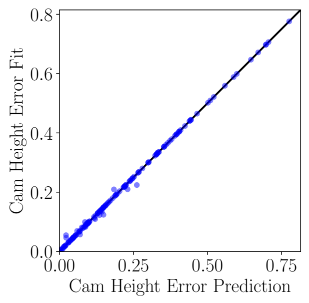
(a) Illumination



(b) Plane color



(c) Camera elevation



(d) Camera height

Figure 4.11: Comparison of the errors between recognition model prediction (X axis) vs. refined (Y axis) of the global latent variables. Each dot represents one scene in the test set. Any point below the diagonal line indicates a positive improvement after refinement.

have an effect size ≥ 0.1, and (c) show improvement for at least 55% or more of the samples. These columns are marked in bold. As expected, the IoU gain is both statistically significant and has a large effect size. A total of 76 % of observations showed improvement. These results suggest that there properly handling the occlusion edges in images is important. It is also encouraging to see that by using these improvements we can use our general purpose pixel-wise likelihood functions, instead of having to rely on commonly used multi-scale error functions based on pyramid representations, as used for instance by Loper and Black (2014). Finally, object colours are also improved with medium effect on 65% of the samples.

In Table 4.4 we can see how the improvements of illumination are statistically significant but have a small effect size, whereas the effect size is large for the plane colour. The camera parameter gains are either statistically not significant or have a very small effect size, which is not surprising given how little they change.

	Projected IoU	Obj Azimuth°	Obj Color
Wilcoxon p-value	<**0.001**	0.001	<**0.001**
Effect size	**0.639**	0.130	**0.324**
Improved samples (%)	**76 %**	48 %	**65 %**

Table 4.3: Object variables evaluation (median) with Truncated Newton optimizer

	Illumination	Plane Color	Cam El°	Cam Height
Wilcoxon p-value	**0.004**	<**0.001**	0.581	0.229
Effect size	**0.115**	**0.405**	0.022	0.068
Improved samples (%)	**59 %**	**94 %**	49 %	50 %

Table 4.4: Global variables refinement effect size and significance using the robust model with DSAA

We are also interested in establishing the difference between the performance with and without the contributions proposed. In Table 4.5 we see a very significant benefit of using antialiasing compared to refinement without using any antialiasing in terms of the projected IoU, though it does not affect the performance with respect to the other variables. If we compare both antialiasing techniques (MSAA vs DSAA) we see a small to medium improvement when using DSAA that is also statistically significant.

If we compare the refinement errors when not modelling self shadows, we see that self shadows improves refinement of object colors. As for the global latent variables, we notice in Table 4.6 that using antialiasing only shows a small benefit when modelling illumination. There seem to be no significant effects when comparing the antialiasing techniques. However, it is clear that modelling self shadows helps better capture the illumination (although an effect size of 0.178 is not very large). This is again consistent with what we expect: modelling self shadows has the main advantage of improved estimation of the illumination as well as object colors.

Fig. 4.12 shows a few examples of the refinement of all the latent variables using the Robust (DSAA) model. We can see how in general, the colors, the objects positions and sizes are noticeably improved to match the ground truth scenes. There are, however, failure cases in which the generative model is not able to obtain a better scene description (or change at all) as seen in the examples 4.12g to 4.12j. In these cases

	Projected IoU	Obj Azimuth°	Obj Color
	No AA vs DSAA		
Wilcoxon p-value	**<0.001**	0.135	0.132
Effect size	**0.657**	0.085	0.085
Improved samples (%)	**71 %**	52 %	53 %
	MSAA vs DSAA		
Wilcoxon p-value	**<0.001**	0.791	0.696
Effect size	**0.226**	0.015	0.030
Improved samples (%)	**55 %**	46 %	47 %
	Including self shadows		
Wilcoxon p-value	0.234	0.933	**<0.001**
Effect size	0.067	0.005	**0.221**
Improved samples (%)	51 %	48 %	**63 %**

Table 4.5: Object variables evaluation (median) with Truncated Newton optimizer

we notice how the background and mugs have similar colours, hence "confusing" the generative model. This seems to be a common failure case.

In the first scene of Fig. 4.13 we can see how using DSAA, as opposed to no antialiasing, is key to having the green and pink mugs fitted properly. A similar situation arises in the second example with a drop of the IoU from 0.71 to 0.52 if antialiasing is not used. We observe similar problems when self shadows are not modelled and the insides of the mugs are not properly darkened, as seen in the first example of Fig. 4.14. We can also see in the example of Fig. 4.14c-4.14d how the illumination is captured better when modelling self shadows since these shadows are affected by the direction and intensity of the directional illumination. This is especially relevant when there is a strong directional lighting in the scene as we see in this test image.

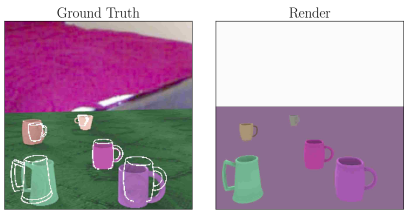
Finally, in Fig. 4.15 we provide a qualitative evaluation of the refinement performance on unlabelled real images using the Robust (DSAA) model. These examples show the ability of our full inverse graphics solution to infer fine-grained descriptions from real images for which it has not been trained. Occasionally, a mug is hallucinated in the scene, or it can miss detecting a mug, but overall it seems to work robustly across different types of real scenes. The main benefits of refinement in these samples come

	Illumination	Plane Color	Cam El°	Cam Height
	No AA vs DSAA			
Wilcoxon p-value	**0.040**	0.546	0.547	0.944
Effect size	**0.116**	0.034	0.034	0.007
Improvement (%)	**59 %**	55 %	51 %	47 %
	MSAA vs DSAA			
Wilcoxon p-value	0.982	0.788	0.716	0.594
Effect size	0.001	0.015	0.021	0.030
Improvement (%)	49 %	50 %	48 %	50 %
	Including self shadows			
Wilcoxon p-value	**0.002**	0.129	0.767	0.614
Effect size	**0.179**	0.086	0.017	0.029
Improved samples (%)	**62 %**	60 %	50 %	55 %

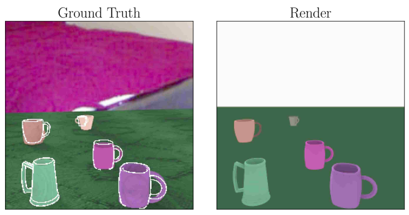
Table 4.6: Global variables refinement effect size and significance using the robust model with DSAA

from improving the object and ground plane colors as well as the position and sizes of the objects. The refinement step can sometimes make mistakes as evidenced in the examples of Fig. 4.15i and 4.15j. Again, we notice how this mistake is caused by having a background colour very similar to the foreground object (black mug). In general, we observe small to medium level of improvements when refining most of the latent variables. While the recognition models have been shown to provide good initial predictions (based on the qualitative analysis provided, as well as on the comparison of the performance with respect to the provided baselines), the improvements obtained by refinement have been shown to be statistically significant, and can be important when the interaction with the environment requires fine-grained precision with regards to the 3D understanding of the position and shape of the objects relative to the scenes (e.g. robotic arm having to operate with delicate objects).

In Fig. 4.16 we show the benefits of inferring a complete scene representation. Given an input image 4.16a, we can render the scene with a more photorealistic renderer such as Blender Cycles as seen in 4.16b. We can infer the depth map as shown in 4.16c, or change the camera, illumination, and object LVs as shown in 4.16d-4.16f.

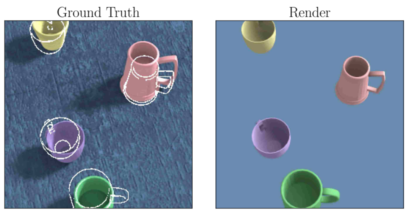


$\xrightarrow{Fit}$

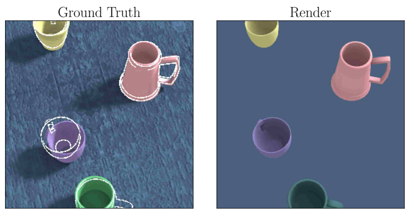


(a) Recognition. IoU=0.53

(b) Fit. IoU=0.75

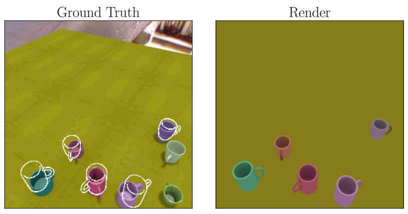


$\xrightarrow{Fit}$

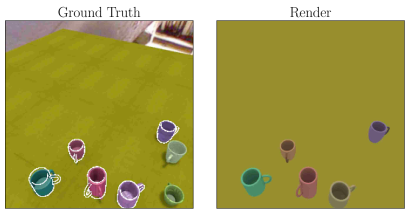


(c) Recognition. IoU=0.65

(d) Fit. IoU=0.81

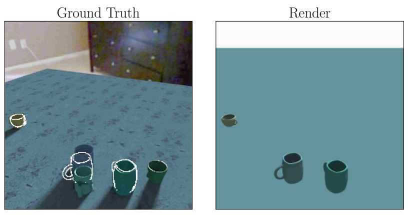


$\xrightarrow{Fit}$

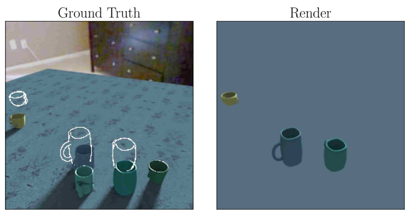


(e) Recognition. IoU=0.54

(f) Fit. IoU=0.88

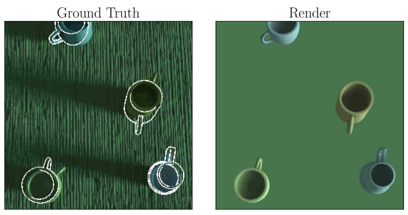


$\xrightarrow{Fit}$

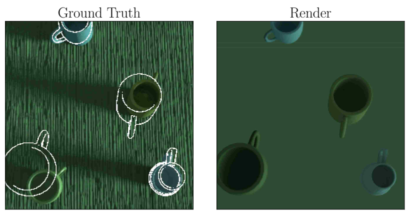


(g) Recognition. IoU=0.89

(h) Fit. IoU=0.05



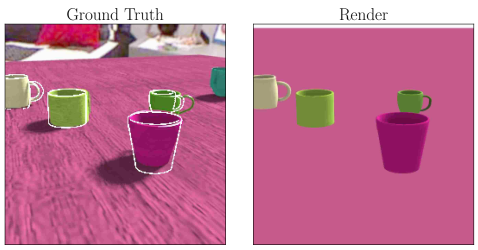
$\xrightarrow{Fit}$



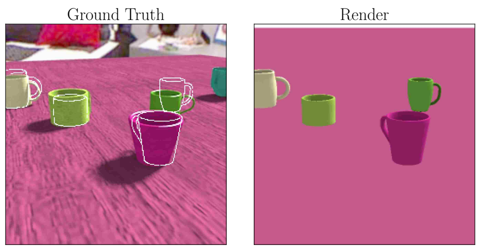
(i) Recognition. IoU=0.73

(j) Fit. IoU=0.57

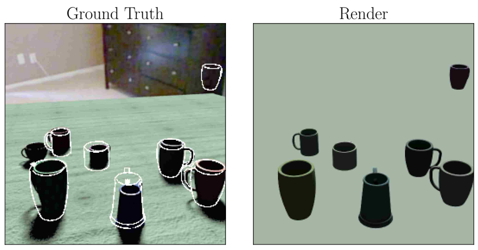
Figure 4.12: Examples of refinement using the robust model with DSAA. For each row we show the GT image next to the rendered scene as given by the recognition model prediction, followed by the corresponding images after refinement. To better visualize the match between GT and predictions, we also overlay the GT images with the white edges of the inferred mugs in the scene.



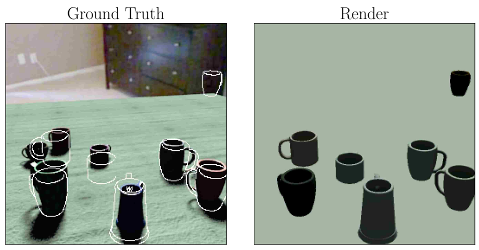
(a) Fit (w. DSAA). IoU=0.82



(b) Fit (no AA). IoU=0.65

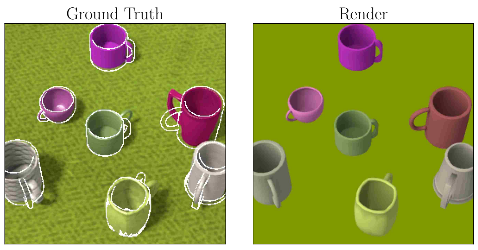


(c) Fit (w. DSAA). IoU=0.71

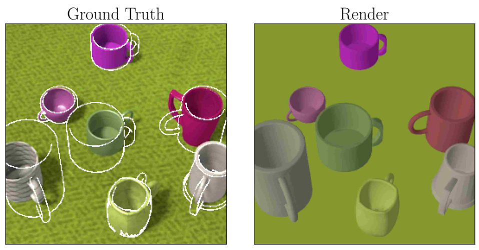


(d) Fit (no AA). IoU=0.52

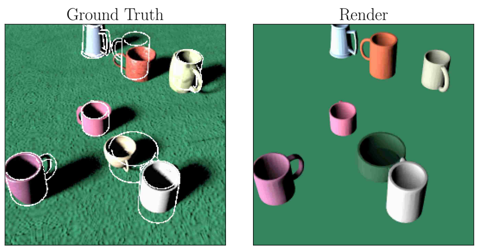
Figure 4.13: Examples comparing the refined scenes when using the DSAA for antialiased rendering (left) vs. using no antialiasing (right).



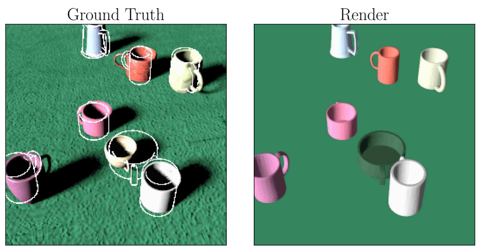
(a) Fit (w. self shadows). IoU=0.86



(b) Fit (no self shadows). IoU=0.69

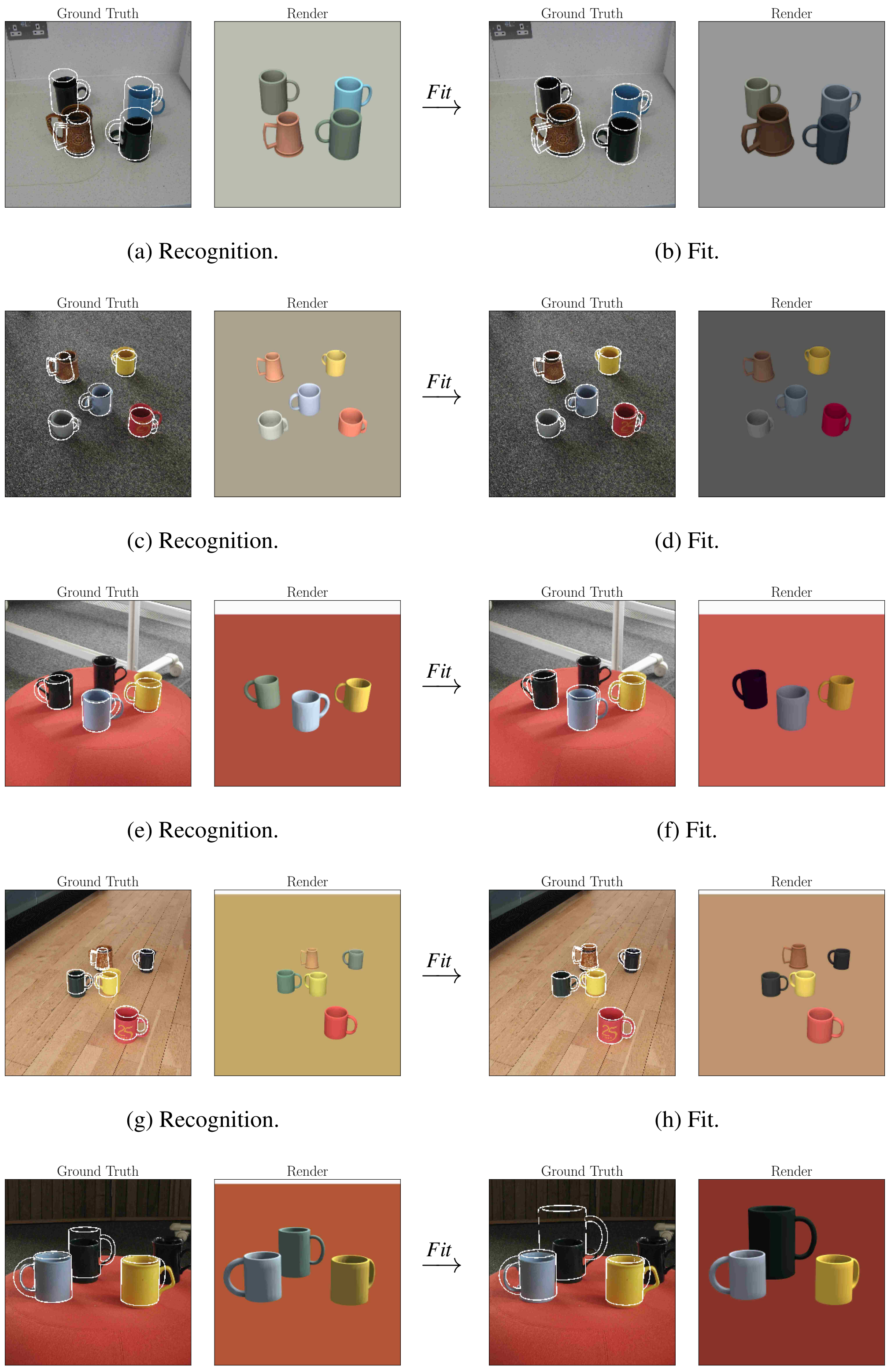


(c) Fit (w. self shadows). IoU=0.81



(d) Fit (no self shadows). IoU=0.59

Figure 4.14: Examples comparing the refined scenes when modelling scenes with self shadows (left) vsṅot using self shadows (right).



(a) Recognition.

(b) Fit.

(c) Recognition.

(d) Fit.

(e) Recognition.

(f) Fit.

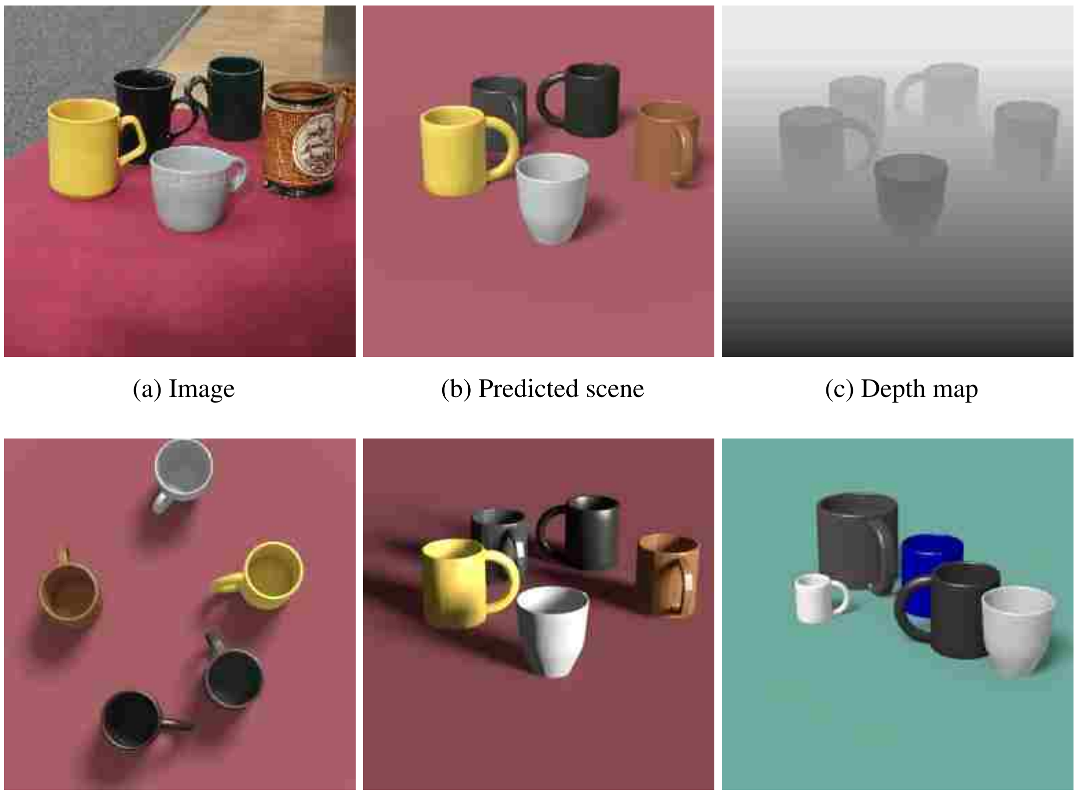
(g) Recognition.

(h) Fit.

(i) Recognition.

(j) Fit.

Figure 4.15: Qualitative examples of refinement using the robust model with DSAA on real images.


(a) Image (b) Predicted scene (c) Depth map

(d) Changing camera (e) Changing illumination (f) Changing objects

Figure 4.16: Applying different transformations to a detailed scene description inferred from an image. In the last image we show how we modify the object latent variables for all mugs as well as the table plane colour.

4.5 Conclusions

In this chapter we have studied the ability of our framework to handle scenes with multiple objects. We have addressed an issue with the generative model in terms of its mismatch with real images. Namely, we proposed ways to pre-compute and efficiently render self shadows. Furthermore, we have proposed improvements in the approximate gradients that result in significantly more accurate derivatives as well as smoother rendering changes as a function of the latent variables. Both of these improvements were shown to be key to having an effective refinement method for most of the latent variables while maintaining the simplicity and flexibility of the OpenDR framework.

Similarly to the self shadow model, future work involves adding rendering features such as specularity and textures that improve the realism of the generative model in an efficient and differentiable way. We are also interested in analysing the limitations of our current refinement method for the camera latent variables as well as object variables such as the object azimuth rotation. In this regard, an interesting direction of future work is in using features of the GT and rendered images that can better capture the discrepancies with respect to these latent variables. Finally, we want to extend these results to a larger variety of object classes and types of scenes (not just indoor).

Chapter 5

Modelling the 3D shape and appearance of objects

5.1 Introduction

In this chapter we propose a deep generative model (the disentangled variational autoencoder, DVAE) that learns a *disentangled* representation in which the 3D shape and appearance of the objects can be interpreted and manipulated independently. The model can be trained from arbitrary 3D mesh datasets without requiring corresponding landmark points across the mesh collection. We demonstrate the ability of the model to generate novel meshes by transferring the disentangled shape and appearance representations between pairs of meshes of a given object class. We evaluate our model on the object classes of cars and faces, and show that it is able to generate high quality samples of these classes.

A key component in inverse graphics solutions is the use of 3D generative models that describe the underlying shape variability and appearance variability of different object classes. In Chapter 3 we used a deformable mesh model with a simple appearance model based on a global RGB colour. The same appearance model was used in Chapter 4 for the different shapes of mugs. Our focus in this chapter is on modelling the 3D shape and appearance of objects using a *disentangled* representation that separates the shape and appearance factors of variation. Such models can be combined with more knowledge-based aspects of vision such as the geometry of object pose and scene lighting to reason jointly about these factors of variation in images. Such models are important, for example, for vision-as-inverse-graphics or analysis-by-synthesis approaches, see e.g. Grenander (1976, 1978); Kulkarni et al. (2015a).

Recent developments in unsupervised machine learning have produced powerful methods such as Variational Autoencoders (VAEs) Kingma and Welling (2014); Rezende et al. (2014) and Generative Adversarial Networks Goodfellow et al. (2014). These could be applied directly to 3D shape and appearance data. However, there is an important *disentangled* aspect of the data that this would miss; for a given shape, it is possible to have different appearances—e.g. consider a car of fixed shape but different body colourings. A disentangled representation allows shape and appearance to be interpreted and manipulated independently. It has also been shown that systems that operate with disentangled representations perform better on novel tasks, as in zero-shot learning Higgins et al. (2016).

Our main contribution in this chapter is the introduction a VAE architecture that is designed to learn a disentangled representation of shape and appearance which can be applied to a large variety of object classes. We also exploit a "hedgehog" representation of shape and appearance (see Sec. 5.2.1) that avoids the need for point correspondences across the objects in the 3D dataset.

The structure of the chapter is as follows: in Sec. 5.2 we describe the hedgehog representation of 3D meshes and the architecture of the disentangled variational autoencoder. Sec. 5.5.2 covers the datasets used, and we evaluate the ability of the model to obtain a disentangled representation. We also show qualitative results of these models based on its samples of 3D objects in Sec. 5.5.4. We discuss the main findings of this work in Sec. 5.6 and provide some interesting directions of future work.

5.2 Methods

We begin in Sec. 5.2.1 with the hedgehog representation, then discuss in sections 5.3.1 and 5.3.2 the disentangled variational autoencoder. In Sec. 5.4 we discuss the related work.

5.2.1 Data representation

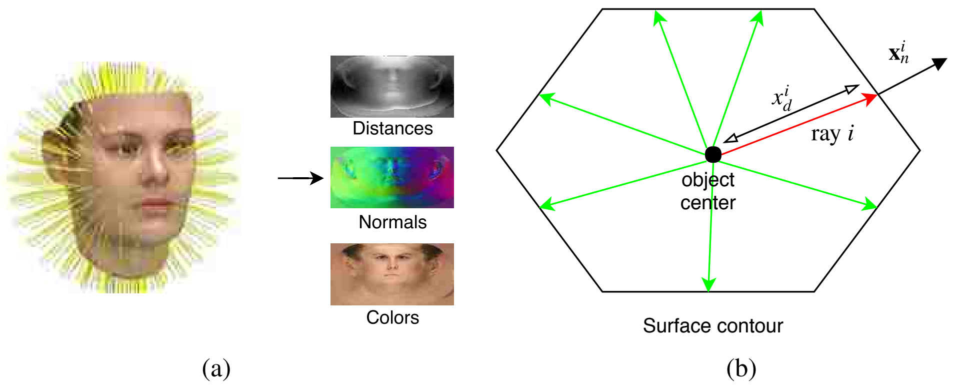


Figure 5.1: (a) The process of taking a mesh (LHS), and turning it into a "hedgehog" representation of distances $\mathbf{x}_d$, surface normals $\mathbf{x}_n$ and colour maps $\mathbf{x}_a$. For visualization purposes the image of normal vectors I_n on the RHS is computed as $I_n = 0.5(\mathbf{x}_n + 1)$. On the right are the equirectangular representations (image) where the X and Y axis correspond to equally distributed azimuth and elevation respectively. (b) An illustration in 2D of how the distance x_d^i and ray normals $\mathbf{x}_n^i$ is computed for the ray i shown in red.

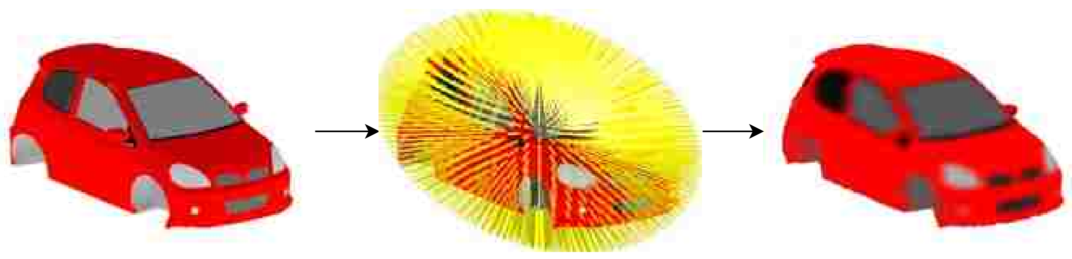


Figure 5.2: Processing meshes into hedgehog representation. On the left we have an input mesh, which is processed using a number of rays sufficient to capture a desired level of detail. On the right we have the hedgehog projection.

The input data for our model consists of meshes having shape and appearance (colour) information. We assume that the meshes are aligned to have the same orientation and origin in x-y-z space, with scaling to similar sizes. Such data can be obtained e.g. from the ShapeNet dataset Chang et al. (2015).

Meshes have an arbitrary number of vertices, and the first step in most shape modelling is to reduce this to a fixed-dimensional representation. Here we use a spherical ray-based representation in which shape is encoded by the distance of R rays which are projected from the centre of the object onto the mesh surface. We create the ray arrangement using a grid of points in azimuth-elevation space.[1] We will refer to it as the *hedgehog* representation. This representation is well suited for convex objects and can efficiently capture the outer surface to a high degree of accuracy. See e.g. (Ballard and Brown, 1982, sec. 9.2.3) for more details of this representation, also referred to as the *direction-magnitude* representation. Fig. 5.1a illustrates the process of converting a mesh into the hedgehog representation using a face from the Basel Face Model (BFM) Paysan et al. (2009). Fig. 5.1b shows how one would compute such representation of a 2D contour surface instead of a 3D mesh. Similarly, Fig. 5.2 shows the result of projecting a car from the ShapeNet dataset Chang et al. (2015) into the hedgehog representation and rendering the resulting mesh.

In addition to the distance of the ray to the intersecting surface, we also encode the unit surface normal vectors, and the RGB colour of the point in the mesh triangle which the ray intersects. As the normal vectors lie in 3 dimensions[2] and the colour is also

[1]Other arrangements are possible, e.g. a triangulated sphere. But the azimuth-elevation choice means it is easy to define spherical convolutions with standard rectangular filters, rather than more exotic arrangements.

[2]We do not explicitly model the unit vector constraint, but this will be learned approximately by the

3D (RGB), the data vector for each ray has 7 dimensions. Modelling the normal vectors serves two purposes, first they improve the quality of reconstruction of a surface mesh from a set of points using methods such as Poisson reconstruction (Kazhdan et al., 2006). Second, both vertex positions and normals, as well as vertex colours and normals are tightly correlated and thus act as a bridge when modelling the shape and appearance of the objects. For instance, the transition between a car's windscreen and roof has sharp changes in both the colour and normal vectors.

In this chapter we depart from table-top type of objects and focus on two well known object classes in computer vision: *faces* and *cars*. Both are classes of high importance and also happen to be a suitable type for the hedgehog representation.

The hedgehog representation is not the only possible choice—one common choice is the use of *landmark* points, as used e.g. in 2D with Active Appearance Models (AAMs) Cootes et al. (2001), and in 3D with 3D Deformable Models that have been widely used for many tasks such as face modelling (Blanz and Vetter, 1999), and to model cars (Leotta and Mundy, 2011). But a major challenge for this line of work is finding landmark correspondences across objects; this is an active research topic and solutions can be quite involved, see e.g. the work of Huang et al. (2015b). A discussion of how the AMM relates to the hedgehog representation is in sec. 5.2.1.1 of the supp. material. Another alternative is to use a *volumetric* representation, by discretizing the space of the mesh vertices using voxels. This type of representation has gained interest from the research community as it is also well suited to deep learning methods that make use of 3D convolutions, e.g. Wu et al. (2015). However, voxel based representations are high dimensional and inefficient due to most voxels not intersecting the object. For example Wu et al. (2015) use only a $30 \times 30 \times 30$ grid, which leads to a very coarse shape representation.

5.2.1.1 Analysis of the hedgehog representation

A hedgehog object is represented by R ray vectors. For each ray i we have the ray distance x_d^i and the 3D surface normal $\mathbf{x}_n^i$. By the shape variables $\mathbf{x}_s^i$ we will refer to the concatenation of both: $\mathbf{x}_s^i = (x_d^i, (\mathbf{x}_n^i)^T)^T$. The appearance variable of the ray $\mathbf{x}_a^i$ is the 3D colour of the mesh at the point of intersection.

Let us first analyze the relationship between $\mathbf{x}_s^i$ and the vertices $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$ that define

model.

the triangle intersected by the ray i. When the vertices move, $\mathbf{x}_s^i$ will change (a) based on the position of $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$, (b) the ray starts intersecting a different triangle. Here we analyze the case (a). The distance of the ray from its origin $\mathbf{x}_O$ to the triangle can be calculated via the well known line-plane intersection formula for the plane in which the triangle is embedded:

$$x_d^i = -\frac{\mathbf{n}_v^T \mathbf{x}_O + D}{\mathbf{n}_v^T \mathbf{r}^i}, \tag{5.1}$$

where $\mathbf{n}_v$ is the plane normal vector defined by $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$, and D is the constant of the plane (defined by $Ax+By+Cz+D=0$) where $(A,B,C) = \mathbf{n}_v^T$, and $\mathbf{r}^i$ is the ray direction in Cartesian coordinates. In fact it is possible to solve for x_d^i without explicitly solving for the normal – one can use e.g. the Möller-Trumbore (MT) algorithm (Möller and Trumbore, 2005). To see the direct relationship with the vertices we can substitute $\mathbf{n}_v^T$ using $\mathbf{n}_v^T = (\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_2)$. Note how $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$ appear in the top and bottom of the equation for x_d^i (notice that D also depends on $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$), so this is in fact a nonlinear computation.

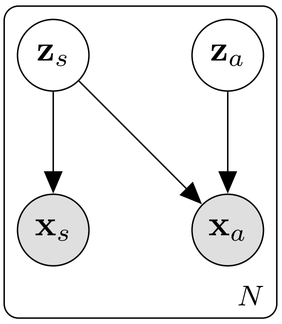
If we model the vertex locations using a Gaussian distribution, it is clear from 5.1 that this will have a nonlinear (non-Gaussian) effect on x_d^i. In other words, even for the case of the Basel Face Model, which uses a Probabilistic Principal Component Analysis (PPCA) model of vertex positions, the resulting hedgehog distance variables will have a non-Gaussian distribution. This motivates the non-linear DVAE introduced in the paper.

As well as asking about the distance along the ray, we can also ask about the appearance $\mathbf{x}_a^i$. We assume that each vertex has a colour, and we obtain the colour of a point in the triangle by linear interpolation in terms of barycentric coordinates. We can see that the normalizing factor T (area of the triangle) in the computation of the barycentric coordinates also introduces a non-linear function with respect to $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$. This is separate to the variation we can get by having a distribution on the colours of the vertices.

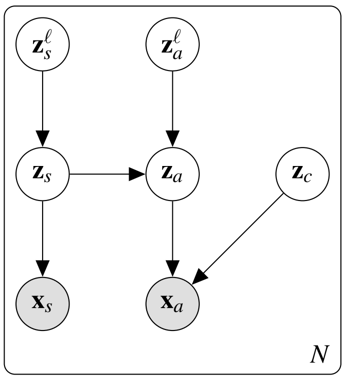
As for the variation of $\mathbf{x}_a^i$ with respect to a distribution over the mesh colours, we can see that appearance variability will have a linear effect on the ray colour as the barycentric coordinates linearly combine to form define the colour at the intersected point.

5.3 Methods

5.3.1 Shape and Appearance Disentangled Generative Model



(a) Faces graphical model



(b) Cars graphical model

Figure 5.3: (a) The graphical model of the disentangled autoencoder used to model faces. The latent representation is split up into shape variables $\mathbf{z}_s$ and appearance variables $\mathbf{z}_a$. (b) The graphical model for the disentangled VAE used to model cars. This model consists of a two-level hierarchy of latent variables: first level $\mathbf{z}_s^\ell$, $\mathbf{z}_a^\ell$ models the high level (coarse) structure of the car, and and the intermediate level $\mathbf{z}_s$, $\mathbf{z}_a$ models the finer detail. The variables $\mathbf{z}_c$ model a latent distribution of the object colours that the object can take. The regions of the car in which these colours are "painted" are modelled by the appearance style variables $\mathbf{z}_a$.

We first consider the directed acyclic graphical model (DAG) shown in Fig. 5.3a, where $\mathbf{z}_s$ and $\mathbf{z}_a$ are the latent variables (LVs). Note the connection from $\mathbf{z}_s$ to $\mathbf{x}_a$: this is due to the fact that if the appearance LVs of the generative model were fixed but the latent variables of shape changed, then the appearance $\mathbf{x}_a$ would need to change correspondingly. See e.g. (Prince, 2012, Sec. 17.7) for a discussion of this point with respect to landmark-based models and warping. The fact that there is no connection between $\mathbf{z}_a$ and $\mathbf{x}_s$ forces the $\mathbf{z}_s$ units to model shape. In other words, the independence of $\mathbf{x}_s$ from $\mathbf{z}_a$ is imposed architecturally by ensuring that pathways for $\mathbf{z}_s$ do not connect with $\mathbf{x}_a$ variables, while the $\mathbf{z}_a$ variables can see both $\mathbf{x}_s$ and $\mathbf{x}_a$. This graphical model resembles the probabilistic version of the well known Canonical Correlation Analysis (CCA)Bach and Jordan (2005), where $\mathbf{z}_s$ would correspond to the variable that captures the joint correlation between both observed variables; however in our model $\mathbf{x}_a$ has extra latent variables that do not connect with $\mathbf{z}_s$, whereas in CCA the latent variables are connected to both vectors of observed variables.

The simplest version of the model shown in Fig. 5.3(a) is to use linear mappings between the latent and visible variables. If we allowed full connectivity between observed and latent variables, such a network would be equivalent to doing PCA when training with a squared-error reconstruction loss, as per Baldi and Hornik (1989).

In the previous section we established the existence of nonlinear effects between the mesh and the hedgehog representation. Despite such nonlinearity, we are still interested in investigating to what extent a linear model can capture the shape and appearance structure of the objects. As we will see in the experiments of Sec. 5.5.2, a linear architecture turns out to be a good model for the face class in the hedgehog representation. This is perhaps due to the underlying linear (PCA) structure of the BFM meshes.

5.3.2 The Disentangled Variational Autoencoder

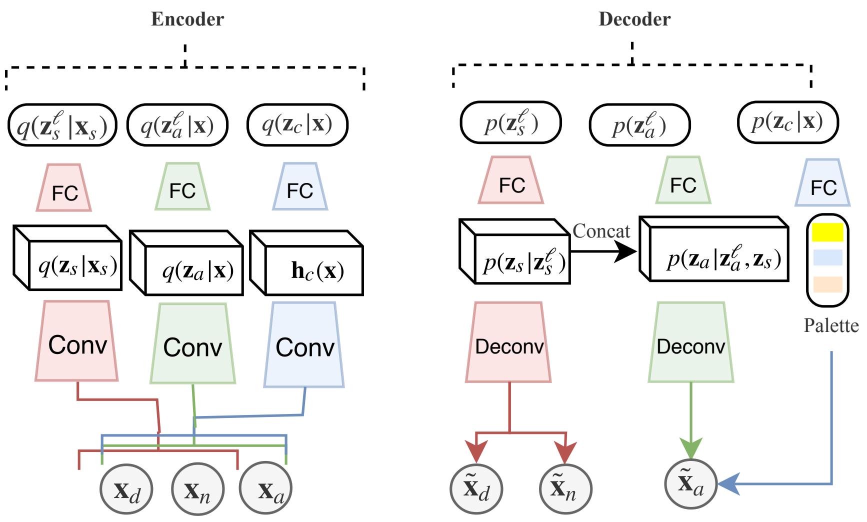

Encoder
Decoder
$q(\mathbf{z}_s^\ell|\mathbf{x}_s)$
$q(\mathbf{z}_a^\ell|\mathbf{x})$
$q(\mathbf{z}_c|\mathbf{x})$
FC
FC
FC
$q(\mathbf{z}_s|\mathbf{x}_s)$
$q(\mathbf{z}_a|\mathbf{x})$
$\mathbf{h}_c(\mathbf{x})$
Conv
Conv
Conv
$\mathbf{x}_d$
$\mathbf{x}_n$
$\mathbf{x}_a$
$p(\mathbf{z}_s^\ell)$
$p(\mathbf{z}_a^\ell)$
$p(\mathbf{z}_c|\mathbf{x})$
FC
FC
FC
Concat
$p(\mathbf{z}_s|\mathbf{z}_s^\ell)$
$p(\mathbf{z}_a|\mathbf{z}_a^\ell,\mathbf{z}_s)$
Palette
Deconv
Deconv
$\tilde{\mathbf{x}}_d$
$\tilde{\mathbf{x}}_n$
$\tilde{\mathbf{x}}_a$

Figure 5.4: Autoencoder architecture of the DVAE for cars. The encoder on the LHS outputs the approximate distributions for the latents. The decoder on the RHS models the probabilistic generative model. Note that $\mathbf{x}_s = (x_d, \mathbf{x}_n)$

The linear model described above may not be suitable for some object classes. A linear combination of shape and appearance latent variables, for instance, cannot properly handle the complex interaction between the shape and the colour of the different parts of cars (i.e. the sharp appearance transitions between colours of different parts of the object such as windows, headlights, etc.). The samples in Fig. 5.9 of the linear model described above trained on the cars dataset clearly show that it cannot properly capture the car structure. We have experimented with a larger number of latent dimensionality and, while reconstructions do look increasingly better, the samples remain quite implausible (especially with regards to the generated appearances of the cars).

We propose a number of changes to the generative model (GM) defined in Fig. 5.3(a), and define the model to be a hierarchical latent variable model as shown in Fig. 5.3b. The network architecture we propose that implements this GM structure is shown in Fig. 5.4. While Fig. 5.3b is more complicated than Fig. 5.3(a), it still follows the same principle of disentangling shape and appearance in its latent representation. We first describe the generative model architecture below, which we refer to as the *Disentangled Variational Autoencoder* (DVAE), and then give the training details in Sec. 5.3.4.

5.3.3 Model architecture

At the highest level of the latent structure we have $\mathbf{z}_s^\ell$, $\mathbf{z}_a^\ell$ which model the global structure of shape and appearance. We set the LVs to be normally distribution (zero mean and diagonal unit variance). Next we have the conditional distribution of the intermediate variables $\mathbf{z}_s$, $\mathbf{z}_a$ that depend on the high level LVs. We can think of such high-level to intermediate-level structure as a coarse-to-fine sampling process. Note that it is in the conditional probability $p(\mathbf{z}_a|\mathbf{z}_s, \mathbf{z}_a^\ell)$ where the shape and appearance latent representations are combined. Conceptually, $\mathbf{z}_s$ contains all the shape information (distances and normals), while $\mathbf{z}_a$ models the latent distribution of appearance.

We propose modelling the intermediate LVs using a PixelCNN (van den Oord et al., 2016) distribution. PixelCNN is a nonlinear auto-regressive probabilistic model in the form of a CNN that has been proven to be a powerful image model that can produce realistic samples, making it a suitable model for our intermediate (fine-grained) representation. A key feature of PixelCNN is the use of convolutional layers that can take advantage of the local structure of the hedgehog rays. Gulrajani et al. (2017) also propose a generative model with a PixelCNN as the distribution model of its latent space in order to model natural images.

The auto-regressive probability distributions for $\mathbf{z}_s$ and $\mathbf{z}_a$ take the form:

$$p(\mathbf{z}_s|\mathbf{z}_s^\ell) = \prod_{i=1}^{R} p\left(\mathbf{z}_s^i|\mathbf{z}_s^1, ..., \mathbf{z}_s^{i-1}, FC(\mathbf{z}_s^\ell)\right), \tag{5.2}$$

$$p(\mathbf{z}_a|\mathbf{z}_s, \mathbf{z}_a^\ell) = \prod_{i=1}^{R} p\left(\mathbf{z}_s^i|\mathbf{z}_a^1, \mathbf{z}_s^1 \ldots, \mathbf{z}_a^{i-1}, \mathbf{z}_s^{i-1}, FC(\mathbf{z}_a^\ell)\right), \tag{5.3}$$

where the ordering $1, ..., i-1$ of rays is top to bottom and left to right starting from the ray at the top and left of the hedgehog representation. This is the order followed in most implementations of PixelCNN, due to the way that convolutions in CNNs are implemented, without loss of generality since PixelCNN models the joint distribution of the pixels as a product of conditionals. These conditional probabilities depend on the high level LVs via $FC(\mathbf{z}_s^\ell)$ and $FC(\mathbf{z}_a^\ell)$, where FC denotes fully connected linear mappings from the LVs to feature maps which are fed to the PixelCNN. Each conditional probability is a multi-modal distribution modelled with a mixture of Gaussian distributions, with both the means and variances predicted by the CNN.

In the shape decoder, the LVs $\mathbf{z}_s$ are mapped to the output distribution $p(\mathbf{x}_s|\mathbf{z}_s)$ via deconvolutional layers which up-sample the latent representation to the spatial dimensionality of $\mathbf{x}_s$. The outputs of the deconvolutional layers are the mean and variance of a diagonal Gaussian distribution per ray i.

In order to describe how the appearance variables $\mathbf{x}_a$ are generated, we first introduce the concept of a palette of colours, and the associated LVs of colour $\mathbf{z}_c$. Cars, as for many other classes, are usually composed of only a few different colours. We model this by explicitly setting the colour of one ray as a convex combination of a palette of K colours (and each object has an associated palette). A similar idea, known as probabilistic index maps, is applied to models of images (Jojic et al., 2004).

Our palette of colours can be seen as a 3D version of a probabilistic index map, and serves as an inductive bias to encourage the model to further disentangle the latent representation: $\mathbf{z}_s^\ell$ and $\mathbf{z}_s$ model the shape (e.g. large pick-up car), $\mathbf{z}_a^\ell$ and $\mathbf{z}_a$ model the appearance style (e.g. the type or layout of the windows of the pick-up, the shape of the headlights, whether it has stripes, etc.) and $\mathbf{z}_c$ models the colouring of those parts of the car (e.g. red body, dark windows, blue stripes, etc.). The latent variable $\mathbf{z}_c$ is a D_c dimensional vector that can be modelled by a Gaussian distribution. Although different possible mappings can be used, the mapping from $\mathbf{z}_c$ to the palette colours we experimented is further explained in Sec. 5.3.4.

The model for the appearance variables $p(\mathbf{x}_a|\mathbf{z}_a,\mathbf{z}_c)$ needs to take into account that colours and textures obtained from mesh datasets are often discretized (each RGB channel usually take one value between $[1,256]$). Hence we model the appearance variables $\mathbf{x}_a$ by a discretized logistic distribution as used e.g. by Salimans et al. (2017) (although they use a mixture distribution while we use a uni-modal one).

Overall, $\mathbf{z}_s^\ell$ and $\mathbf{z}_s$ model the shape (e.g. large pick-up car), $\mathbf{z}_a^\ell$ and $\mathbf{z}_a$ model the appearance style (e.g. the type or layout of the windows of the pick-up, the shape of the headlights, whether it has stripes, etc.) and $\mathbf{z}_c$ models the colouring of those parts of the car (e.g. red body, dark windows, blue stripes, etc.).

We train the model as a *variational autoencoder* (VAE) Kingma and Welling (2014); Rezende et al. (2014). In the general form, given observed variables $\mathbf{x}$, there is an approximate posterior distribution $q(\mathbf{z}|\mathbf{x},\phi)$ which depends on the encoder parameters ϕ. This is usually modelled as a Gaussian with a diagonal covariance matrix. In our encoder both the Gaussian mean and variances of all the latent variables are a function of the input. The architecture of the encoder-decoder disentangled VAE (DVAE) is shown in Fig. 5.4.

The encoder is defined by three different bottom-up streams from visible to the latents as shown in the LHS of Fig. 5.4. Each stream is composed of the following elements: (1) *Conv* denotes a sequence of convolutional layers that map the input to a hidden feature map representation. In the case of $\mathbf{z}_s$, $\mathbf{z}_a$ these feature maps represent the parameters of the Gaussian distributions $q(\mathbf{z}_s|\mathbf{x}_s)$ and $q(\mathbf{z}_a|\mathbf{x}_a)$. (2) The last hidden layer feature map is linearly projected to produce the parameters for the Gaussian distributions of $q(\mathbf{z}_s^\ell|\mathbf{x}_s)$, $q(\mathbf{z}_a^\ell|\mathbf{x})$ and $q(\mathbf{z}_c|\mathbf{x})$.

Note how we impose a constraint in the encoder so that $\mathbf{z}_s$ does not have access to $\mathbf{x}_a$ (red stream of the encoder in Fig. 5.4). The purpose of this constraint is that if the shape latent variables are not informed by $\mathbf{x}_a$ when encoding, the decoder is encouraged to learn to properly use $\mathbf{z}_s$ when generating $\mathbf{x}_a$, instead of simply compressing appearance information in $\mathbf{z}_a$. A similar idea is shown to be useful in unsupervised learning by cross-channel predictions (Zhang et al., 2017) as it forces the network to learn the the underlying structure instead of simply compressing the visible signals in its latent representation. The appearance stream and colour stream are shown in green and blue, and have access to all the latent variables.

The encoder and decoder networks are trained to maximize the evidence lower bound (ELBO) on the true log-likelihood:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\theta,\phi) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right), \text{ where} \\ q_\phi(\mathbf{z}|\mathbf{x}) &= q_\phi(\mathbf{z}_s|\mathbf{x}_s)q_\phi(\mathbf{z}_s^\ell|\mathbf{x}_s)q_\phi(\mathbf{z}_a|\mathbf{x})q_\phi(\mathbf{z}_a^\ell|\mathbf{x})q_\phi(\mathbf{z}_c|\mathbf{x}), \\ p_\theta(\mathbf{x}|\mathbf{z}) &= p_\theta(\mathbf{x}_s|\mathbf{z}_s)p_\theta(\mathbf{x}_a|\mathbf{z}_a,\mathbf{z}_c), \\ p_\theta(\mathbf{z}) &= p_\theta(\mathbf{z}_s^\ell)p_\theta(\mathbf{z}_a^\ell)p_\theta(\mathbf{z}_c)p_\theta(\mathbf{z}_s|\mathbf{z}_s^\ell)p_\theta(\mathbf{z}_a|\mathbf{z}_s^\ell,\mathbf{z}_a^\ell),\end{aligned} \tag{5.4}$$

and θ denotes the parameters of the decoder network. In the general case the lower bound is intractable, so it is approximated using a Monte-Carlo estimator, and optimized using stochastic gradient ascent (Kingma and Welling, 2014; Rezende et al., 2014).

5.3.4 Architecture and training details

Above we described the high level view of the DVAE architecture. A number of different hyperparameters were tried on the training and validation sets and the following configuration worked well for cars:

- We set $\mathbf{z}_s^\ell$, and $\mathbf{z}_a^\ell$ as two 100-D normally distributed vectors, $\mathbf{z}_c$ to be a 30-D vector. The intermediate latent variables and $\mathbf{z}_s$ and $\mathbf{z}_a$ are set to be a tensor of 16×16 with 16 channels each.
- To implement Eq. 5.2 we use the PixelCNN++ implementation (Salimans et al., 2017) in which we apply a residual convolutional layer, followed by one downscale-upscale convolutional layer as a way to augment the effective receptive field of the model. We refer to the work of (Salimans et al., 2017) for further details. Each conditional of Eq. 5.2 is parametrized by a mixture of 10 Gaussians (where the means, variances and mixture coefficients are all predicted by the PixelCNN).
- The *Conv* component of the encoder illustrated in Fig. 5.4 is composed of two $4 \times 4 \times 64$ convolutional layers and stride $[2,2]$ filters (which halves the spatial resolution twice), followed by two residual convolutional layers which outputs the means and variances for the intermediate latent variables. A residual layer is defined as $h_{l+1} = ReLu(Convolution(h_l)) + h_l$, if rectified linear units (ReLu) for the nonlinear activation functions are used.
- Conversely, the *Deconv* component of the decoder is composed of two residual

convolutional layers (with $4 \times 4 \times 128$ filters) and two deconvolutional layers. We define a deconvolutional layer to be a nearest-neighbour 2×2 up-scaling of the input feature maps followed by a convolutional layer (with $4 \times 4 \times 64$ filters). The output of this component is $[\mathbf{h}_a, \mathbf{s}_c] = \mathit{Deconv}(\mathbf{z}_a)$, where $\mathbf{s}_c$ is the log-scale variable used to compute the variance of the discretized logistic model, and $\mathbf{h}_a$ is used to compute the corresponding mean parameters μ_a as explained below.

- **Palette of colours.** As explained above, we choose to model the mapping between $\mathbf{z}_c$ to $\mathbf{x}_a$ using a key-value type of decomposition. While we could have instead predicted the colour of each ray directly, our decomposition allows the DVAE to encode the semantic meaning of each ray in the keys (e.g. same ray keys naturally segment same parts of the object) from their values (colours). The colour at index m of the palette of a car is given by:

$$\mathbf{c}_m = \sigma(W_c^m \mathbf{z}_c + \mathbf{b}_c^m), \tag{5.5}$$

where $\sigma(x)$ is the sigmoid function which constrains the colours to be between 0 and 1. W_c is a $3 \times D_c$ weight matrix and $\mathbf{b_c}$ is the bias vector. Each colour m of the K colours in the palette has an associated global "colour key" $\mathbf{g}_m$ which is used for the generative model to "pick" the colour for each ray. This key-value mechanism is often used as a way to define a differentiable dictionary (see an example of its use in the work of Pritzel et al. (2017)). The mean appearance for a ray i is computed by combining the ray predicted colour keys $\mathbf{h}_a^i$, which is the output of the appearance stream in the decoder, i.e. $\mathbf{h}_a^i = \mathrm{Deconv}(\mathbf{z}_a)$, and the $K \times 3$ palette C by a weighted combination of its colours:

$$\mu_a^i = \sum_{m=1}^{K} w_m^i \mathbf{c}_m, \tag{5.6}$$

where w_m is the weight of colour m computed using the normalized similarity between the predicted keys $\mathbf{h}_a^i$ and the palette keys:

$$w_m^i = \frac{k(\mathbf{g}_m, \mathbf{h}_a^i)}{\sum_{j=1}^{K} k(\mathbf{g}_j, \mathbf{h}_a^i)}. \tag{5.7}$$

There are different types of similarity measures we can use, but a squared exponential kernel worked well in our case:

$$k(\mathbf{g}_m, \mathbf{h}_a^i) = \exp\left(-\sum_{j=1}^{3}(g_m^j - h_a^{j,i})^2\right). \tag{5.8}$$

We found that $K = 10$ colours to be a sensible number for cars, as well dimension 3 for the key vectors. Note that the key vectors of the palette are global for all cars. The keys of the palette are randomly initialized with zero-mean Gaussian and $\sigma^2 = 0.01$, and learned during training process.

- No dropout or batch norm are used, and all the non-linear activation funtions are rectified linear units (ReLu). The Adam optimizer (Kingma and Ba, 2015) is used with a learning rate of 0.0002.

Auxiliary losses. We include in the cost function two auxiliary losses (t_e and t_a) which relate specifically to the palette. First we want each of the colours in the palette to represent a final colour, but if a ray's blending weights of Eq. 5.7 are not concentrated into a single colour, these will be blended to produce the colour of a ray of the mesh. We can encourage the weights to become more peaked by adding the auxiliary term t_e to the ELBO:

$$t_e = \sum_{i=1}^{R}\sum_{m=1}^{K} w_m^i \log(w_m^i) \tag{5.9}$$

where again R is the number of rays. The weight of this term is annealed from 0 to 10 in the first 100 epochs. Barron and Malik (2015) use a similar prior term for their GM referred to as "parsimony". The authors use such term to encourage their model to globally use a small number of colours for the whole image, while we use it to encourage each ray to ideally only take from one colour in the palette.

The second auxiliary term is added to enforce consistency in the use of the palette across objects. Since we use powerful nonlinear encoders and decoders in our DVAE, the training network might not converge to a palette usage that is consistent across cars (e.g. the first colour of the palette might model the upper body of one car, and the lower body of another). To encourage such consistency, we add to the ELBO an extra reconstruction term for $\mathbf{x}_a^n$ for each car n in a mini-batch: $t_a^n = \log p(\mathbf{x}_a^n|\tilde{\mathbf{x}}_a^n)$, where $\tilde{\mathbf{x}}_a^n$ is the output of the decoder under latent variables $\mathbf{z}_s^n$, $\mathbf{z}_a^m$ and $\mathbf{z}_c^n$. By using an appearance style representation from a different car in the training set, $\mathbf{z}_a^m$, we encourage the decoder to make use of $\mathbf{z}_s^n$ independently from its appearance style when generating $\tilde{\mathbf{x}}_a^n$ (i.e. similarly shaped cars should produce similar output appearances)[3]. The weight of

[3]While this is one way to initially make the network learn to generate $\mathbf{x}_a$ using $\mathbf{z}_s$ and $\mathbf{z}_c$ alone, there

this term is annealed from 0.5 to 0.01 in 100 epochs. This is because if the weight of this term was not annealed to 0 or a very small number, the decoder would never learn to use the appearance style variables $\mathbf{z}_a^\ell$ and $\mathbf{z}_a$.

Object class symmetry. In addition to the above configuration, we take into account the hedgehog representation as a spherical structure in our architecture as follows. First, we include an additional input to the encoder which is an $R \times 3$ dimensional "image" composed of the angle information of each ray (i.e. sin and cos of the azimuth angle as well as the elevation angle). Having this extra input allows lower level filters in the convolutional layers to more quickly specialize its features with respect to different areas of the cars. Note that we do not model these extra variables (they are assumed fixed), but rather condition the model on them.

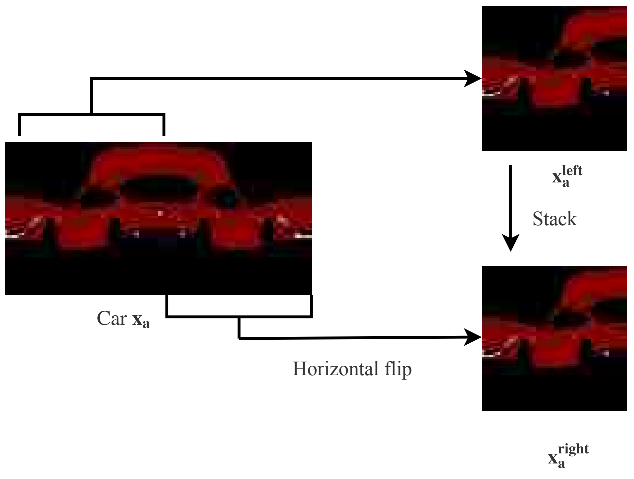


Figure 5.5: Handling the symmetry property of object classes such as cars. The right side image of $\mathbf{x}_a$ is first flipped horizontally to match the left side. Note that the same process is applied to x_d and $\mathbf{x}_n$. This allows the network convolutional filters to capture the symmetrical structure while still having the flexibility to model different shapes and appearance structures in its left and right sides.

Furthermore, we note that cars (and many other categories such as faces) have a symmetry property which we take advantage of. We exploit this property by a careful handling of $\mathbf{x}$ as shown in Fig. 5.5. Note that we concatenate the symmetric sides for both $\mathbf{x}_s$ and $\mathbf{x}_a$. Furthermore, we use *reflective* padding in the convolutional layers so that filters have access to the contiguous ray near the edges of hedgehog images. Without

can be other ways of achieving the same effect. We could for example use the average of the encoded appearances $\mathbf{z}_a$ instead of using a random appearance of another car $\mathbf{z}_a^m$ when computing this auxiliary loss term these two modifications the generated samples sometimes produced non-symmetrical samples with visual artifacts along the boundaries.

5.4 Related work

We have covered much of the related work above, e.g. on active appearance models (AAMs) and their 3D generalizations and volumetric models.

While AAMs have been shown to be very useful models, often it is difficult to obtain the landmark data necessary to train these for an arbitrary class. In this work we are interested in models that learn these factored representations from data *directly*. The importance of *disentangling* factors of variation was emphasized by Bengio et al. (2013).

An interesting approach to learning factored representations is by means of bi-linear (or multi-linear) generative models (Tenenbaum and Freeman, 2000; Grimes and Rao, 2005; Tang et al., 2013). The key idea is that by allowing multiplicative interactions of the latent representations, these are able to capture notions of style (e.g. font of a digit) and content (e.g. class of a digit) separately. While these works focus on models of different image domains (e.g. digits, letters, faces, etc.), we are interested in models of 3D meshes.

Perhaps a closer work to ours with regards to learning representations from data is that of Kulkarni et al. (2015b). The authors addressed learning disentangled representations, e.g. of facial shape/texture, pose and lighting. However, because they were tackling the hard problem of using images as input, it was necessary to signal to the neural network which factor was being varied, and which were being held fixed. Also they used a neural network as a differentiable graphics renderer, which meant that the effect of e.g. pose variation has to be learned specifically for an object class, rather than being generic geometric knowledge.

To our knowledge, the most similar work to ours is the recent work of Wang and Gupta (2016) on Style and Structure Generative Adversarial Networks. They separate the generation of structure (a surface normal map) from style (texture and illumination) in a two-stage process similar to Fig. 5.3a. However, note that their structure map for our task would need to model both object shape and pose variability, and the style map would need to model both object appearance and lighting variation. Also the planar surface normal map representation they use is not useful for modelling the full viewing sphere of shape and appearance variation. However, their results are impressive when dealing with the wide range of variation that arises from scenes containing multiple objects.

5.5 Evaluation

Having defined our models for faces and cars, we are interested in evaluating their learned representations and the quality of their samples. In Sec. 5.5.1 we describe the cars and faces data used in our experiments. In Sec. 5.5.2 we describe how to evaluate the ability of the GMs to transfer the appearances between two objects, and Sec. 5.5.3 gives the results of these transfer experiments. In Sec. 5.5.4 we give a qualitative evaluation of the samples from the disentangled GMs.

5.5.1 Data

We focus on two classes that are of significant importance to the computer vision research community: cars and faces. These classes also have the desirable property of generally having a convex shape thus being well suited for the hedgehog representation.

5.5.1.1 Cars dataset

We use ShapeNet (Chang et al., 2015, v1) as the source of 3D *car* meshes. To date the ShapeNetCore subset provides over 51,300 unique, clean, categorized and aligned models for 55 common object categories. We extract 6,648 meshes from all different sub-categories and isolate the main body of the car by removing the wheels using the part-segmentations provided by the recent work of Yi et al. (2016). The availability of such large-scale part-segmentation extends the range of possible classes to which we can apply our methods, i.e. non-convex objects can be decomposed into convex-like parts (e.g. the body of a car and the wheels are jointly more non-convex than if modelled separately). While the car meshes are already provided normalized, after isolating the main body of the car we place the center of the bounding box of the object at the origin. We found that a ray grid of 128 azimuth and 64 elevation angles (8,192 in total) was a sufficiently accurate representation of the original meshes.

5.5.1.2 Faces dataset

For the face class we obtain the 3D shapes and appearances from the Basel Face Model (BFM) Paysan et al. (2009). This model produces high-resolution meshes and textures using 199 principal components trained from real scans of faces. While this model allows us to generate as many faces as we want, we use 10,000 randomly generated faces out of which 90% are used as training and validation sets – we keep the number of faces to a similar magnitude as the number of objects of most classes in ShapeNet. Since the meshes generated by the BFM are all aligned we simply apply a global scaling by normalizing them to max unit distance.

When projecting the spherical rays into the mesh, we must handle the rays that do not intersect with the mesh. Non-intersecting rays are a rare occurrence in cars as they usually have a mesh surface defined over all possible directions from the origin. However, when a ray does not intersect we instead project the ray in the opposite direction and encode it as a negative distance. If the ray does not intersect on either direction we set the distance to zero. For the BFM faces however, a large portion of the sphere is not covered by the face mesh. Thus, we crop the azimuth-elevation representations to the smallest equirectangular subset that includes the rays that did intersect (note that such subset may include non-intersecting rays). For a non-intersecting ray within that subset, we set its distance, normal vector and colour to be the average ray values that did intersect across the rest of the meshes (i.e. mean imputation).

Since faces have important smaller details than cars, we increased the spherical ray resolution to a ray grid of 256×128 azimuth-elevation angles. The subset of rays after cropping, as explained above, are $172 \times 94 = 16{,}168$ dimensions.

5.5.2 Evaluating shape and appearance transfer

One of the most exciting aspects of our model is that it allows us to independently transfer the shape and appearance between two meshes. A model that can successfully disentangle appearance from shape can thus be used to swap appearances (or shapes) between different objects. For example in computer graphics this would allow us to enhance the number of models available by creating new ones.

The evaluation metric we use for the experiments below is the mean squared error (MSE) between a ground truth and predicted hedgehog representations. For instance, the appearance error for objects A and B is given by:

$$E_a(A,B) = \frac{1}{R}\sum_{i=1}^{R}\sum_{c=1}^{3}(x_a^{A,i,c} - x_a^{B,i,c})^2, \tag{5.10}$$

where $x_a^{A,i,c}$ denotes the ith ray and cth colour of object A, R denotes the number of rays per object, the number 3 refers to the RGB channels of colour. The shape error is defined equivalently but applied to the 4 elements of shape instead. The metrics used for the transfer experiment are described below.

For the **face experiment** we quantitatively evaluate our models with an appearance transfer experiment as follows. Given two faces A, and B from the BFM, we generate the ground truth combination face C (C for *Combined*) by using the latent BFM representations of the 3D Deformable Model for the *shape of A and the appearance of B* to obtain $(\mathbf{x}_s^C, \mathbf{x}_a^C)$.

A naïve way of transferring the appearance of B onto the shape of A is simply to combine $(\mathbf{x}_s^A, \mathbf{x}_a^B)$. This suffers from the issue that $\mathbf{x}_a^B$ has not taken into account the fact that the shape of A is different from the shape of B, and thus that the appearance should have adjusted to take this into account. Thus we define the naïve transfer error $E_a(\mathbf{x}_a^C, \mathbf{x}_a^B)$ as MSE Transfer X. In contrast, rather than naïve transfer using the visible representations $\mathbf{x}$, we should carry out transfer in the latent space of our GM, to predict the new face from $(\mathbf{z}_s^A, \mathbf{z}_a^B)$ instead. The corresponding error is denoted MSE Transfer Z.

For the experiments we generate a test set of ground truth triplets $\mathbf{x}^A$, $\mathbf{x}^B$ and $\mathbf{x}^C$ using the BFM face generator as explained above and projecting them into the hedgehog representation. The experiment is carried out using 100 randomly generated triplets. Using this dataset we compare two linear face models, both with 100 latent components for shape and appearance. The first GM models the shape and appearance completely separately (i.e. independent PPCA models), while the second GM is the disentangled linear model. The first GM is used as a reference to assess the improvement when using our disentangled model.

Unfortunately for the **cars experiment**, we do not have ground truth (GT) meshes with transferred appearances. We address this by manually segmenting two regions of cars: the body and the roof. Ideally we would segment all the possible parts of the cars but the body and the roof colours can already give a good sense of the transfer ability of our models. With these car segmentations we can swap the colours of any of the N_{test} segmented test cars (where $N_{test} = 15$). We define the ground truth triplet as follows: Given two cars A, and B, we generate the ground truth C by transferring the colours of the segmented roof and body of car B into car A. As with the face experiment, we compute the MSE Transfer X as $E_a(\mathbf{x}_a^C, \mathbf{x}_a^B)$. For the DVAE model, we compute the MSE Transfer Z as the error between $\mathbf{x}_a^C$ and the appearance generated from $(\mathbf{z}_s^{\ell,A}, \mathbf{z}_a^{\ell,A}, \mathbf{z}_c^B)$.

Note that the sum in Eq. 5.10 is only defined over the rays that we have manually segmented for the cars (those that belong to either roof or body) since we do not have control over the appearance transfer of the other regions. We use a total of 56 randomly selected pairwise combinations of the segmented cars. As baselines we compare the DVAE with two linear models (separate and disentangled) that have the same number of LVs as the high-level LVs of the DVAE (i.e. 100 for shape and 130 for appearance).

5.5.3 Transfer Results

We first analyze the results using the **faces** test set. Table 5.1 first shows the reconstruction MSE of the shape and appearance variables separately (without any transfer), then the two transfer error metrics described above. As expected the disentangled model has lower shape and appearance reconstructions (due to having $\mathbf{z}_s \rightarrow \mathbf{x}_a$ connections). We also see (as expected) how the disentangled model results in clear gains when comparing MSE Transfer X with MSE Transfer Z.

Fig. 5.6 shows the results in transfer the experiment of two test faces. The first three columns show faces A, B and C. We can see how the Transfer Z face (synthesized from $(\mathbf{z}_s^A, \mathbf{z}_a^B)$) properly captures the shape of face A and appearance of B. In addition to the synthesized faces following the processes Transfer Z and Transfer X, we also show two plots that visualize the corresponding errors with respect to the GT appearance of C. Dark gray in the faces indicate no change, and the magnitude of the red colour encodes the degree of error (i.e. squared difference from the GT). It is interesting to notice how the main sources of errors in MSE Transfer X are in the parts of the appearance that change the most when the shape changes (i.e. eyes, lips, nose and ears).

Table 5.1: Faces reconstruction and transfer errors with 95% confidence intervals.

	MSE Shape	MSE Appear.	MSE Transfer Z	MSE Transfer X
Linear Separate	0.0441 ±0.0001	0.0085 ±0.0011	0.0113 ±0.0009	0.0105 ±0.0010
Linear Disentangled	**0.0380** ±0.0010	**0.0048** ±0.0001	**0.0076** ±0.0005	0.0105 ±0.0010

For the **cars** experiment, Table 5.2 shows that the linear models have quite a high MSE of shape and appearance compared to the DVAE. But the more interesting aspect is the gains we see when comparing MSE Transfer X and MSE Transfer Z. For the disentangled linear model we see no improvement, which suggests that car meshes are perhaps too complicated to be linearly modelled. On the other hand we see a significant improvement of the DVAE model with a 0.039 error of the MSE Transfer Z, compared to 0.172 for Transfer X. This suggests that the shape and colours of the cars are properly disentangled in the DVAE.

Fig. 5.7 shows the transfer of the appearance for two test examples. Indeed, we can see how the DVAE model captures the correct way in which shape and colour are structured compared to the linear model. There is still room of improvement however in its ability to capture some of the finer detail (e.g. sometimes the colour of the lights and windows are not transferred properly).

	MSE Shape	MSE Appear.	MSE Transfer Z	MSE Transfer X
Linear Separate	0.144 ±0.020	0.046 ±0.006	0.122 ±0.032	0.172 ±0.039
Linear Disentangled	0.141 ±0.019	0.047 ±0.005	0.126 ±0.031	0.172 ±0.039
DVAE	**0.038** ±0.005	**0.025** ±0.006	**0.039** ±0.012	0.172 ±0.039

Table 5.2: Cars reconstruction and transfer errors with 95% confidence intervals.

5.5.4 Sampling

A standard way to evaluate the quality of a GM is by looking at its samples. Fig. 5.8 shows samples from the disentangled linear model for faces, and how they change when we only sample from either of the shape or appearance latent variables. What is interesting to notice is how the shape component has learned to capture the fine-grained features that depend mostly on the structure of the face mesh, while the appearance variables capture the global colour information across the whole face. This effect is

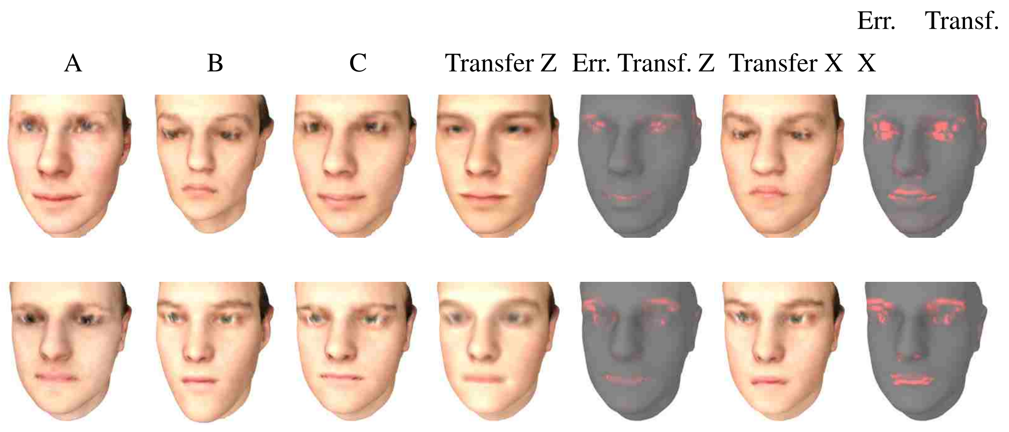


Figure 5.6: Transfer of face appearances. from left to right we show the original models A and B, the GT combination C, the result of the Transfer Z process using the linear disentangled model, the errors visualization for the Transfer Z, the result for Transfer X, and the error visualization for Transfer X.

evidenced by comparing the errors of the third and fifth columns; notice how the high error regions in the third column (when we sample $\mathbf{z}_s$) the high error regions happen in the nose, mouth and eyebrows which is the regions in which colour change the most as the shape of a face changes.

In Fig. 5.9 we can see a number of samples from the disentangled linear **car** model which clearly show that the model is not able to capture the structure of cars well. Fig. 5.10 shows four samples from the DVAE via ancestral sampling. These samples indeed look much sharper and well defined, even though they may sometimes have artifacts and may lack some of the rich variability in styles from the training set. What is more interesting is that it seems that the model has learned to segment the appearance of the cars in terms of different semantic parts (roof, body, windows, lights) without having any explicit segmentations in the data.

Sampling from the prior can sometimes give rise to improbable samples if the learned model does not fully exploit the latent space (e.g. if there is still manifold structure in there). A sensible alternative is to make use of the "aggregated posterior" (see Yang et al. (2012) for a definition), which in our case would give rise to a latent distribution of $\frac{1}{N}\sum_{i=1}^{N} q(\mathbf{z}|\mathbf{x}^i)$, where $q(\mathbf{z}|\mathbf{x})$ is the distribution produced by the encoder network, as described in Sec. 5.3.2. This is a mixture distribution centered around each of the training examples. We carry out "aggregated posterior" sampling for the higher-level

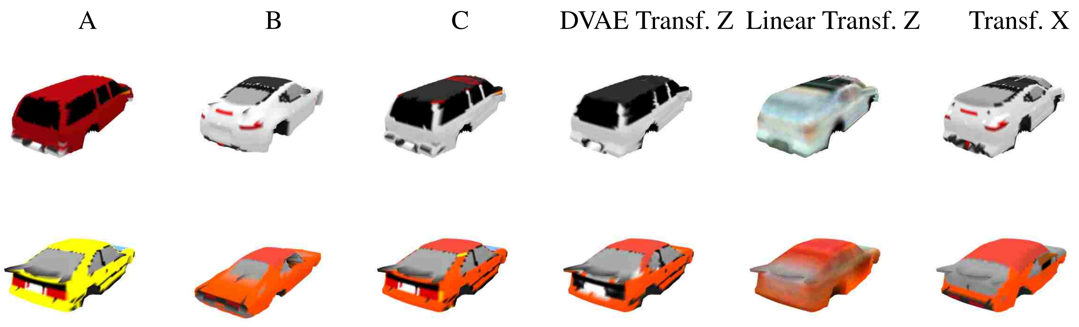


Figure 5.7: Transfer of car appearances. From left to right: the original models A and B, the GT combination C, the result of the Transfer Z process using the DVAE, the result of the Transfer Z process using the linear model, and the result for Transfer X.

latent variables and, conditioned on those, we sample from the intermediate (via Pixel-CNN) and observed variables via the GM. Fig. 5.11 shows a number of samples using this method. We can see how sampling the intermediate appearance variables changes the fine-grained details (second column) while sampling from the palette latent variables properly changes the whole colours of the car (third column). In Fig. 5.12 we perform the same sampling procedure, but instead show two samples of $\mathbf{z}_c$ that exhibit the variability captured in the colour model.

Finally, in Fig. 5.13 we demonstrate the fact that with our learned model we can sample a 3D mesh which can be embedded in any 3D scene. Two cars generated using the "aggregated posterior" sampling method. We also include wheels for an improved visual effect.

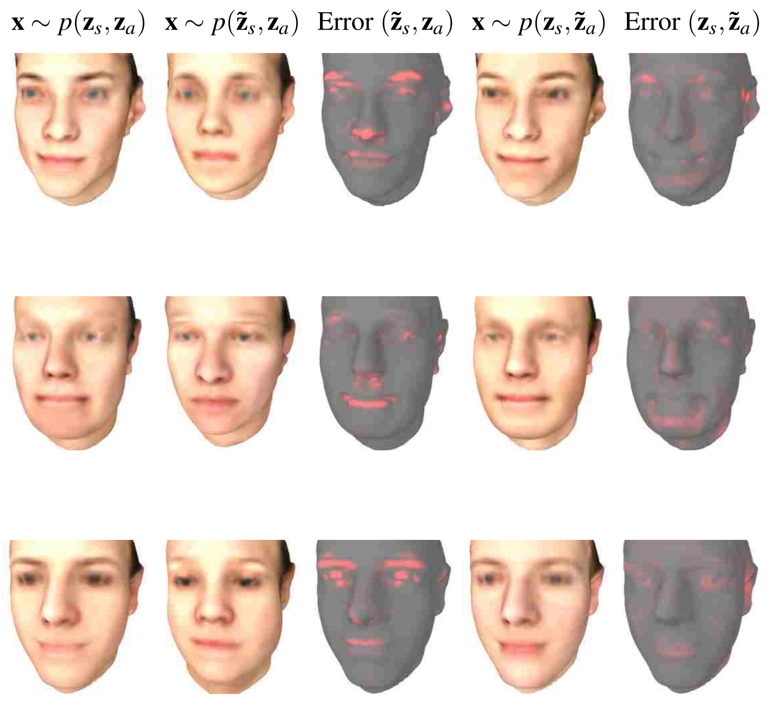

$\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a)$
$\mathbf{x} \sim p(\tilde{\mathbf{z}}_s, \mathbf{z}_a)$
Error $(\tilde{\mathbf{z}}_s, \mathbf{z}_a)$
$\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a)$
Error $(\mathbf{z}_s, \tilde{\mathbf{z}}_a)$

Figure 5.8: Samples from the disentangled linear model. From left to right: for every sample in the first column, we first fix the appearance and re-sample shape, then show the error in appearance with respect to the initial sample. Alternatively, we fix the shape and re-sample the appearance, and show the error with respect to the initial sample.

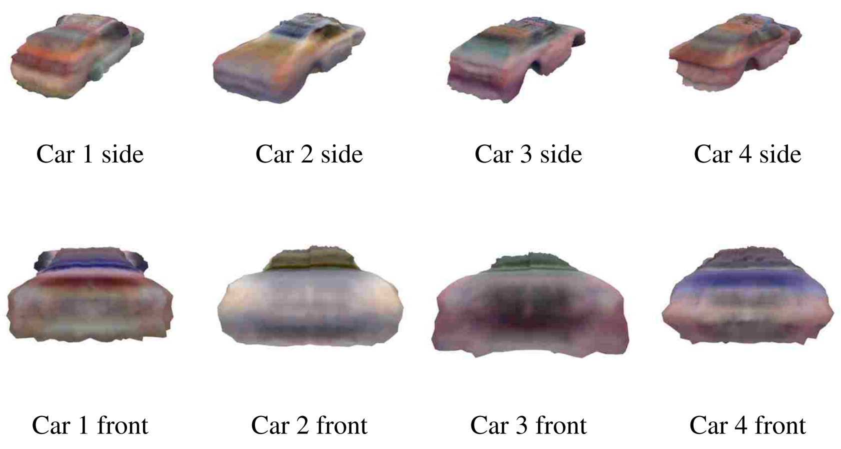


Figure 5.9: Samples from the car disentangled linear model.

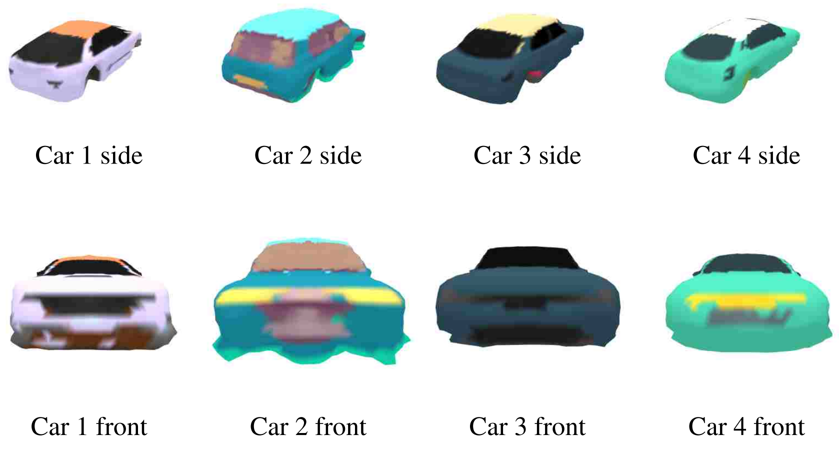


Figure 5.10: Samples of the car DVAE model.

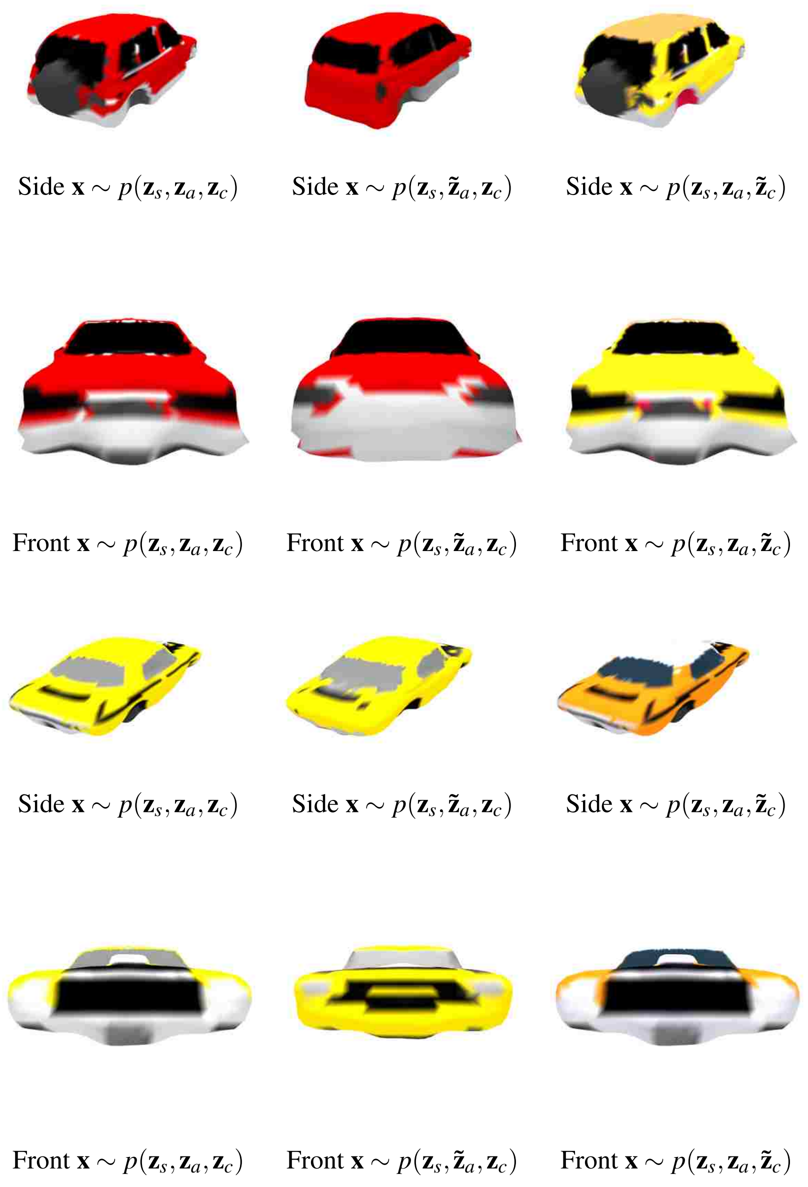

Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$

Figure 5.11: Aggregated posterior sampling of the car DVAE model. The first column is the training set reconstruction. Columns two and three show how sampling from different groups of latent variables change the cars. Sampled cars are shown from the side and front views.

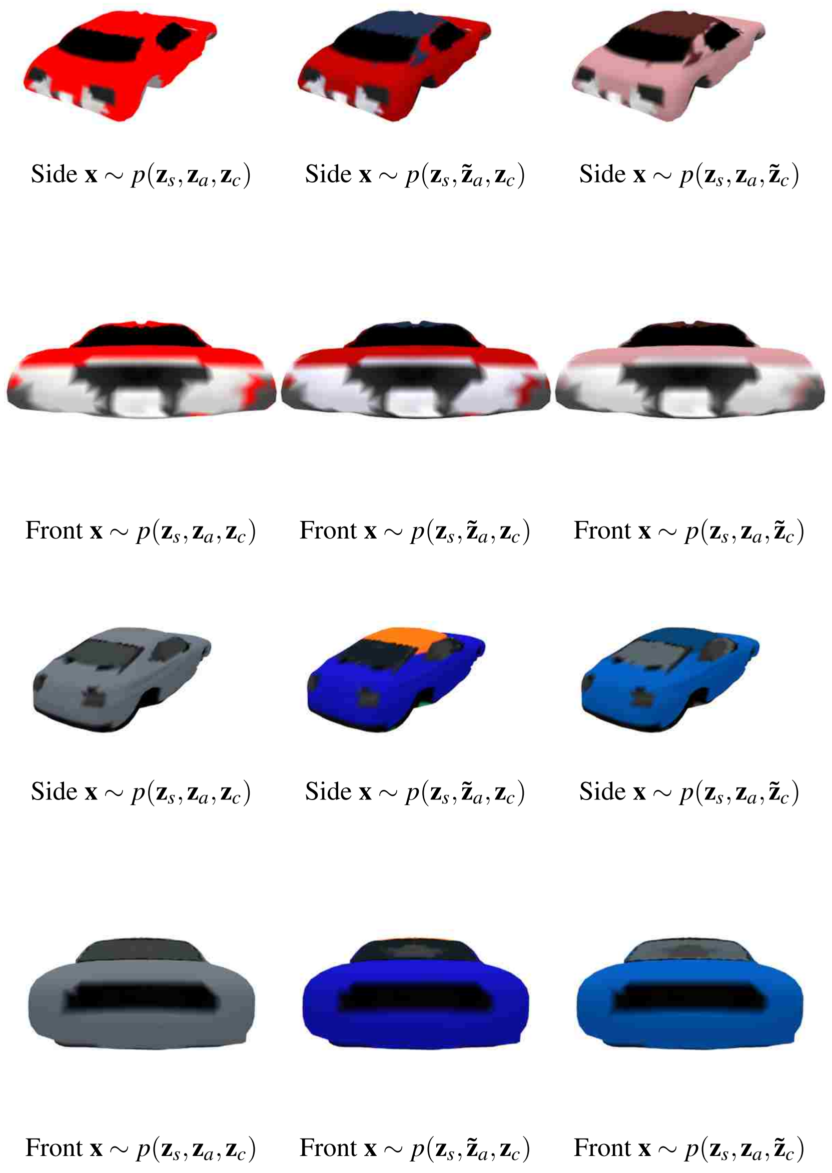
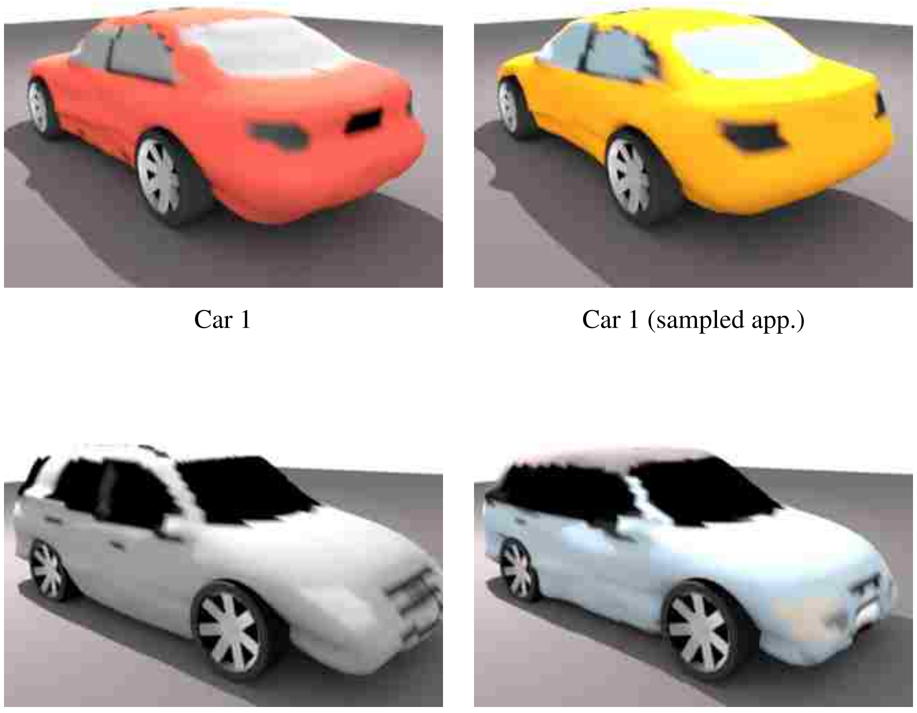

Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Side $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \tilde{\mathbf{z}}_a, \mathbf{z}_c)$
Front $\mathbf{x} \sim p(\mathbf{z}_s, \mathbf{z}_a, \tilde{\mathbf{z}}_c)$

Figure 5.12: Aggregated posterior sampling of the car DVAE model. The first column is the training set reconstruction. Columns two and three show how different samples from the colour $\mathbf{z}_c$ while leaving the remaining fixed as per the posterior sample. Sampled cars are shown from the side and front views.



Car 1

Car 1 (sampled app.)

Car 2

Car 2 (sampled app.)

Figure 5.13: Samples visualized with Cycles. These are generated using aggregated posterior sampling of $\mathbf{z}_s^\ell$ and $\mathbf{z}_a^\ell$ (first column), and conditioned on those, sampling from the palette and intermediate layers (second column).

5.6 Discussion

In this chapter we addressed two challenges to obtaining 3D generative models of shape and appearance of objects. The first relates to using the under-explored "hedgehog" representation of the data to avoid the need for the complicated process of obtaining mesh landmark correspondences. In this regard we investigated the use of a representation that had not yet been fully exploited in the literature as a model of both shape and appearances of objects. The "hedgehog" representation is well suited for a number of classes, among which are the faces and cars. We also provided an analysis of the functional relationship of the hedgehog representation with respect to 3D meshes.

The second is the notion of disentangled representations. Our appearance transfer experiments (both quantitative and qualitative) showed the ability of a disentangled model to factor out shape and appearance. For faces a linear model gave good performance, but for the more complex cars class the nonlinear DVAE was superior.

For cars and other complicated classes of objects, we proposed an auto-encoder architecture with the ability to learn high level structure (e.g. the coarse shapes of cars) and intermediate level variability (e.g. fine detail in shape and appearance) in a disentangled way. Our inclusion of a palette model acts as an inductive bias for the GM to segment the cars in its different semantic parts, as we saw when sampling from the different latent variables. Another important aspect of the architecture is the use of convolutions in order to take advantage of the local structure of the hedgehog representation. The appearance transfer quantitative experiments on a test set of manually segmented cars suggest that the learned car representations can be used to produce novel cars by combining the latent shape and appearances factors of different cars.

One of the most popular deep learning approaches to modelling 3D shapes uses volumetric representations as an alternative way to avoid finding landmark correspondences. As we mentioned before, these representations are computationally and memory-wise inefficient. An additional issue with the voxel approach is that the conversion from voxels to a data structure that can be rendered usually involves a procedure such as Marching Cubes (Lorensen and Cline, 1987), whereas in our case the ray distance representation directly maps to a mesh. The benefits of the voxel representation, however, is its flexibility to model any kind of shape. Tatarchenko et al. (2017) address the inefficiency of voxels with novel ways of modelling the volumetric representations using Octrees. Another recent work (Soltani et al., 2017) generates 3D shapes by taking as input multi-view images, depth maps and silhouettes of objects instead of voxels. To our knowledge none of these recent pieces of work include appearance information when modelling objects, leaving out an essential aspect of the description of an object.

There are a number of improvements that can be made to the DVAE. It will be interesting to try different ways of modelling the high-level latent representations. For instance, a straightforward improvement would be to allow multi-modal representations in the prior latent variables (e.g. mixture of Gaussians) to better capture the different clusters of shapes in cars.

An interesting possibility for classes that are not always as convex as faces or cars would be to independently model each of the parts with a hedgehog representation, and have a higher level model of the combination of these parts. In this regard, the work of Yi et al. (2016) can be used to obtain some of these object part segmentations.

We should also not lose aim of the central task of thesis. Our initial motivation for this work was to obtain richer models of 3D objects to be ultimately used in an analysis-by-synthesis framework. An interesting line of future work therefore involves applying these models of faces and cars on a scene understanding task (e.g. reconstruction of the shape and appearance of cars in an image).

Chapter 6

Conclusions and future work

The methods we have investigated in this thesis are designed to tackle the task of detailed scene understanding. We are, in other words, interested in obtaining representations from images in their most natural form: the set of physical elements that describe the underlying process that generated the image. In computer graphics this is often referred to as a scene graph, see e.g. Angel (2003, Sec. 9.8). That is, a data structure that encodes the 3D objects, illumination components, and one or multiple cameras in the scene. This scene graph representation, for instance, enables a user to interactively and intuitively modify its components and generate different scene arrangements, as we saw in Fig. 4.16. It is also an adequate representation for subsequent computer vision methods that carry out higher level reasoning about the scene in order to make complex decisions.

Given how broad the discipline of computer vision is, a large number of areas are outside the scope of consideration in this thesis (for instance, structure from motion methods have shown to be very effective for scene reconstruction and pose estimation, but rely on having multiple input images). The most relevant methods to our work are those that make inferences by taking a single image as input. Some of the most recent successful methods are based on data-hungry discriminative learning techniques. While in recent years we have seen great progress with respect to some of these methods, as we reviewed in Sec. 2.3, the task of detailed 3D understanding remains an active research area.

In one of the earlier works in the inverse graphics literature, Baumgart (1974, Sec. 6.3) compares different approaches to solving the general scene understanding problem and states that "[...] however, alone the statistical abstraction of world features in the presence of occultation, rotation and illumination seems as hopeless as the abstraction of a man's personality from the pattern of tea leaves in his cup". Over three decades later, perhaps it seems less hopeless now in the light of the success of deep learning of image features. Having said that, the increasing number of works based on inverse graphics in recent years, including the ones presented in this thesis, supports the notion that such an approach offers useful and principled methods to solving many scene understanding problems. I believe that it is the combination of deep learning methods with generative models of images that will enable scalable solutions to real world vision tasks.

6.1 Summary of contributions

In **Chapter 3** we described an inverse graphics solution to recovering the shape, pose, appearance and illumination from an object embedded in a scene. In order to train the recognition models and systematically evaluate the performance of our models we introduced a synthetic dataset which has rich variability of indoor scenes, object shape via a deformable shape model, and varying illumination obtained from environment maps of real scenes. In addition to making the dataset available, we provide the stochastic scene generator which allows researchers to produce different scenes with the object classes of their choice. We proposed a type of likelihood function that explicitly handles occlusion with a pixel-wise latent variable of occlusion. We showed that the gains of refinement using the robustified GM are consistently significant, up to a very large level of occlusion for the appearance and illumination latent variables. Furthermore, having the occlusion-aware GM allows us to make inference with regards to the probabilities of pixels being occluded, thus giving us fine-grained assessment of which parts of the objects are visible.

In **Chapter 4** we proposed enhancements to the GM that helped improve the refinement of all the scene latent variables. The scene dataset used in this task contains multiple objects per image, thus making the inference task a more interesting and challenging one. The recognition model architecture used to predict the camera latent variables, i.e. the Probabilistic HoughNet (PHNs), is designed to aggregate individual elements in order to robustly predict the intrinsic and extrinsic parameters of the camera. The combined global camera predictions given by the PHNs, and the illumination and object latent variables given by the CNNs are then fed to a generative model that iteratively optimizes over all the latent variables. We proposed using a method of precomputing self shadows in order to more accurately model the scenes. Results support the idea that a GM is more effective at fitting (refining) a scene if it is able to more accurately model the scene. We also analyzed the effect of rasterization with regards to image derivatives and proposed sampling methods that improved the gradient-based refinement results.

In **Chapter 5** we focused on obtaining richer 3D models of objects. For inverse graphics to be a useful framework for scene understanding, the GM should be able to capture the complex variability of objects in real images, while also separately modelling the underlying factors of shape and appearance of the objects in a scene. The main contribution of the chapter is a generative model structure that learns to disentangle shape from appearance in its latent representation. We also propose a way to quantitatively evaluate such disentanglement via the appearance transfer experiment. Results also show the usefulness of the “hedgehog” representation of 3D meshes as a way to overcome the limitations of other approaches (such as the landmark based approach and the volumetric representation).

We wish to emphasize that all the methods we have investigated are designed to be flexible and applicable to a large variety of scene and object domains. The use of a general purpose differentiable graphics renderer as the core of our GM in chapters 3 and 4 means it can take any type of object as long as it is represented as a 3D mesh. Nowadays, we can easily obtain 3D meshes from a large variety of object classes from datasets such as ShapeNet (Chang et al., 2015).

6.2 Future work

We conclude with an overview of what we believe are some interesting research questions with respect to inverse graphics. Some of the directions described below follow from the work presented in this thesis, while other ideas are based on a more general assessment of the state-of-the-art methods at the moment of writing.

The ideal way to handle **occlusions** when using a GM would be to actually model all the objects in the scene (in which case we would not need to explain it away with an occlusion model). Reconstructing all the objects in a scene, however, might not always be a realistic goal. It therefore seems that strong occlusion models are an important part of a practical solution. While our RMs in Chapter 3 are occlusion-aware in the sense that the images used to train them have rich and plausible occlusions, they do not explicitly predict the occlusion mask. Predicting the occlusion mask can in turn be used to initialize the occlusion latent variables for the GM. The recent work of Yildirim et al. (2017) train RMs of faces with images containing explicit occlusions (like we do) while also predicting these occlusions. Similarly, Li et al. (2017) propose a sophisticated CNN architecture (also trained on synthetic images) to predict the 3D keypoints distribution as well as visibility (which includes occlusion) information. There are also interesting directions of work with regards to designing GMs with occlusion models. For instance, Egger et al. (2018) go beyond pixel-wise outlier models by integrating a

Markov Random Field segmentation model into the GM. The success of these different techniques encourage the combination of richer occlusion-aware architectures both in RMs and GMs in order to effectively handle complex occlusions in real images.

In Sec. 3.7 we concluded that detecting when the **GM iterative refinement** has fallen into an improbable basin of attraction is an interesting research problem. In chapters 3 and 4 we treated the GM latent variables as fixed but unknown quantities during inference. A sensible idea is to model these variables with probability distributions based on prior knowledge, or by estimating these distributions from data (i.e. empirical Bayes). This in turn can help to avoid falling into unlikely regions of the latent space. Search in a probabilistic latent variable model when using a complex generative process such as a graphics renderer usually involves sampling-based methods such as Markov Chain Monte Carlo (Kulkarni et al., 2015a; Jampani et al., 2015) and approximate Baysian computation (ABC) (Mansinghka et al., 2013; Kulkarni et al., 2014) in the likelihood-free case. Perhaps the combination of data-driven proposals and gradient-based search e.g. Hamiltonian Monte Carlo (for which we can use the OpenDR derivatives) can lead to an efficient search over regions closer to the global optimum. Having estimates of the posterior distributions can also give us information about the uncertainty of the estimates, which can be useful in subsequent decision making.

In this thesis we have made heavy use of **synthetic data**. This is a trend in recent work in which RMs are trained to make fine-grained predictions from images, partly due to the lack of labelled real image datasets. Our dataset with photorealistic illumination and plausible indoor scenes with clutter and occlusions attempt to close the gap between synthetic and real images. There is still, however, a number of interesting possibilities to further improve the performance of these models on real images. An important area of research is thus overcoming the domain adaptation performance gap (when models trained on synthetic images need to be applied to real images). A common way to do so is, for instance, to train the machine learning models (such as neural networks) to predict the variables of interest with large quantities synthetic data, and once trained use the smaller real image (and labelled) dataset to fine-tune the model's parameters in order to reduce the domain adaptation gap. However, more careful combination of real and synthetic datasets seem to be key to getting the benefits from both domains when training RMs, see e.g. Wu et al. (2017, 2016), in which real data and synthetic data are used to train different components of the models in the spirit of weakly labeled regimes.

In Chapter 5 we presented a generative model of 3D objects learned from datasets of meshes. The idea mentioned above of combining synthetic data with real data to obtain richer models is also relevant in this case. For example the collection of meshes we used to train cars or faces, as described in Sec. 5.5.1, might not cover all the variability seen in real cases. An interesting line of work is thus combining models of objects learned from such collections of meshes with techniques that learn the shapes and appearances directly from real images. See the works of Rezende et al. (2016); Kanazawa et al. (2018) for examples of learning shapes and textures of objects directly from images.

One of the long term aims of scene understanding is to develop methods that can be applied to **complex scenes** with a wide range of objects and variability, such as the scenes that people interact with on a daily basis. Methods that attempt to tackle the complexity of scenes with multiple objects of different classes need specific techniques to remain computationally efficient. There are interesting research directions with respect to the interaction between RMs and GMs in the context of attention based-mechanisms. The PHN architecture we used in Sec. 4.3.1 is an interesting way to handle a variable number of objects in parallel while robustly aggregating each of their predictions. One can alternatively process the objects in an image following a sequential attention-based process. Eslami et al. (2016) propose a neural architecture that sequentially combines an encoder (RM) and decoder (GM) architecture in order to parse the complete image. It will be interesting to see if the computer vision research community can draw more useful ideas (such as that of attention) from the bottom-up and top-down flows of information seen in biological neural mechanisms.

Bibliography

Akenine-Moller, T., Haines, E., and Hoffman, N. (2008). *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition.

Amberg, B., Romdhani, S., and Vetter, T. (2007). Optimal step nonrigid ICP algorithms for surface registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2007*.

Angel, E. (2003). *Interactive Computer Graphics: A Top-Down Approach Using OpenGL (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Averkiou, M., Kim, V. G., Zheng, Y., and Mitra, N. J. (2014). Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In *Computer Graphics Forum*, volume 33, pages 125–134. Wiley Online Library.

Bach, F. R. and Jordan, M. I. (2005). A probabilistic interpretation of canonical correlation analysis. Technical Report TR 688.

Baldi, P. and Hornik, K. (1989). Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Ballard, D. H. and Brown, C. M. (1982). *Computer Vision*. Prentice Hall.

Barron, J. T. and Malik, J. (2015). Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687.

Barrow, H. G. and Tenenbaum, J. M. (1981). Interpreting Line Drawings as Three-Dimensional Surfaces. *Artificial Intelligence*, 17(1):75–116.

Basri, R. and Jacobs, D. W. (2003). Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233.

Baumgart, B. G. (1974). Geometric Modeling for Computer Vision. Technical report, Tech. Report AI Lab Memo AIM-249, Stanford University.

Belhumeur, P. N. and Kriegman, D. J. (1998). What is the set of images of an object under all possible illumination conditions? *International Journal of Computer Vision*, 28(3):245–260.

Belhumeur, P. N., Kriegman, D. J., and Yuille, A. L. (1999). The bas-relief ambiguity. *Int. J. Comput. Vision*, 35(1):33–44.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Binford, T. O. (1971). Visual perception by computer. In *IEEE Conference on Systems and Control*, volume 261, page 262.

Binford, T. O., Levitt, T. S., and Mann, W. B. (1987). Bayesian inference in model-based machine vision. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 73–96.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Black, M. J. and Rangarajan, A. (1996). On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91.

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th Annual Conference on Computer graphics and Interactive Techniques*, pages 187–194.

Blender Online Community (2017). *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.

Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014). Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 536–551. Springer.

Brooks, R. A. (1983). Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):140–150.

Chandler, B. and Mingolla, E. (2016). Mitigation of effects of occlusion on object recognition with deep neural networks through low-level image completion. 2016:1–15.

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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Choi, C. S., Okazaki, T., Harashima, H., and Takebe, T. (1991). A system of analyzing and synthesizing facial images. In *IEEE International Sympoisum on Circuits and Systems*, pages 2665–2668 Vol.5.

Cinbis, R. G., Verbeek, J., and Schmid, C. (2013). Segmentation driven object detection with Fisher vectors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2968–2975.

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates.

Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.

Crow, F. C. (1977). Shadow algorithms for computer graphics. *SIGGRAPH Computer Graphics*, 11(2):242–248.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893.

Davison, A. C., Hinkley, D. V., et al. (1997). *Bootstrap methods and their application*, volume 1. Cambridge university press.

de La Gorce, M., Fleet, D. J., and Paragios, N. (2011). Model-based 3d hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1793–1805.

de La Gorce, M., Paragios, N., and Fleet, D. J. (2008). Model-based hand tracking with texture, shading and self-occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Debevec, P. (1998). Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA.

Dickinson, S. (2009). The Evolution of Object Categorization and the Challenge of Image Abstraction. *Object Categorization: Computer and Human Vision Perspectives*, pages 1–58.

Dickinson, S. J., Pentland, A., and Rosenfeld, A. (1992). 3-d shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):174–198.

Dosovitskiy, A., Springenberg, J. T., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Egger, B., Schönborn, S., Schneider, A., Kortylewski, A., Morel-Forster, A., Blumer, C., and Vetter, T. (2018). Occlusion-aware 3d morphable models and an illumination prior for face image analysis. *International Journal of Computer Vision*, pages 1–19.

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 Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3225–3233.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2008). The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Fidler, S., Dickinson, S., and Urtasun, R. (2012). 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 611–619.

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*, 24(6):381–395.

Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., and Hanrahan, P. (2012). Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):135:1–135:11.

Fletcher, R. (1987). *Practical Methods of Optimization; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA.

Fransens, R., Strecha, C., and Van Gool, L. (2006). A Mean Field EM-algorithm for Coherent Occlusion Handling in MAP-Estimation Problems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 300–307.

Gao, T., Packer, B., and Koller, D. (2011). A segmentation-aware object detection model with occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1361–1368.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587.

Glasner, D., Galun, M., Alpert, S., Basri, R., and Shakhnarovich, G. (2011). Viewpoint-aware object detection and pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1275–1282.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Grenander, U. (1976). *Lectures in Pattern Theory: Vol. 1 Pattern Synthesis*. Springer-Verlag.

Grenander, U. (1978). *Lectures in Pattern Theory: Vol. 2 Pattern Analysis*. Springer-Verlag.

Grimes, D. B. and Rao, R. P. N. (2005). Bilinear sparse coding for invariant vision. *Neural Computation*, 17(1):47–73.

Grimson, W. E. L. and Lozano-Perez, T. (1984). Model-based recognition and localization from sparse range or tactile data. *The International Journal of Robotics Research*, 3(3):3–35.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vázquez, D., and Courville, A. C. (2017). Pixelvae: A latent variable model for natural images. In *The International Conference on Learning Representations (ICLR)*.

Gupta, A., Efros, A. A., and Hebert, M. (2010). Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 482–496. Springer.

Guzman-Rivera, A., Kohli, P., Glocker, B., Shotton, J., Sharp, T., Fitzgibbon, A., and Izadi, S. (2014). Multi-Output Learning for Camera Relocalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1114–1121.

Haag, M. and Nagel, H.-H. (1999). Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Image Sequences. *International Journal of Computer Vision*, 35(3):295–319.

Hara, K., Nishino, K., and Ikeuchi, K. (2005). Light source position and reflectance estimation from a single view without the distant illumination assumption. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 27(4):493–505.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hejrati, M. and Ramanan, D. (2012). Analyzing 3d objects in cluttered images. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 593–601.

Hejrati, M. and Ramanan, D. (2014). Analysis by synthesis: 3d object recognition by object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2449–2456.

Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. (2016). Early Visual Concept Learning with Unsupervised Deep Learning. *ArXiv e-prints*.

Hoiem, D., Chodpathumwan, Y., and Dai, Q. (2012). Diagnosing error in object detectors. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 340–353, Berlin, Heidelberg. Springer-Verlag.

Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15.

Huang, H., Kalogerakis, E., and Marlin, B. (2015a). Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. *Computer Graphics Forum*, 34(5).

Huang, H., Kalogerakis, E., and Marlin, B. (2015b). Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. In *Computer Graphics Forum*, volume 34(5), pages 25–38. Wiley Online Library.

Ittelson, W. H. (1952). *The Ames demonstrations in perception; a guide to their construction and use.* Princeton University Press.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87.

Jampani, V., Nowozin, S., Loper, M., and Gehler, P. V. (2015). The Informed Sampler: A discriminative approach to Bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32–44.

Jojic, N., Caspi, Y., and Reyes-Gomez, M. (2004). Probabilistic index maps for modeling natural signals. In Chickering, D. M. and Halpern, J. Y., editors, *UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*, pages 293–300. AUAI Press.

Kajiya, J. T. (1986). The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA. ACM.

Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V. (2012). A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55.

Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. (2018). Learning Category-Specific Mesh Reconstruction from Image Collections. *ArXiv e-prints*.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

Keselman, Y. and Dickinson, S. (2001). Bridging the representation gap between models and exemplars. In *Proceedings of IEEE Workshop on Models versus Exemplars in Computer Vision*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *The International Conference on Learning Representations (ICLR)*.

Kingma, D. P. and Welling, M. (2014). Stochastic gradient VB and the variational autoencoder. In *The International Conference on Learning Representations (ICLR)*.

Koffka, K. (2013). *Principles of Gestalt psychology*. Routledge.

Koller, D., Weber, J., and Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In Eklundh, J.-O., editor, *Proceedings of the European Conference* *on Computer Vision (ECCV)*, pages 189–196, Berlin, Heidelberg. Springer Berlin Heidelberg.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances In Neural Information Processing Systems 25*, pages 1106–1114.

Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., and Rother, C. (2015). Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 954–962.

Kulkarni, T. D., Kohli, P., Tenenbaum, J. B., and Mansinghka, V. K. (2015a). Picture: a probabilistic programming language for scene perception. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4390–4399.

Kulkarni, T. D., Mansinghka, V. K., Kohli, P., and Tenenbaum, J. B. (2014). Inverse Graphics with Probabilistic CAD Models. *ArXiv e-prints*.

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. (2015b). Deep Convolutional Inverse Graphics Network. *Advances in Neural Information Processing Systems 28*, pages 2539–2547.

Kulkarni, T. D., Yildirim, I., Kohli, P., Freiwald, W. A., and Tenenbaum, J. B. (2014). Deep Generative Vision as Approximate Bayesian Computation. *Approximate Bayesian Computation (ABC) Workshop. Neural Information Processing Systems (NIPS)*.

Leotta, M. J. and Mundy, J. L. (2011). Vehicle surveillance with a generic, adaptive, 3d vehicle model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1457–1469.

Li, C., Zia, M. Z., Tran, Q. H., Yu, X., Hager, G. D., and Chandraker, M. (2017). Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 388–397.

Lim, J. J., Khosla, A., and Torralba, A. (2014). FPM: Fine pose Parts-based Model with 3D CAD models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 478–493. Springer.

Lim, J. J., Pirsiavash, H., and Torralba, A. (2013). Parsing ikea objects: Fine pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2992–2999.

Liu, T., Chaudhuri, S., Kim, V. G., Huang, Q.-X., Mitra, N. J., and Funkhouser, T. (2014). Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 33(6).

Loper, M. M. and Black, M. J. (2014). OpenDR: An Approximate Differentiable Renderer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 154–169. Springer.

Lopez-Sastre, R. J., Tuytelaars, T., and Savarese, S. (2011). Deformable part models revisited: A performance evaluation for object category pose estimation. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1052–1059.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In Stone, M. C., editor, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987*, pages 163–169. ACM.

Lowe, D. G. (1985). Perceptual organization and visual recognition. *Kluwer international series in engineering and computer science. Robotics: vision, manipulation and sensors*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157.

Mansinghka, V. K., Kulkarni, T. D., Perov, Y. N., and Tenenbaum, J. (2013). Approximate bayesian image interpretation using generative probabilistic graphics programs. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 1520–1528.

Marr, D. and Nishihara, H. K. (1978). Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 200(1140):269–294.

Mockus, J. (1974). On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pages 400–404, London, UK, UK. Springer-Verlag.

Möller, T. and Trumbore, B. (2005). Fast, minimum storage ray/triangle intersection. In Fujii, J., editor, *32. International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2005, Los Angeles, California, USA, July 31 - August 4, 2005, Courses*, page 7.

Nash, C. and Williams, C. K. I. (2017). The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Computer Graphics Forum*, 36(5):1–12.

Nash, S. G. (1984). Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788.

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729.

Paysan, P., Knothe, R., Amberg, B., Romdhani, S., and Vetter, T. (2009). A 3d face model for pose and illumination invariant face recognition. In Tubaro, S. and Dugelay, J., editors, *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2009, 2-4 September 2009, Genova, Italy*, pages 296–301.

Pentland, A. P. (1986). Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331.

Pepik, B., Gehler, P., Stark, M., and Schiele, B. (2012). 3d2pm – 3d deformable part models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 356–370, Firenze. Springer.

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the Fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 143–156. Springer.

Persson, E. (2011). Geometric post-process anti-aliasing. ”`http://humus.name/index.php`”.

Pineda, J. (1988). A parallel algorithm for polygon rasterization. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’88, pages 17–20, New York, NY, USA. ACM.

Prince, S. J. D. (2012). *Computer Vision: Models, Learning and Inference*. Cambridge University Press.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. (2017). Neural episodic control. In Precup, D. and Teh, Y. W., editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 2827–2836. PMLR.

Ramamoorthi, R. (2006). Modeling illumination variation with spherical harmonics. pages 385–424. Elsevier.

Ramamoorthi, R. and Hanrahan, P. (2001a). A signal-processing framework for inverse rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 117–128.

Ramamoorthi, R. and Hanrahan, P. (2001b). A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 117–128, New York, NY, USA. ACM.

Reinhard, E., Heidrich, W., , P., Pattanaik, S., Ward, G., and Myszkowski, K. (2010). *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann.

Rezende, D. J., Eslami, S. M. A., Mohamed, S., Battaglia, P., Jaderberg, M., and Heess, N. (2016). Unsupervised learning of 3d structure from images. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4997–5005.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T., editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China.

Rhodin, H., Robertini, N., Richardt, C., Seidel, H.-P., and Theobalt, C. (2015). A versatile scene model with differentiable visibility applied to generative pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 765–773.

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 *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*, pages 940–948.

Rowley, H. A., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *The International Conference on Learning Representations (ICLR)*.

Schneider, A., Schönborn, S., Egger, B., Frobeen, L., and Vetter, T. (2017). Efficient global illumination for morphable models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3865–3873.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and Lecun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.

Sloan, P.-P., Kautz, J., and Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 527–536.

Soltani, A. A., Huang, H., Wu, J., Kulkarni, T. D., and Tenenbaum, J. B. (2017). Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2511–2519. IEEE Computer Society.

Sumner, R. W., Schmid, J., and Pauly, M. (2007). Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Sez-Trigueros, D., Meng, L., and Hartnett, M. (2017). Enhancing convolutional neural networks for face recognition with occlusion maps and batch triplet loss. *Image and Vision Computing*.

Tan, T.-N., Sullivan, G. D., and Baker, K. D. (1998). Model-based Localisation and Recognition of Road Vehicles. *International Journal of Computer Vision*, 27(1):5–25.

Tang, Y., Salakhutdinov, R., and Hinton, G. (2013). Tensor Analyzers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 163–171.

Tang, Y., Salakhutdinov, R., and Hinton, G. E. (2012). Deep lambertian networks.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 2107–2115. IEEE Computer Society.

Tenenbaum, J. B. and Freeman, W. T. (2000). Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Schiel, B., and Van Gool, L. (2006). Towards multi-view object class detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1589–1596.

Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1799–1807.

Toshev, A. and Szegedy, C. (2014). DeepPose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660.

Tu, Z. and Zhu, S.-C. (2002). Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673.

van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. (2011). Segmentation as selective search for object recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1879–1886.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1747–1756.

Wang, X. and Gupta, A. (2016). Generative Image Modeling using Style and Structure Adversarial Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 9908, pages 318–335.

Wang, X., Han, T. X., and Yan, S. (2009). An HOG-LBP human detector with partial occlusion handling. In *IEEE International Conference on Computer Vision (ICCV)*, pages 32–39.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Williams, C. K. I., Revow, M., and Hinton, G. E. (1997). Instantiating deformable models with a neural net. *Computer Vision and Image Understanding*, 68(1):120–126.

Williams, C. K. I. and Titsias, M. K. (2004). Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, 16(5):1039–1062.

Williams, L. (1978). Casting curved shadows on curved surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 12, pages 270–274. ACM.

Wu, B. and Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, W. T., and Tenenbaum, J. B. (2017). MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Advances In Neural Information Processing Systems 30*, pages 540–550.

Wu, J., Xue, T., Lim, J. J., Tian, Y., Tenenbaum, J. B., Torralba, A., and Freeman, W. T. (2016). Single image 3d interpreter network. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, Cham. Springer International Publishing.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920.

Yang, L., Liu, J., and Tang, X. (2014). Object detection and viewpoint estimation with Auto-masking Neural Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 441–455. Springer.

Yang, Y., Salakhutdinov, R., and Hinton, G. E. (2012). Deep Mixtures of Factor Analysers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1123–1130.

Yeh, Y.-T., Yang, L., Watson, M., Goodman, N. D., and Hanrahan, P. (2012). Synthesizing Open Worlds with Constraints Using Locally Annealed Reversible Jump MCMC. *ACM Transactions on Graphics*, 31(4):1–11.

Yi, L., Kim, V. G., Ceylan, D., Shen, I., Yan, M., Su, H., Lu, A., Huang, Q., Sheffer, A., Guibas, L., et al. (2016). A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210.

Yildirim, I., Janner, M., Belledonne, M., Wallraven, C., Freiwald, W., and Tenenbaum, J. (2017). Causal and compositional generative models in online perception. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017, London, UK, 16-29 July 2017.*

Yildirim, I., Kulkarni, T. D., Freiwald, W. A., and Tenenbaum, J. B. (2015). Efficient analysis-by-synthesis in vision: A computational framework, behavioral tests, and comparison with neural representations. In *Proceedings of the Thirty-Seventh Annual Conference of the Cognitive Science Society*.

Yuille, A. and Kersten, D. (2006). ”Vision as Bayesian inference: analysis by synthesis?”. *Trends in Cognitive Sciences*, 10(7):301 – 308. Special issue: Probabilistic models of cognition.

Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73(2):213–238.

Zhang, N., Paluri, M., Rantazo, M., Darrell, T., and Bourdev, L. (2014). Panda: Pose aligned networks for deep attribute modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, R., Isola, P., and Efros, A. A. (2017). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 645–654. IEEE Computer Society.

Zheng, Y., Zemel, R. S., Zhang, Y.-J., and Larochelle, H. (2014). A Neural Autoregressive Approach to Attention-based Recognition. *International Journal of Computer Vision*, pages 1–13.

Zia, M. Z., Stark, M., Schiele, B., and Schindler, K. (2013). Detailed 3D Representations for Object Recognition and Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2608–2623.