Cross-dimensional Analysis for Improved Scene Understanding

Moos Hueting

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

December 19, 2017

I, Moos Hueting, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Acknowledgements

First off, I would like to thank my academic advisor Niloy Mitra for all the advice and direction he has given me throughout the last four years. His optimism and vision have been inspiring and uplifting, especially on those occasions where things got hard. His drive has more than once motivated me to push just that bit harder, with great results. I owe a lot of gratitude to the people in the SmartGeometry group. Aron, thanks for the countless discussions, coffee breaks, good (and bad) puns, and above all, foosball matches. Paul, thanks for being the ideal postdoc table mate. Nicolas, thanks for the good conversations, and showing me how to be a good C++ programmer. James, Bongjin, Tuanfeng, Carlo, Robin – thanks for all the lunch-time discussions, be it about international politics or the dangers of teleportation (I still would not enter the cabin). Many thanks to my other colleagues and friends. Clément, thanks for all the discussions, beer, food, and laughs, especially when things didn't seem so funny. Corneliu, thank you for your never-ending good humour and the impromptu nights of drinking. Peter, thank you for the silliness as well as the brainstorming sessions – the space next to my desk is going to miss you. Tara, Clément, and Tracy, thanks for being there! To all my friends in Amsterdam, thank you for being home, and for supporting me during the hard times. To my parents, thank you for the unconditional interest, love and support you have given me throughout the PhD as well as the rest of my life. This work would have been so much harder without it. Thank you to all my family, especially Tante Veer, as well as Carla & Freek for cheering me on from the sidelines. Finally, Lucy – where to begin? Thank you for your endless encouragement, for the constant fun and games, for the shared songs, and just for being you. We have deserved ourselves a celebration!

Abstract

Visual data have taken up an increasingly large role in our society. Most people have instant access to a high quality camera in their pockets, and we are taking more pictures than ever before. Meanwhile, through the advent of better software and hardware, the prevalence of 3D data is also rapidly expanding, and demand for data and analysis methods is burgeoning in a wide range of industries. The amount of information about the world implicitly contained in this stream of data is staggering. However, as these images and models are created in uncontrolled circumstances, the extraction of any structured information from the unstructured pixels and vertices is highly non-trivial. To aid this process, we note that the 2D and 3D data modalities are similar in content, but intrinsically different in form. Exploiting their complementary nature, we can investigate certain problems in a cross-dimensional fashion – for example, where 2D lacks expressiveness, 3D can supplement it; where 3D lacks quality, 2D can provide it. In this thesis, we explore three analysis tasks with this insight as our point of departure. First, we show that by considering the tasks of 2D and 3D retrieval jointly we can improve performance of 3D retrieval while simultaneously enabling interesting new ways of exploring 2D retrieval results. Second, we discuss a compact representation of indoor scenes called a "scene map", which represents the objects in a scene using a top-down map of object locations. We propose a method for automatically extracting such scene maps from single 2D images using a database of 3D models for training. Finally, we seek to convert single 2D images to full 3D scenes using a database of 3D models as input. Occlusion is handled by modelling object context explicitly, allowing us to identify and pose objects that would otherwise be too occluded to make inferences about. For all three tasks, we show the utility of our cross-dimensional insight by evaluating each method extensively and showing favourable performance over baseline methods.

Contents

1 Introduction **12**

2 Related Work **18**

2.1 Image and shape retrieval 18

2.2 Image and shape analysis 19

2.3 Scene understanding 22

3 CrossLink: Joint Understanding of Image and 3D Model Collections through Shape and Camera Pose Variations **25**

3.1 Introduction 25

3.2 Overview 28

3.3 Input and Data Representation 30

3.3.1 Rendering of the models 30

3.3.2 Feature extraction 31

3.4 3D Model Collection Filtering 33

3.4.1 3D model alignment 35

3.5 Image Collection Organization 38

3.5.1 Camera pose estimation 38

3.5.2 Modeling of classifier weights 40

3.5.3 2D repository re-sorting by shape 41

3.6 Evaluation 43

3.6.1 3D repository filtering using 2D 44

3.6.2 3D model alignment 46

3.6.3 2D repository view sorting 48
3.6.4 2D view classifier modeling 49
3.6.5 2D repository shape sorting 51
3.7 Exploring Image and 3D Model Collections 53
3.8 Conclusions and Future Work 54

4 Scene Structure Inference through Scene Map Estimation 57
4.1 Introduction 57
4.2 Method 59
4.2.1 Scene Map 60
4.2.2 Scene Map Inference Overview 62
4.2.3 Network 63
4.2.4 Non-maximum suppression 63
4.2.5 Rendering pipeline 65
4.3 Evaluation 67
4.3.1 Baseline 67
4.3.2 Training vs. test 69
4.3.3 Comparison 69
4.3.4 Effect of object density 70
4.4 Discussion and Conclusion 72

5 Finding Chairs in Indoor Scenes under Heavy Occlusion using Scene Statistics 74
5.1 Introduction 74
5.2 Motivation and overview 77
5.3 Method 79
5.3.1 Camera estimation 81
5.3.2 Keypoint maps 81
5.3.3 Candidate generation 84
5.3.4 Candidate selection 89
5.3.5 Iterative optimization 93

5.3.6 Model selection 95
5.3.7 Hyper parameters 95
5.3.8 Data 96
5.4 Evaluation 99
5.4.1 Ground truth annotation 99
5.4.2 Performance measures 100
5.4.3 Baseline methods 102
5.4.4 Comparison 104
5.4.5 Ablation study 105
5.5 Discussion 106

6 Discussion and Future Work **108**
6.1 Summary 108
6.2 Future work 110

Appendices **112**

A Full Results for Chapter 3 **112**

B Full Results for Chapter 5 **125**

C List of publications **149**

Bibliography **150**

List of Figures

1.1 In-the-wild, cross-dimensional analysis 12
1.2 Iconic images and 3D models 15
1.3 Non-iconic images 15

2.1 Crosslink vs. previous work 18
2.2 Mockup vs. previous work 21

3.1 Framework overview 25
3.2 Input collections 28
3.3 Rendered views 31
3.4 HOG features 32
3.5 3D re-sorting 35
3.6 Model co-alignment 36
3.7 Alignment algorithm visualization 37
3.8 Camera pose estimation 39
3.9 PCA of view classifier weights 41
3.10 Re-sorting baseline 42
3.11 3D filtering ROC 44
3.12 Co-alignment result 46
3.13 2D view classification 47
3.14 View classification precision 48
3.15 Histogram of error magnitudes 49
3.16 Background clutter 50
3.17 Modeled SVM performance 51

3.18 Attribute sorting 51
3.19 Shape estimation accuracy 52
3.20 Effect of data size 53
3.21 Exploration possibilities 54

4.1 Intended output 57
4.2 Scene map 60
4.3 Network architecture 60
4.4 Non-maximum suppression 64
4.5 Train/test split 66
4.6 Data samples 67
4.7 Qualitative results 68
4.8 Effect of scene density 71
4.9 Result on real data 72

5.1 Scene mockup example 74
5.2 Context example 76
5.3 Baseline sample 77
5.4 Decreasing context 78
5.5 Pipeline 80
5.6 Expected output 80
5.7 Vanishing point detection 82
5.8 Keypoint types 82
5.9 Gaussian lobes 83
5.10 Keypoint map postprocessing 85
5.11 Template PCA 86
5.12 Parameters estimated during the candidate fitting process. 87
5.13 GMM visualization 90
5.14 Mixture components 90
5.15 Candidate selection visualization 91
5.16 Max mixture model 94

5.17 Houzz samples 96
5.18 PBRS dataset 97
5.19 MTurk interface 98
5.20 Annotation tool 100
5.21 Intersection-over-union visualization 101
5.22 Baseline output 103
5.23 Qualitative results 105
5.24 Changes in performance under varied angle and IoU thresholds. . . 105

List of Tables

3.1 3D alignment accuracy 47

4.1 Baseline comparison 69
4.2 Effect of object density 70

5.1 Architecture performance 84
5.2 Network architecture 84
5.3 Hyper parameters 96
5.4 Quantitative performance 104
5.5 Ablation study 106

Chapter 1

Introduction

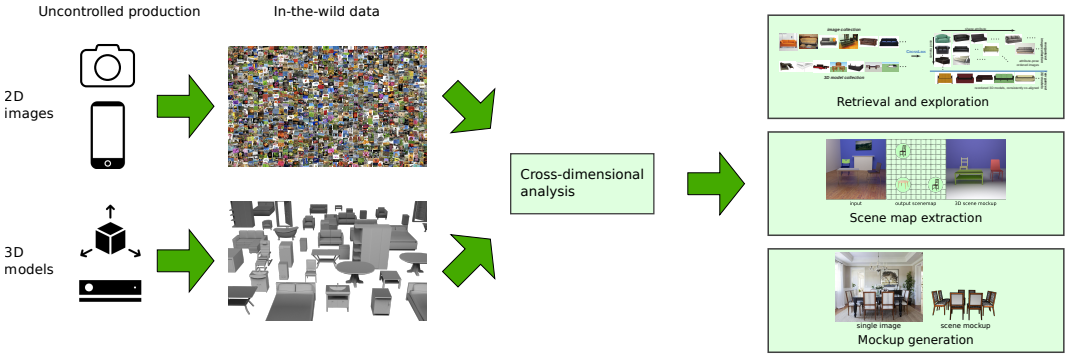


Figure 1.1: In this dissertation, multiple methods for the analysis of in-the-wild visual data are discussed. The emphasis lies on the role of cross-dimensional analysis, i.e. the exploitation of specific advantages of 2D and 3D data sources to garner success in the opposite domain.

The role of images in our society has never been greater than now. High quality photographs are available to all through the smartphone in our pockets, the amount of data being generated in this way is at an all time high, and through the internet a considerable part of these images is available to all, for free. The numbers are staggering: in 2017, the total number of photos captured by mobile devices and still cameras will reach 1.2 trillion [57]. In that same year, 2.3 billion people use a smartphone, with that number set to grow to 2.8 billion in 2020 [108].

These images constitute an extraordinary representation of the visual world around us. The higher level information contained within the pixels of a single image can be used for a wide range of tasks, but only if we can extract it. For example, object detection in an uncontrolled image can be used for image retrieval, helping

industrial automation, and informing self-driving cars, while high quality semantic segmentations and single image depth estimates can be used for path planning in uncontrolled, real-life scenarios. Furthermore, combining information extracted from large sets of uncontrolled images allows us to perform more high level inference tasks from the distribution of said information, such as discovering typical usage of certain types of objects for the purpose of product design and architecture, training semantically informed models for image editing, and creating realistic synthetic data for the purposes of simulation and entertainment.

By its nature, these images are *uncontrolled* – we have no control over how these photographs are taken, and thus cannot rely on rigid methods for extracting all this information. It is no surprise then, that the analysis of uncontrolled images, also called images "in the wild", has been focus of significant research efforts over the past years. This has resulted in performance increases in a variety of different image analysis tasks, ranging from object recognition [94] and semantic segmentation [85] to pose [124] and depth estimation [37], and applications abound in both industry and consumer markets.

At the same time, 3D data sources have become more commonplace. An abundance of 3D models is available and growing through public access to 3D search engines such as 3D Warehouse [115], Turbosquid [117], Sketchfab [105], CGTrader [17], and many others. Furthermore, the creation of depth images is easily accessible through the advent of depth sensors such as the Microsoft Kinect, Google's Project Tango, and Intel RealSense, as well as through many smartphones with dual cameras, enabling 3D photo and video through stereo vision. Demand for 3D data is also growing, with interest from many fields ranging from architecture and indoor design to virtual reality, digital fabrication, and the game industry. As with 2D data the need to understand and organize these data automatically has increased in step with this growth, for example yielding works in shape labeling [53], exploration [141], and semantic editing [134].

When considering both 2D and 3D data together, there are some important things to note. First, these two types of data are intrinsically different in a number

of ways. Photographs are usually captured and provide a high quality representation of the world, whereas 3D models are usually created by hand, with high variance in quality. On the other hand, 3D models provide a third dimension, yielding information that 2D images by definition lack. Then again, even though the amount of 3D data available has increased significantly, there is still a gap of multiple orders of magnitude with the vast number of 2D images that can be found on-line – the latest public estimate from 2010 puts the number of images indexed by Google at 10 billion [40], whereas the number of available models on 3D Warehouse and Turbosquid combined is around 3 million [115, 117].

Despite these differences it is also important to acknowledge the common ground between the 2D and 3D domains. After all, they ultimately represent the same subject, i.e. what we are actually interested in – our reality, the world in which we live. In other words, the tasks we set out to perform which use these data as input are usually concerned with understanding something about the world *represented* by them.

Taking into account the differences as well as the commonalities leads to the insight that for certain tasks relating to in-the-wild analysis, it could be useful to take into account *both* data types. In these cases, the common ground makes cross-dimensional analysis possible, while the differences ensure the availability of a greater amount of information than in the single-dimensional case. This can bring about solutions to problems that are otherwise impractical to solve, and in other cases lead to performance improvement. In this thesis, we will consider three analysis problems of images and models in-the-wild, while taking this insight of cross-dimensional linking into account (see Figure 1.1).

The first problem we will address arises when considering retrieval of iconic images and 3D models. An iconic image is, as the name suggests, an iconic exemplification of the object it represents. Figure 1.2, left shows an example of an iconic image of a car. The internet is full of such images, and through highly developed search engines such as Google and Bing images anyone can access thousands of high quality iconic images of nearly any object class. 3D object models (Figure 1.2,

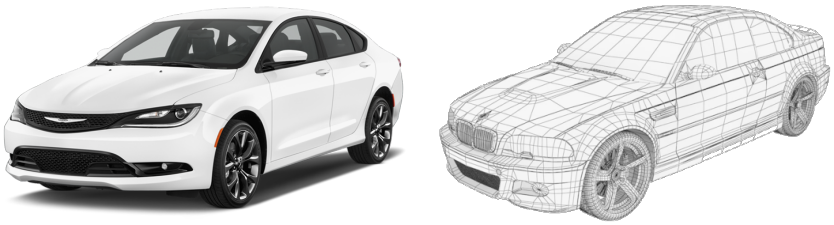


Figure 1.2: Iconic images and 3D models are available in large numbers on the internet, and have differing advantages and disadvantages

right) represent a similar concept for the 3D domain, and are also available in high numbers through 3D search engines. However, the retrieval quality of these engines still trails significantly behind that of the 2D search engines. On the other hand, the 3D models present us with geometry and pose information, which are lacking in the 2D domain. By considering the retrieval task in a joint fashion, we can improve the quality of 3D retrieval while enabling novel exploration methods of 2D retrieval results. This idea is explored in Chapter 3, where such a method is proposed. We show significant improvement in quality of 3D retrieval results, as well as new exploration methods of 2D retrieval results based on pose and geometry. These results are valuable in itself, but will also be of use when considering non-iconic images.

Figure 1.3: Non-iconic images such as indoor scenes make up a large part of the in-the-wild images on the web

Indeed, the majority of photographs are not iconic but depict *scenes*, which contain arrangements of multiple objects of different types. Many of these scene photographs are indoor, of which some typical examples can be seen in Figure 1.3. Many efforts have been made to understand these images automatically in different ways, such as finding objects (Ren et al., 2015 [94]), estimating depth (Godard et al., 2017 [37]), and semantic segmentation (Noh et al., 2015 [85]). All of these

methods try to infer 3D semantic and geometric structure of a 2D image – what 3D objects are present, and where they are. Limiting ourselves to objects that are placed on the ground, this information can be compactly summarized by a top-down map of the scene, henceforth called a *scene map*. In other words, when looking at the scene from top down, which objects are placed where? Automatically extracting this information is useful for many domains, such as path planning from single images, computing statistics of furniture usage from large datasets of indoor scene photographs, as well as scene type classification. In Chapter 4 we will investigate this problem, notably using the clean and co-aligned model sets resulting from Chapter 3 as a source for generating synthetic training data.

The resulting method's main weakness is occlusion. This is not so surprising – even we as humans need to use contextual information to reason about scenes under heavy occlusion. Moreover, for some purposes the format of the scene map is too coarse. For example, if we want the extracted 3D information to help us edit the input image in some way, we need more than just rough 3D location – we also need the pose of the 3D objects, as well as the pose and intrinsic parameters of the camera. In Chapter 5 we propose a method for generating a richer output called a *scene mockup*: a 3D scene consisting of 3D objects from a clean and co-aligned database (again collected from Chapter 3), together with an estimated camera, such that the reprojection of the scene is as close as possible to the input image. To deal with occlusion, we will introduce a model of object co-occurence, which helps in finding objects for which otherwise too little visual information would be available. This model is trained on a database of 3D indoor scenes, once more introducing cross-dimensional links for performance improvement. The method is shown to beat state-of-the-art methods.

The main contributions of this thesis are:

- a method for improving 3D model retrieval using results from high quality 2D image retrieval,
- a method for exploring 2D image retrieval results by view and shape using 3D model retrieval results,

- a multi-scale neural network setup for the estimation of top-down scene maps from single images,
- a method for synthesizing 3D scenes from 3D model collections for the purpose of training this neural network, and
- a multi-stage optimization framework for finding chairs in single images using keypoints from a neural network, as well as a learnt statistical model of object co-occurrence.

Chapter 2

Related Work

Analysis of in-the-wild 2D and 3D visual data has been explored in the context of many different tasks. The research most related to the work presented in this thesis can roughly be split into two different parts: methods which attempt similar tasks to the ones considered in this thesis, and methods whose goals are different but whose underlying insight (the linking of the 2D and 3D domains) is similar. Both types are discussed here.

2.1 Image and shape retrieval

2D image search. Many supervised and unsupervised methods have been developed for image retrieval (see recent surveys by Datta et al., 2008 [27] and Zhang et al., 2013 [136]). Broadly, the majority of methods rely on image tags or accompanying annotations and/or extracted image features (e.g., HOG, SIFT, PCA-SIFT,

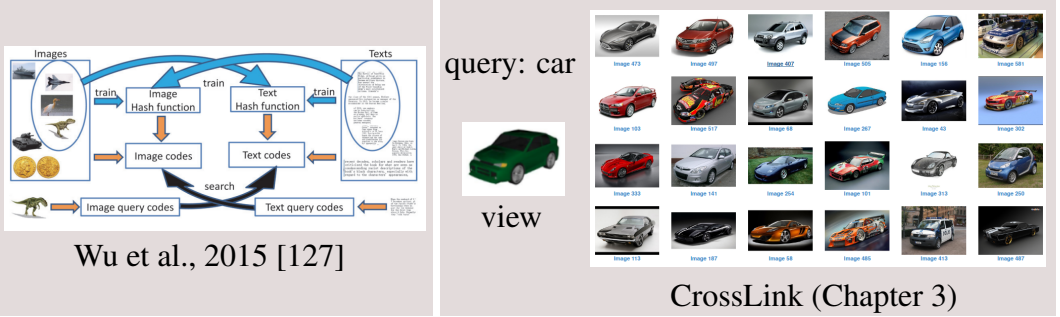


Figure 2.1: Many other works look at cross-modal retrieval, such as Wu et al., 2015 [127], which allows for retrieval of text using images and vice versa. In contrast, we do text-based retrieval of two different modalities jointly to improve performance and enable new exploration methods.

SURF, etc.) to train category-specific classifiers. More recently, correlation across multiple information channels (e.g., text, images) has been explored for better retrieval performance [125, 78, 88, 121, 127, 77]. With a similar motivation, in Chapter 3 a method is proposed to link and utilize information coming from 3D models for richer image search and exploration. In contrast to most methods, we are not looking to enable cross-modal search (i.e. search for images using shapes and vice versa) but to improve text-based search in both modalities by implicitly linking the retrieval process of both modalities (see Figure 2.2).

3D shape search. There has also been significant work on 3D shape retrieval from large collections. These techniques can be classified according to the type of query, and include *text-based*, or *content-based*, where a sample 3D shape is given and the goal is to retrieve similar ones, either *image-based*, or *sketch-based*. Although, in practice, text-based search is both simplest and most accessible, the poor quality of user-assigned tags in public 3D model collections means that the quality of pure text-based retrieval has so far been unsatisfactory [81, 38]. At the same time, content-based and sketch-based retrieval approaches, while often accurate [111, 30, 70, 120, 130], assume a 2D or 3D query, which can be non-trivial to obtain for a casual user.

2.2 Image and shape analysis

Image analysis. In the context of image collections, such as those returned by a 2D search engine, the grand goal is to annotate the content of each image and link it to an ontology of semantic concepts (e.g., ImageNet Visual Recognition Challenge [96] and references therein), often by using a large repository of ground truth annotations. While this line of work is fundamental, we argue that some properties, such as geometric attributes of objects (a 'narrow' chair) or camera pose are rarely present in the annotations of even the largest image collections. Therefore, using side information from a different modality can contribute to better overall image understanding, as discussed in Chapter 3. In the context of image collections of a single scene, learning discriminative patches has been proposed to characterize the

underlying 3D scene. For example, Srivastava et al. [101] proposed exemplar SVM to establish cross-domain image matching, while Aubry et al. [7] factor out various sketching effects to facilitate painting-to-3D alignment.

Semantic segmentation. Many traditional approaches for in-the-wild scene understanding are based on semantic segmentation, which tries to associate class labels to pixels in the image (see Gould et al., 2014 [42] for an overview of related methods). Most recently, successful techniques heavily exploit training data to guide semantic segmentation (e.g. [16, 20, 85, 21], among many others). Moreover, some recent approaches such as Handa et al. [44] have used synthetic (rendered) data to augment the training set resulting in more accurate labeling. In Chapters 4 and 5, similar extraction of semantic data from single images is investigated, but unlike these methods, the goal is not to associate class labels to image pixels, but to directly output an abstracted scene in the form of either a scene map or scene mockup, which summarizes the objects in the image *in scene coordinates*. In this way, our approach is related to techniques that estimate depth together with semantics (e.g. Eigen et al., 2015 [29]), although we avoid the error-prone depth estimation step by training on the scene maps and mockups directly.

Shape analysis. Analogously, in the context of 3D model collections, co-analysis approaches like Mitra et al. [82] have focused on extracting part-level anisotropic scale variations for characterizing style (Xu et al., 2010 [131]), linking point-level correspondence detection across shape variations to learn template-based shape variations (Kim et al., 2013 [64]), semi-supervised learning strategies for fine grained labeling of shapes collections (Huang et al., 2013 [50]), learning characteristic deformation directions from models collections (Yumer et al., 2014 [135]), performing semantic editing (Yumer et al., 2015 [134]), using functional maps to analyze unstructured model collections (Huang et al., 2014 [51]), or learning function of objects through interaction co-analysis (Hu et al., 2016 [49]). These methods rely on access to single-category collections free from outlier shapes. Often, such model collections are manually curated, which limits extensions to different classes. In Chapter 3 it is shown how large collections of natural images can be used to au-

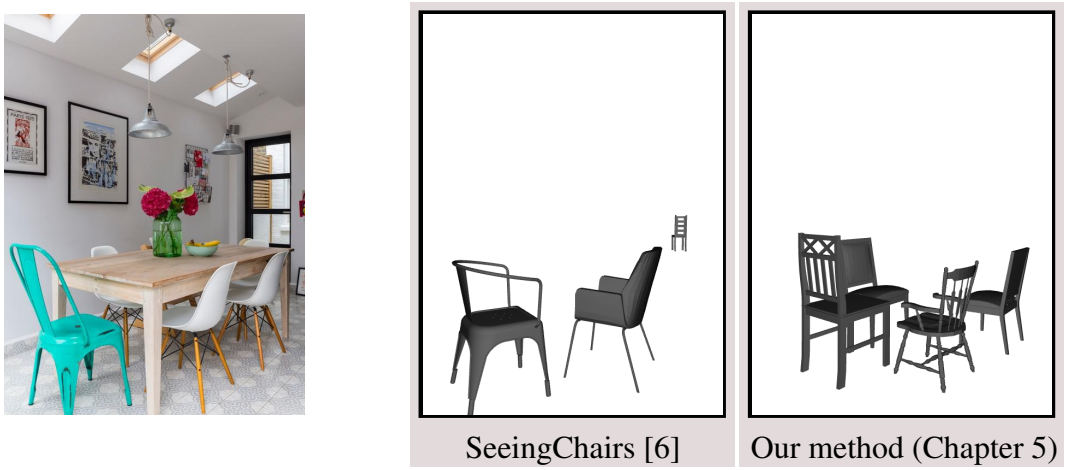


Figure 2.2: In the case of scene mockups, we manage to improve beyond existing baselines, such as Aubry et al., 2014 [6], by exploiting contextual information between objects in a scene. This allows us to reason about highly occluded objects, a case where other methods fail.

tomatically filter irrelevant 3D models and to co-analyse and explore collections of 3D models. The resulting collection of models can be used as input to any of the previously mentioned methods, and is used in this work to create object templates for mockup creation in Chapter 5.

Coupled image-shape analysis. The classic work on morphable faces by Blanz & Vetter [12] demonstrated the utility of modal analysis using point-level correspondence across shapes to computer graphics. Xu et al. [133] used model collections to perform part-based model synthesis using photographs for style guidance, while Li et al. [72] fuse photographs and LiDAR scans to create depth-layer decomposition of urban facades. More recently, Wang et al. [122] analyze different 2D projections of 3D shapes to transfer information from labeled images to consistently segment the 3D shapes. In two related efforts, Vicente et al. [119] use landmark-based correspondences to roughly estimate camera locations for images of different but related shape instances in an effort to reconstruct the VOC data, while Su et al. [109] estimate deformation fields regularized by a network of shapes to convert segmented images to corresponding depth maps. Note that in the later effort the input images are assumed to be presegmented. With a similar motivation, Aubry et al. [6] use renderings of 3D models from multiple viewpoints to train a pose classifier for

data-driven part-based 2D-3D alignment in a single manually curated object class. The work presented in Chapter 5 is similarly motivated, but uses added scene statistics information gleaned from a large database of 3D scenes to aid the process in case of heavily occluded objects (see Figure 2.2).

Other recent methods exploit large databases of 3D models to facilitate image analysis. Most notably, 3D model collections have been used for single-view reconstruction (Huang et al., 2015 [52]), object detection (Aubry et al., 2014 [6], Massa et al., 2015 [79]), view-point estimation (Su et al., 2015 [110]), scene parsing (Zhao et al., 2013 [139]), or even for learning generative models for object synthesis (Girdhar et al., 2016 [36]). Model collections are particularly useful as a source of additional training data that can be incorporated into learning algorithms for labeling (Handa et al., 2015 [44]) or pose estimation tasks (Chen et al., 2016 [22]), among many others [123, 129, 13]. The work presented in Chapter 4 and 5 overlap with this area, being most closely related to those methods that use 3D data as side information for scene understanding (e.g. Liu et al., 2015 [74] and Zhang et al., 2016 [137]).

Finally, Li et al. [71] propose a method for embedding shapes and images within the same embedding space via CNN image purification. This is in line with the goal of Chapter 3 of reasoning about image and 3D model collections simultaneously without explicit links. However, their method requires clean 3D model sets, while our pipeline is designed for the usually highly noisy model collections returned by common 3D model search engines.

2.3 Scene understanding

Pose estimation. Another line of work that reasons about 3D and 2D data jointly, is research in Camera Pose Estimation, which can also be stated as alignment of a single 3D object with a natural 2D image. Although a classical and well-studied problem, there are many variations that range in robustness and complexity (see Dambreville et al., 2008 [26], Corsini et al., 2009 [23], and Prisacariu et al., 2012 [92] among many others, as well as work by Russell et al. [97] applying this idea to align

historical architectural paintings with 3D models obtained using multi-view stereo). In Chapter 3 a method is proposed that achieves both *scalability* and *robustness* to large changes in geometry and appearance, and *avoids* any explicit correspondence. Our use of classifier smoothing and interpolation in this context can also be seen as a special-case of regularized multi-task learning (Evgeniou et al., 2004 [31]), where we exploit the circular nature of the parameter space. In addition, in Chapter 5 the co-aligned models resulting from Chapter 3 are used to extend the 3D object alignment problem to multiple instances.

Scene mockups. A number of methods have also been proposed for high-level scene understanding and labeling, by exploiting additional depth information available from RGB-D sensors [107, 43, 106]. The goals of Chapter 4 are similar, but we only use 2D image information at test time, and exploit rendered synthetic scenes for training. In Chapter 5 the extra information used comes in the form of a database of 3D models and scenes, which heavily regularize the mockup process. A similar recent technique by Bansal et al. [10] uses a database of 3D models and retrieves the closest model to a *given* bounding box in the image. In addition, they do dense normal estimation first, which again introduces additional complexity and a potential source of inaccuracies.

Most recently, Izadinia et al. [60] demonstrated scene reconstruction with CAD models from a single image using image based object detection and pose estimation approaches. Although their objective is similar to ours in Chapter 5, the performance is bounded by the individual vision algorithms utilized in their pipeline. This obstacle is similar to the one encountered by our second baseline, which is based on FasterRCNN [94] (see Section 5.4). For example, if FasterRCNN misses an object because of significant occlusion, there is no mechanism to recover it in the reconstruction. On the contrary, our novel pairwise based search incorporates high level relationships typical to indoor scenes to recover from such failures successfully.

Scene priors for reconstruction. Scene arrangement priors have been successfully demonstrated in 3D reconstruction from unstructured 3D input, as well as scene synthesis (Fisher et al., 2012 [34]). Shao et al. [99] demonstrated that scenes with

significant occlusion can be reconstructed from depth images by reasoning about the physical plausibility of object placements; a similar observation concerning the statistical plausibility of object placements is used by us in Chapter 5. Monszpart et al. [83] uses the insight that planar patches in indoor scenes are often oriented in a sparse set of directions to regularize the process of 3D reconstruction. Fisher et al. [35] leveraged human activity priors together with object relationships as a foundation for 3D scenes synthesis. In contrast to the complex and high order joint relationships used in these works, the object centric templates used in Chapter 5 aim to capture object co-occurence statistics using first order relationships between two object placements. This compact and simple template representation help ensure our search is tractable at runtime.

Chapter 3

CrossLink: Joint Understanding of Image and 3D Model Collections through Shape and Camera Pose Variations[1]

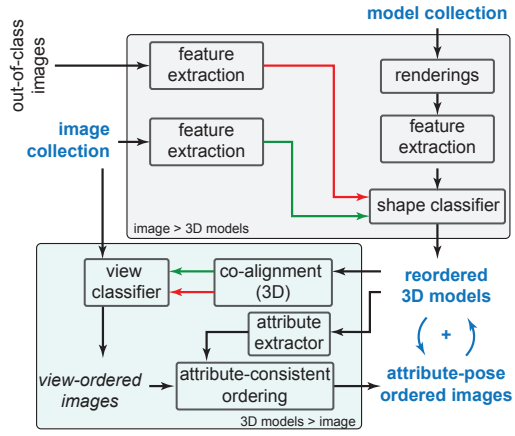


Figure 3.1: Overview of the proposed framework for joint understanding of class-labelled 2D image and 3D model collections.

3.1 Introduction

Image and model collections are ubiquitous and continue to grow rapidly. Analyzing and processing such collections has been the focus of a large body of work in

[1]Published at SIGGRAPH Asia 2015 [54]

computer graphics, computer vision and related fields over the past several decades. However, despite a great amount of progress, several tasks remain challenging, including text-based 3D model search and camera pose estimation in images, in large part due to the significant noise (e.g. in tags or annotations of 3D models), or the lack of training data (e.g. in camera pose estimation) in the unorganized online repositories.

At the same time, several recent techniques have been proposed for co-analysis and exploration of 3D model collections, leading to the area of *structure-aware shape processing* [82]. Works in this direction are motivated by the fact that the structure and relations between 3D objects are best understood within the context of other related models in a large repository. However, the vast majority of these techniques require a pre-filtered set of models falling within the same category ('car', 'chair', 'bicycle', etc.), and obtaining such a set often requires manual intervention, especially because most existing text-based 3D search approaches are based on user-generated tags, which can be noisy and unreliable.

Similarly, in the image domain, estimating *shape attributes*, such as geometric properties including height or width of the object, or the viewing angle/camera pose, is difficult even for the largest image collections, since these attributes are rarely provided as part user labelings. Therefore, training image classifiers that would be able to discriminate across object *views* or provide geometric information about the object in an image still remains challenging (e.g., 'show an image of a long bicycle from a particular viewing angle').

This domain is the first in which the main point of this thesis is explored. The strengths and weaknesses of 2D and 3D data retrieval are combined and put to the test as a joint system. Several key problems are addressed, for which significant quantitative improvement in performance over existing baseline methods are reported: text-based 3D search, 3D model co-alignment, as well as camera pose estimation and geometric property (height or width of an object) in 2D images. For each of these problems it is demonstrated how to exploit the strengths of different types of data to co-analyze image and 3D model collections for common object

categories. The ultimate objective is to allow better understanding and joint exploration of image and 3D model collections. This is achieved by improving 3D search results, consistently aligning the models, and exposing to the user an exploration interface, which allows image retrieval based on pose and shape, learned from 3D models. Furthermore, the resulting co-aligned model sets provide the basis for the scene understanding tasks explored in Chapters 4 and 5.

In creating this system, henceforth to be called CROSSLINK, several key technical challenges are solved: (i) show how image-based feature representations can be used to learn efficient classifiers on 3D shapes for better text-based search, (ii) propose an efficient 3D co-alignment procedure, based on a hybrid 2D-3D representation, (iii) demonstrate how this representation leads to efficient pose estimation by using the inherent periodic (spherical or circular) nature of the space of views of a 3D model, and finally (iv) develop an object shape estimation method in images, using 3D models for training. Note that although to solve the problems we heavily use classical techniques such as Support Vector Machines and non-linear regression, significant technical contributions are made to exploit the particular and novel structure of multimodal data at hand.

The proposed framework is extensively evaluated on 20 object categories obtained using the Bing Image Search and the Trimble 3D Warehouse. Several quantitative metrics are proposed to evaluate each step of the system, setup appropriate ground truth datasets, and compare the performance over existing baseline approaches. In summary:

- we introduce and study the problem of *joint analysis* of 2D image and 3D model collections, while factoring out significant shape and camera pose variations; and

- we propose a framework for multi-modal data analysis across collections for *search and exploration*, without explicitly solving for point- or patch-level correspondences, requiring background detection, or assuming manual filtering.

3.2 Overview

Our key hypothesis is that by jointly considering the inherent qualities of 2D and 3D collections, we can harness the power of one of the domains to improve performance of tasks in the other. However, translating this intuition into a practical framework is challenging as the connections between 2D and 3D repositories are not trivial to discover and exploit, due to the fundamental differences in the two representations. More importantly, natural images (i.e., photographs of real objects) usually have very different appearance compared to counterpart user-generated 3D models. Both the presence of the background clutter in the natural images and the variation of geometry and texture in the 3D models often lead to significant differences in the resulting representations (see Figure 3.2).

Figure 3.2: Using a keyword search, 'airplane' in this example, we retrieve the default ordered images and 3D models from Bing and the Trimble 3D Warehouse. Note the poor quality on the bottom.

CROSSLINK (see Figure 3.1) takes as input a class-labelled 2D image collection $\boldsymbol{I}$, and a class-labelled 3D model repository $\boldsymbol{M}$. We represent each model in $\boldsymbol{M}$ by a set of renderings $\boldsymbol{V}$ taken from a fixed set of viewing angles. We retrieve the class-labelled collections by respectively querying the Bing and the Trimble 3D Warehouse repositories with text queries (e.g., 'car,' 'airplane,' etc.).

Images to improve 3D model search. We observe that image collections often have more accurate labels than their 3D counterparts. For example, a keyword search of 'car' on a 2D search engine yields nearly only true positives among the top hits, whereas the same search on a 3D search engine yields more questionable results (see Figure 3.2). This is partly explained by the fact that online image repos-

itories are orders of magnitude larger than 3D shape collections, which enables training significantly more accurate image labelling and classification mechanisms. We exploit this difference in quality by training a classifier using the data in $\boldsymbol{I}$, by considering them to be 'ground truth positives,' using image descriptors to first convert the images to canonical feature vectors (Section 3.3.2). We then re-sort the models in $\boldsymbol{M}$ using the scores of the classifier on the renderings in $\boldsymbol{V}$ and discard models based on the classification scores. This leaves us with the filtered set of 3D models $\boldsymbol{M}_f$ (Section 3.4). These 3D models, however, are not consistently oriented. We develop a novel image-based approach to co-align the filtered set of 3D models $\boldsymbol{M}_f$ (Section 3.4.1) by exploiting the circular structure in the view space. Thus, an image collection helps to re-sort and co-align a corresponding model collection.

Re-sorted models to reorder images. We then use the clean set of co-aligned 3D models for a given object class to better organize and annotate the input set of 2D images. Our motivation is that whereas 3D models can be viewed from any angle, 2D images are fixed, and extracting the viewpoint (i.e., camera pose) from a given 2D image is non-trivial. We therefore propose an approach for viewpoint estimation that exploits the set of 3D models. We train view-specific classifiers, this time with renderings of one viewpoint from each model in $\boldsymbol{M}_f$ as positives, and the other viewpoint renderings as negatives. As an interesting technical novelty, we show how a one-parameter *family* of classifiers can be obtained via fitting, thus alleviating the need to train many independent classifiers for each view independently (Section 3.5.2). Given a target viewpoint, we then re-sort the image collection $\boldsymbol{I}$ based on the classifier score. Further, we assign to each image in $\boldsymbol{I}$ its most likely view.

Having factored out view variation, we turn to shape variation. While both the 3D models and the natural images exhibit shape variation, we note that certain geometric shape attributes are trivial to extract from the re-sorted 3D models, while the same task is difficult in 2D images. Hence, we train a nonlinear regressor by using the models $\boldsymbol{M}_f$, regularized using a novel formulation to exploit the circular view structure, for each shape attribute and each viewpoint. We then use the regressors to estimate the geometric properties for images in $\boldsymbol{I}$, while using the view informa-

tion extracted in the previous stage (Section 3.5.3). Thus, the 3D model collection helps to re-organize the counterpart image collections according to view and shape variations. The extracted cross links are then used for jointly exploring the image and model collections (Section 3.6).

3.3 Input and Data Representation

The goal of our framework is to reason about 2D and 3D data concurrently. For the 2D part, we take as input a collection $\boldsymbol{I}$ of natural images, which consists of sets of images $\boldsymbol{I}_c$, obtained by issuing a text query c (e.g., 'car,' 'chair,' 'bicycle,' etc.) to a standard 2D image search engine (Bing Image Search). Similarly, we take a 3D repository $\boldsymbol{M}$ consisting of sets of models $\boldsymbol{M}_c$, obtained by issuing the same text query c to a 3D model search engine (Trimble 3D Warehouse). One of our goals is to use the set $\boldsymbol{I}_c$ to filter out the incorrect 3D models from $\boldsymbol{M}_c$. For this, as well as for the other steps in our pipeline, we first bring both sets into a common representation, as described next.

3.3.1 Rendering of the models

To obtain a common representation for both 2D and 3D data, we summarize each model $M_c^i \in \boldsymbol{M}_c$ via a set of 2D renderings $\boldsymbol{V}_c^i$. These renderings are made from a set of N_v viewing angles, at a fixed elevation in a ring around the object. In our experiments, the renderings were taken directly from the 3D Warehouse search results, which provide $N_v = 36$ views per 3D model. Figure 3.3 shows an example. Note that we assume that the up-vector for all models is consistent, and is similar to the standard up-vector in 2D images. While this may not be the case for all types of objects, we have found this assumption to hold for most categories that we considered (except 'guitars' and 'helicopters'). Let us stress that we represent each 3D model as *a structured collection* of images. Throughout the pipeline we will heavily exploit the circular ordering (i.e., views are wrapped around 360°) and the consistent up-direction of the renderings of each 3D model.

Figure 3.3: Each 3D model retrieved from the Trimble 3D Warehouse is used to create 36 renderings in different views, by sampling 10 degree rotations at a fixed elevation (only a few shown in this example). We assume that every input model is upright oriented.

3.3.2 Feature extraction

The 2D repository consists mostly of photographs, which often have background clutter, whereas the 3D models are rendered against no background. In addition, although some of the models have textures, they are not nearly of photographic quality. Finally, the geometry of the models in the 3D repository is often of poor quality, further increasing the difference in appearance. In order to gain resilience to a large class of transformations, we employ a feature encoding approach. We use two different feature representations, both selected for their individual strengths.

KC-encoded HOG features. The first type of feature is a combination of local histogram of gradients (HOG) [25] features, and a Kernel Codebook (KC) encoding [118] to combine them globally. The success of HOG in many recent object detection and classification approaches (cf., [33]) and its sensitivity to changes in orientation makes this descriptor especially interesting in our application. For each 8×8 patch, the HOG features yield a 36-dimensional feature vector, which captures local image gradients (see Figure 3.4).

We combine the local HOG features into global descriptors using the aforementioned kernel codebook encoding. Specifically, we first perform K-means clustering on all local HOG features of all images in both $\boldsymbol{I}_c$ and $\boldsymbol{V}_c$, resulting in a codebook $\boldsymbol{\mu}_c$ of K visual words for each class c ($K = 800$ in our tests). Then, for a given image $\boldsymbol{x}$ with HOG features $H(\boldsymbol{x})$, each local HOG feature $h \in H(\boldsymbol{x})$ is encoded as a K-dimensional vector, which has non-zero elements only for the

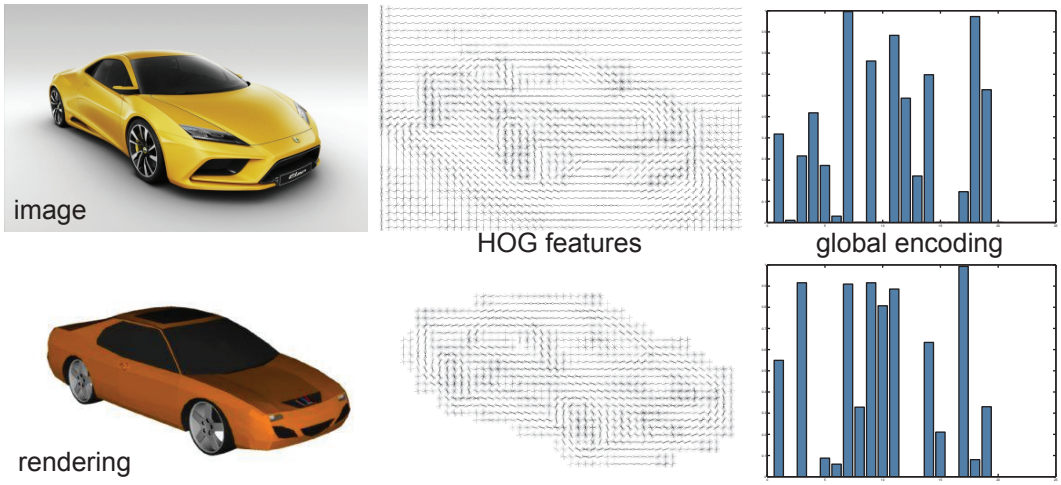


Figure 3.4: Left-to-right: Input images, corresponding HOG features, global feature vector encoding using max pooling (K=20 for visualization). Top row: car image from Bing; bottom row: a comparable camera view rendering from Trimble 3D Warehouse.

closest k ($k = 4$) neighbors of h in $\boldsymbol{\mu}_c$:

$$[KC_c(h)]_i = \begin{cases} \frac{d(h,\mu_c^i)}{\sum_{j \in \mathcal{N}_k} d(h,\mu_c^j)} & \text{if } i \in \mathcal{N}_k(h, \boldsymbol{\mu}_c) \\ 0 & \text{otherwise} \end{cases}$$

where, $\mathcal{N}_k(h, \boldsymbol{\mu}_c)$ are the indices of the k closest neighbors of h in $\boldsymbol{\mu}_c$, and $d(h, \mu)$ is a distance kernel, defined as

$$d(h, \mu) = \exp(-\gamma ||h - \mu||^2).$$

We used $\gamma = 100$ in our experiments. A major advantage of this approach is that the global feature vector will vary more smoothly with small changes in the local feature vectors. This is important when modelling the view classifiers as a smoothly varying family, as discussed in Section 3.5.2. Finally, we combine these local encodings into a global encoding using max pooling [62, 15]

$$[KC_c(H(\boldsymbol{x}))]_i = \max_h [KC_c(h)]_i, \quad h \in H(\boldsymbol{x}).$$

This setup results in one 800-dimensional Euclidean feature vector per image.

Please note the variation from the original approach [118], where the local encodings are summed together. Our reasoning here is that the 2D images in $\boldsymbol{I}$ often have background. Using sum pooling, the gradients of the background would always be taken into account, which can adversely affect later image comparison. Our experiments confirm this boost in performance.

CNN features. The second type of feature is extracted using a pre-trained convolutional neural network (CNN) [67, 63]. This network was trained on ImageNet [28] – a dataset of over 10 million images in over 10000 subcategories – and is the state-of-the-art for classification on this particular dataset. We use the final 4096-dimensional fully connected layer from this network as a global image feature. One CNN feature vector represents one image, and is of dimension 4096. Note that this feature descriptor is expected *not* to discriminate well between views, as it was specifically trained to recognize objects in any configuration. We show and evaluate the difference in performance between the two features in Section 3.6.

For each object class c, we compute HOG and CNN features of all images in $\boldsymbol{I}_c$ and all rendered views in $\boldsymbol{V}_c$. For any image $\boldsymbol{x}$, we will refer to its KC-encoded HOG features as $\mathrm{HOG}(\boldsymbol{x})$, and to its CNN features as $\mathrm{CNN}(\boldsymbol{x})$. As most of the pipeline is agnostic to the type of feature used, we refer to any feature generically as $\mathrm{F}(\boldsymbol{x})$.

3.4 3D Model Collection Filtering

Our first observation is that label accuracy of the images in $\boldsymbol{I}_c$ is significantly higher than of the models in $\boldsymbol{M}_c$. Therefore, we capture the characteristic features of the given class in a classifier by using the high accuracy of the images in $\boldsymbol{I}_c$, and then re-sort the models in $\boldsymbol{M}_c$ using this classifier.

By using one of the feature representations $\mathrm{F}(\boldsymbol{x})$ mentioned above, we first create a set of 'positives' from the feature encodings of the in-class images in $\boldsymbol{I}_c$, i.e.,

$$P_c^{\text{filt}} := \mathrm{F}(\boldsymbol{I}_c).$$

Similarly, as negatives, we use the feature encodings of the out-class images, as well

as the feature encodings of a sampling of renderings from the out-class models:

$$N_c^{\text{filt}} := \mathrm{F}(\boldsymbol{I}_{C\setminus c}) \cup \mathrm{F}(V_{C\setminus c}^{i,j})$$

where, $V_c^{i,j}$ is a sampling of the j-th rendered view of 3D model i. In our experiments, we take 10 models per class, and 2 views per model, randomly sampled from the set. Empirically, the addition of these out-of-class negatives significantly improved the results. Intuitively, the features in the negatives from rendered 3D models are more likely to appear in the test data, and therefore, force the classifier to find a better trade-off between the positive features from the natural images and negative features from both images and rendered 3D models.

We use the sets P_c^{filt} and N_c^{filt} to train a standard linear support vector machine (SVM) [24] f_c^{filt} that finds the separating hyperplane with the largest margin between the positives and negatives. This will allow us to give a confidence score that a particular 3D model in $\boldsymbol{M}_c$ actually belongs to class c, by considering the scores of its rendered views on f_c^{filt}.

We apply the trained support vector machine to all views in $\boldsymbol{V}_c$, resulting in a per-view score $f_c^{\text{filt}}(V_c^{i,j})$. For a 3D model, we define the score as the maximum score over all its views:

$$f_c^{\text{filt}}(M_c^i) = \max_j f_c^{\text{filt}}(V_c^{i,j}).$$

An alternative would be to use the average instead of maximum. However, we have found that the distribution of different views in $\boldsymbol{I}_c$ is most often not well-balanced, and thus we cannot expect views in $\boldsymbol{V}_c$ that are not well represented in $\boldsymbol{I}_c$ to have a high score, even when they are views of an in-class model (see Figure 3.5).

Finally, we sort the models in $\boldsymbol{M}_c$ based on their scores to obtain an ordering on the 3D models based on the confidence of them belonging to class c. In Section 3.6, we show that this ordering significantly outperforms the default ordering given by Trimble 3D Warehouse or obtained by a direct shape descriptor-based clustering. Finally, the filtered model set is obtained by removing all models with a negative

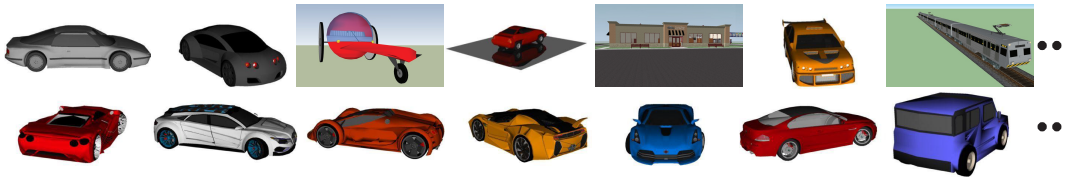


Figure 3.5: We re-sort the (top) original ordering of 'cars' from the 3D model collection to obtain a new ordering (bottom), automatically pushing the false positives to the end of the list.

classifier score.

3.4.1 3D model alignment

Having filtered the 3D models for each class c, next we organize them in a way that makes joint shape analysis and exploration simple and effective. First, we co-align the shapes into a shared canonical position (see Figure 3.6). To achieve this, we developed a novel method for joint alignment, which is sensitive to object appearance. Our approach takes as input a set of 3D models that mostly belong to a single category c, with a few potential mis-classified instances, gathered using the method described in the previous section. Our goal then is to find the rotation for each 3D model such that a properly defined global alignment error is minimized. The proposed technique particularly exploits the circular nature of the view space, and hence is both efficient and robust in the presence of mis-classified instances.

In this part we use the same hybrid 2D-3D shape representation as described above. Namely, we represent each 3D model as a collection of 36 images rendered at 10 degree increments of azimuth from a fixed elevation. Recall that this representation assumes that all 3D models have a consistent "up" direction. We summarize each of the rendered views compactly using a Euclidean vector of size K with the chosen feature encoding F. This means that each 3D model is represented as a matrix of size $K \times 36$, where the rows and columns stand for feature encodings and views (camera poses), respectively.

Given a set of matrices $\{M_1, M_2, \ldots M_N\}$ of size $K \times 36$ corresponding to N different shapes, our goal is to find a vector V of size N, where each V_i is an integer

Figure 3.6: The input models do *not* come co-aligned (top). We propose a simple effective method to consistently align them (bottom).

in the range $[0 \dots 35]$, such that the following error is minimized:

$$E(V) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=0}^{35} \left\| M_i^{V_i+k \bmod 36} - M_j^{V_j+k \bmod 36} \right\|_2^2 . \tag{3.1}$$

Here we let M_i^l be the lth column of matrix M_i, and the norm corresponds to the L_2 norm between the corresponding column vectors. Intuitively, the alignment problem amounts to finding the offset V_i for each shape i, such that the feature representation of the $k + V_i$ view of 3D model i corresponds to the representation of $k + V_j$ view of 3D model j for each j and each k, modulo 36. Note that $E(V) = E(G)$ if G is any vector such that $G_i = V_i + c \mod 36$ for any constant c for all i.

We optimize Equation (3.1) using a simple iterative technique, similar to Iterated Conditional Modes inference of Markov Random Fields, and to congealing [68] with a discrete search space. In particular, we start by considering a random vector V. Then, for each $i \in [1 \dots N]$, we find the minimizer of $E(V)$ with all but i^{th} dimension of V fixed, and update V_i, if necessary. Note that since $V_i \in [0 \dots 35]$ the optimum can be found by direct inspection. We repeat this procedure, iterating over the different dimensions of V (corresponding to different shapes), until convergence. We then restart this procedure for several (200 in our experiments) initial random vectors V and keep the solution V^{opt} which minimizes the error $E(V)$. See Figure 3.7 for a visualization of this procedure. In practice, we have noticed that keeping one dimension of V fixed during optimization helps to speed up convergence by reducing the presence of multiple global optima due to the circular nature of the energy function E.

The final vector V^{opt} provides a set of views, one per shape, that are as consis-

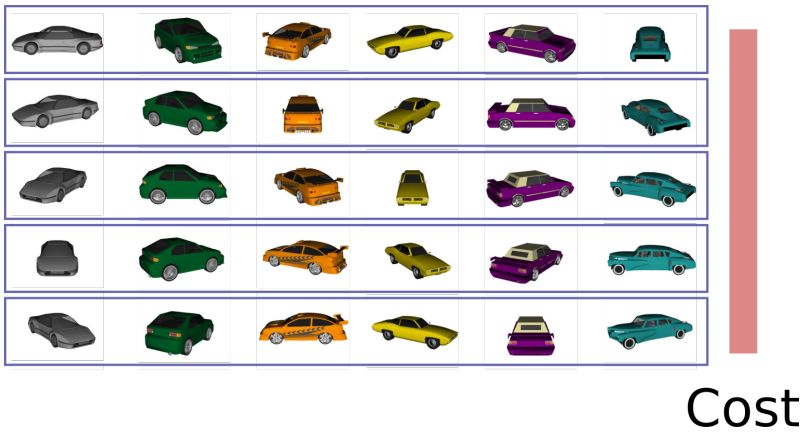


Figure 3.7: Visualization of the alignment algorithm. Exploiting the cyclic nature of each model's collection of renders, we find the offset that minimizes the feature description distance between each render within the same row. After a few iterations a stationary point is reached. By running this algorithm with many different initializations, we avoid local minima as much as possible. The result is an accurate and efficient alignment of the models.

tent as possible (see Figures 3.6 and 3.12). Note that if we expect V_i^{opt} and V_j^{opt} to correspond for shapes i and j, then views $V_i^{\text{opt}} + k \mod 36$ and $V_j^{\text{opt}} + k \mod 36$ should correspond for any k as well. This simple method is remarkably efficient and works well resulting in an average error of only 5-10° (see Section 3.6).

3.5 Image Collection Organization

The output of the method described in the previous section is a set of co-aligned and filtered 3D models with a small number of false positives for each class c. We then use these models to better organize the corresponding image collections. Namely, we use the co-aligned 3D models to estimate both the camera pose and certain geometric properties of objects in the 2D image set $\boldsymbol{I}_c$, which then enables queries such as 'show images of a tall chair from a particular viewpoint'.

Note that we can render the co-aligned 3D models from specific viewing angles for each model in $\boldsymbol{M}_f$. Specifically, we put together sets of images (renderings) taken from the same view, and use them as positives in a linear classifier, as described below.

3.5.1 Camera pose estimation

Suppose we are given a class c and a viewing angle θ. To compute a classifier corresponding to this view, we first construct a set of positive examples by gathering all views of the 3D models in $\boldsymbol{V}_c$ corresponding to θ:

$$P_{\text{view}} := \mathrm{F}(\boldsymbol{V}_c^{\theta}).$$

As negatives, we use the *other* views from the same class:

$$N_{\text{view}} := \mathrm{F}(\boldsymbol{V}_c \setminus \boldsymbol{V}_c^{\theta}).$$

As neighbouring views are often very similar, including them in the negatives decreases the score for positives as well. However we are interested only in the relative scores of the classifiers to decide which view a given 2D image should be assigned to. This overall decrease in classifier score is thus not an issue for our case.

Having trained a linear SVM with these positives and negatives, we run the obtained classifier f_c^{view} on the images in $\boldsymbol{I}_c$. The resulting classifier scores correspond to the confidence of each 2D image being associated with the viewing angle θ. Note that the approach here is similar to the 3D repository filtering from Section 3.4 – although the modality from which the training data and the test data originate are now switched, the overall classification approach we employ is the same.

We generate one classifier f_c^{view} per viewing angle, yielding N_v different classifiers (36 in our case). Given this set of classifiers, we use them to re-sort the 2D images according to their scores $f_{c,v}^{\text{view}}$, which puts the images in $\boldsymbol{I}_c$ taken from the view v at the top.

In some applications we may also need assign a single view to each image. However, we cannot directly compare classifier scores, as SVM classifier scores are not calibrated, i.e., similar scores in $f_{c,i}^{\text{view}}$ and $f_{c,j}^{\text{view}}$ $(i \neq j)$ does not imply similar confidence of the image belonging to view i or j. Thus, we use Platt scaling [90] to convert the SVM score of a test image to a probability.

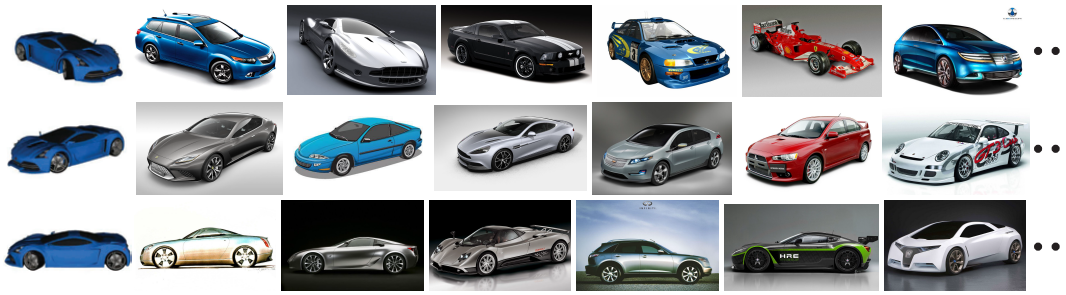


Figure 3.8: Camera pose estimation. Here, we show the top results for three different views, one per row, for the 'car' dataset. The icons on the far left indicate the corresponding 3D model view.

Instead of directly taking the view with the maximum probability, we once again exploit the circular structure of the data. Specifically, given an image and its ground truth view, we expect the score of the classifier pertaining to that specific view to be high, but as neighboring views are similar in appearance, we furthermore expect the neighboring classifiers to score above average as well. For the feature vector of a given image $\boldsymbol{x}$, we take this into account by computing a score for each view v as the weighted sum of the classifier score $f_{c,v}^{\text{view}}(\boldsymbol{x})$ and its neighboring

views $f^{\text{view}}_{c,v\pm r_v}(\boldsymbol{x})$, with weights chosen as a Gaussian distribution g with zero mean and unit variance:

$$f^{\text{viewWeighted}}_{c,v}(\boldsymbol{x}) = \sum_{i=-r_v}^{r_v} g(i) f^{\text{view}}_{c,v+i}(\boldsymbol{x}).$$

We then assign each image to the view which has the highest weighted score. We have observed this weighting across neighboring views to improve performance (Figure 3.8).

Note on regression and non-linearity. Regression seems to be the natural choice for learning the relationship between the viewing angle and the feature vector of an image. However, this approach is made difficult by the inherent non-linearity of the relationship. As the Euclidean distance between feature vectors of two neighboring views is expected to be small, the feature vectors of the 36 views of a given model lie on a loop in feature space. We tried support vector regression with a number of different kernels (radial basis function, hypertangent and polynomial) with a range of different parameters, but the performance was very low in all cases (see Section 3.6). Although writing a kernel capable of dealing with this specific type of non-linearity may be possible, our approach is particularly appealing due to its simplicity and good performance in practice.

3.5.2 Modeling of classifier weights

Training one classifier for each of the 36 different views is both costly in practice, and moreover, does not allow classification corresponding to views outside of this fixed set. Intuitively, the weights of the classifiers f^{view}_c will have a regular structure: we expect the weights w^i_c of $f^{\text{view}}_{c,i}$ to be quite similar to the weights of $f^{\text{view}}_{c,i+1}$. We illustrate this by performing a Principal Component Analysis (PCA) on the classifier weights of the 36 classifiers for the class 'car.' Examining the resulting PCA dimensions we note that the first three explain over 80% of the variance in the classifier weights (Figure 3.9, left). Plotting the coefficients in these dimensions (Figure 3.9, right) of the computed SVM weights f^{view}_c against the viewing angle shows very smooth and regular structure. This regularity is also present for the classifier biases b_c. We exploit this structure to setup a model of the classifier weights and bias,

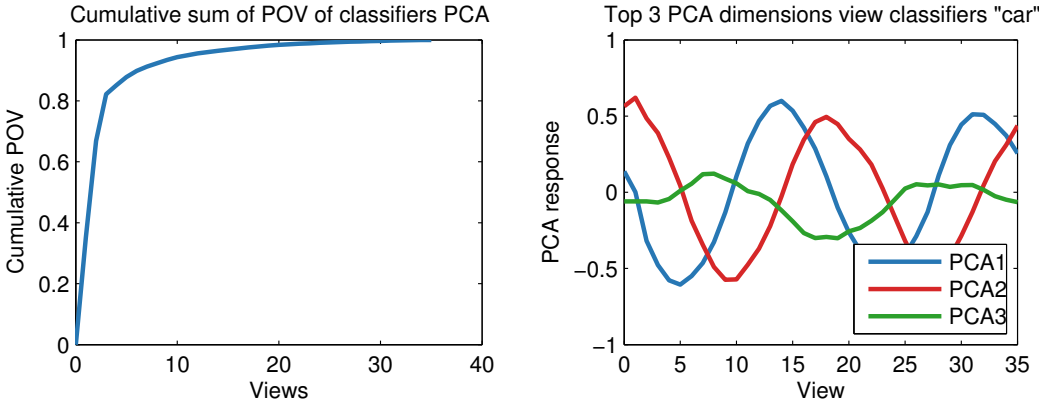


Figure 3.9: Left: Cumulative fraction of variance of the PCA dimensions for the view classifier weights of class 'car.' Note that 80% of variance is explained by just 3 PCA dimensions. Right: The top 3 PCA dimensions show regular structure, suggesting that they can be modeled.

allowing us to create a one-parameter *family* of classifiers per object class, parameterized by the viewing angle, *without* explicitly training an independent classifier for each angle.

Specifically, we sample N_{sample} of 36 viewing angles, evenly across the circle (every $360/N_{\text{sample}}$ degrees), and perform PCA on the weight vectors of the resulting classifiers (see Section 3.5.1). For each of the top 3 PCA dimensions $[p_c^i]_{i=1}^3$ and for the bias vector b_c we find interpolating cubic splines $[q_i]_{i=1}^3, q_b$. Then, to compute a classifier for a given view θ, we evaluate the interpolators for θ and reproject to the original space:

$$w_c^\theta = \sum_{i=0}^{3} p_c^i \, q_i(\theta) \quad \text{and} \quad b_c^\theta = q_b(\theta). \tag{3.2}$$

This yields a classifier $f_{c,\theta}^{\text{viewModeled}}$ that provides a confidence score for a given example of class c as to whether it belongs to view θ. Note that the higher we set N_{sample}, the less data we take into account, and thus the lower the actual training cost. Section 3.6 describes the effect of varying N_{sample}.

3.5.3 2D repository re-sorting by shape

At this stage, we have a one-parameter family of classifiers that we use to re-sort the 2D images in $\boldsymbol{I}_c$ for any viewing angle. Having now tackled view variation, we

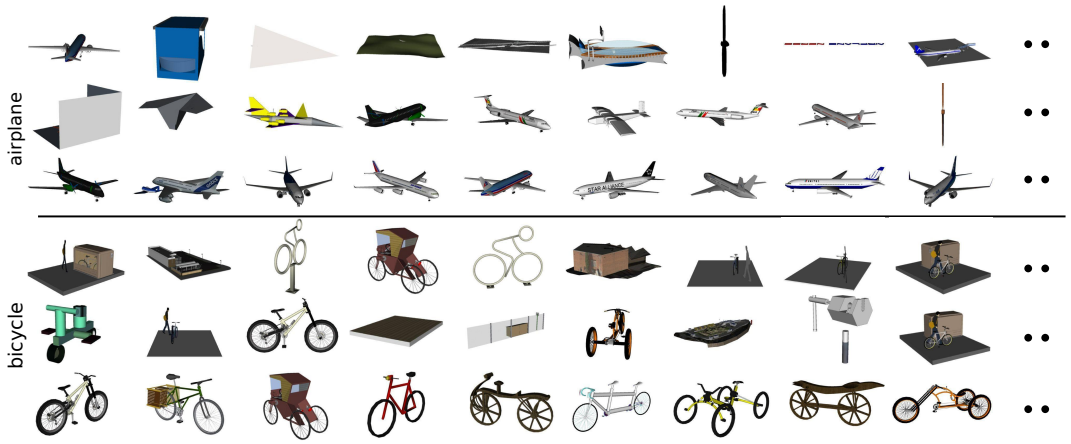


Figure 3.10: Re-sorting of model collections for 'airplanes' and 'bicycles.' In each case, top shows original ordering from the 3D Trimble Warehouse, middle shows ordering using Lightfield clustering, while bottom shows CROSSLINK results. See Figure 3.11 for performance evaluation against manually annotated ground truth. Note that the results are not co-aligned at this stage.

focus on in-class shape attribute variation. Capturing such variations in the image domain allows not only sorting the images by camera pose (view), but also to sort the images within each view by object structure.

To achieve this, we extract, from the filtered models $\boldsymbol{M}_f$ in class c a certain scalar geometric shape attribute $X \in \mathbb{R}$, which is chosen to be trivial to extract in the 3D domain, but challenging in the 2D domain. In our experiments, X is the ratio of the height over the width of the model (we consider the ratio to account for image/model scale variations). Concatenating these X_i for each model in $\boldsymbol{M}_c$ results in a vector $\boldsymbol{X}_c$.

Our goal is to arrive at an estimator that links the property X using the features of the images in $\boldsymbol{I}_c$. We do so by training kernel ridge regressors (KRR) [98] on the feature encodings F of the views in $\boldsymbol{V}_c$. To factor out view variation, we train one regressor r_c^θ per viewing angle θ.

More so than in the previous parts of the pipeline, the stark difference in feature distribution between photographs and renderings significantly limits the regressor's performance on the 2D images when trained on the 3D renderings. To offset this difference between the domains, we use a geodesic flow kernel [39] in the KRR. This kernel constructs an implicit feature domain assembling information from the

source and target domains (photographs and renderings) and an infinite number of domains interpolating between the two. Our results show a significant performance increase when using such a domain-adaptation technique.

To estimate the value of the attribute $\boldsymbol{X}$ in a given image $I_c^i \in \boldsymbol{I}_c$, we use the regressor corresponding to the assigned view of I_c^i, which we extracted in the previous section, and apply it on the feature encoding of I_c^i:

$$x(I_c^i) = r_c^\theta(\mathrm{F}(I_c^i)).$$

The output of this procedure is a set of estimators of the given geometric attribute, one per each viewing angle. We use these estimators to sort the natural images according to the shape of the objects in them.

3.6 Evaluation

We extensively evaluated the proposed framework on various real world examples. First, we shortly discuss the data sources on which we performed our experiments, including their origins, size, and associated ground truth. Then, we show the results of applying each part of the system on the data, and both discuss and show how we evaluate the various results.

Data collections. Below, we present results for 4 (of 20) classes, namely *airplane*, *bicycle*, *car* and *chair*. For each of these classes, we scraped the top 150 results from a keyword search from Bing and Trimble 3D Warehouse for 2D and 3D data, respectively. As mentioned in Section 3.3.1, for each model 3D Warehouse provides 36 views, rendered from evenly spaced angles in a circle around the model from a fixed elevation. Results for another 16 classes can be found in Appendix A.

Ground truth. We manually annotated all 3D models as being either a good example of their class (true positive), or a bad example (false positive). In some cases, a model file contained either more than one instance of the class, or contained multiple other objects as well. In those cases, we only counted the model as positive if the model was prominent in the renderings. In the 2D repository, each image was

annotated with its ground truth viewing angle (discretized to 10 degree bins to correspond with the renderings of the models). Note that this assumes camera poses level with the ground (fixed up-vector) and a view taken from a similar elevation as the renderings. This assumption breaks down for some classes, such as airplane and helicopter.

3.6.1 3D repository filtering using 2D

In the first experiment, we filter the 3D models using the 2D images, based on the feature encodings of the 2D images and the renderings of the models (see Section 3.4). Note that the only user supervision in the whole process is the choice of classes, via a choice of the text query issued to the 2D and 3D search engines (Bing Image Search and Trimble 3D Warehouse search), such as 'car,' 'chair,' 'bicycle,' etc. We tested the setup with both KC-encoded HOG features, as well as with the pretrained CNN features.

Lightfield baseline. Intuitively, models in the same class should be geometrically similar, and are thus expected to be clustered in any standard descriptive feature space. We extracted 3D shape descriptors from all models and then performed unsupervised hierarchical clustering on the feature vectors. The optimal distance criterion for the clustering was found by trial and error on a number of classes and then fixed. Next, we ordered the clusters by size (largest first, smallest last), and

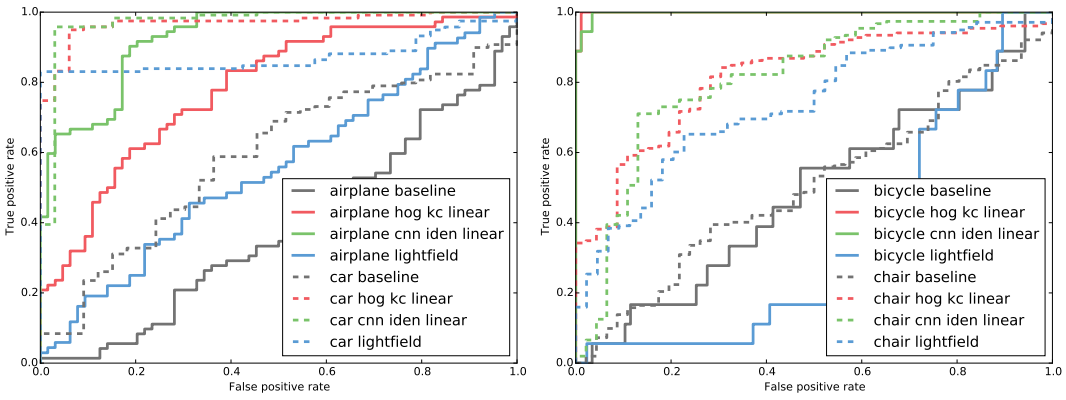


Figure 3.11: ROC curves for 4 different classes measuring effect of 3D repository filtering using 2D image information as compared to manually annotated ground truth quality for 3D models. As baselines, we present both the original orderings in the model collections and the ordering based on the Lightfield descriptor.

within each cluster sorted the models by distance to the cluster centroid (closest first, farthest last). We use this as a baseline to compare our method. We employed a Lightfield descriptor [18], which was found to be the most discriminative shape descriptor available [100].

Qualitative evaluation. Figure 3.10 shows the results of two keyword searches according to the original ordering in which they were returned by 3D Warehouse, the ordering garnered from the unsupervised clustering of the Lightfield descriptor as mentioned above, as well as the ordering by our algorithm. Note that the original top results returned by 3D Warehouse contain a high number of false positives, even though many true positives do appear later in the set of results. Although the Lightfield clustering does improve results, the presence of many false positives throws it off balance. After reordering using our algorithm, most of the false positives in the top results for both classes have disappeared, having been assigned low scores by the classifier.

Quantitative evaluation. In Figure 3.11, we show the ROC curves for 4 different classes, for the original ordering from 3D warehouse, the one using the Lightfield descriptor, and our ordering for both feature setups. For all classes, the performance is significantly better than the original ordering. For classes with little shape variation and many true positives, such as 'car', the Lightfield descriptor clustering works well. However for classes with a significant number of false positives, such as 'bicycle' and 'airplane', our method is noticeably better than the Lightfield clustering baseline.

CNN vs. HOG. The CNN based feature outperforms HOG on nearly all classes. This is especially apparent for classes where there is very consistent background in the photographs, airplane and boat (see Appendix A). We believe that the HOG based feature associates the class with the background, which is missing from the renderings. In contrast, CNN based feature does not have this problem.

3.6.2 3D model alignment

Having filtered the datasets, we now test the 3D alignment method in image space, as proposed in Section 3.4.1. We apply our method to the model set of each class. The resulting alignment for one such class is shown in Figure 3.12. Note that although the original set of models is quite random in its co-alignment, our simple image-based algorithm finds the correct alignment for most models.

Pairwise consistency. The mean pairwise angular offset (MPAO) of all models before and after alignment is shown in Table 3.1. This error metric is the expected difference in alignment between two randomly picked models from the set. The 'airplane' class still has a high MPAO after alignment, which is mostly due to the algorithm not being able to distinguish well between flipped versions of the same model, due to their feature encoding similarity. Changing the feature representation could improve performance in this case, and exploring this avenue is left for future work.

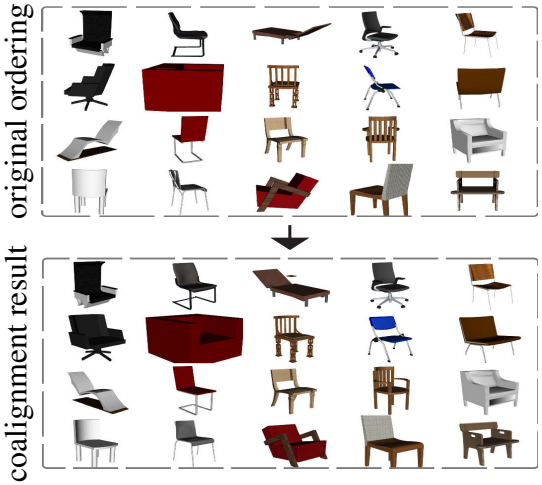


Figure 3.12: Co-alignment of 3D models. (Top) Initial alignment across the 3D models, (bottom) consistently co-aligned 3D models for the 'chair' models. Please refer to Table 3.1 for detailed error analysis against hand annotated ground truth data.

Comparison with mesh-based method. For comparison, we ran the filtered model set for two of the classes (laptop and car) through an existing mesh-based alignment method, as described in [50, 8]. Although the accuracy of this method is slightly higher than ours, the method takes significantly longer to run. Note that for both our

Figure 3.13: View-classifying Bing images using classifiers trained using 3D models. We assign a best view-estimate for each Bing image. Here, we show a set of example view estimations for 'car' images. The corresponding ROC curves are computed again ground truth data compiled by manually annotating each of the retrieved Bing images (only a sampling shown here).

method and the mesh-based method, we started with the filtered sets of 3D models. The performance decreases significantly for both methods when using the unfiltered model collections.

Effect of filtering. We show the importance of filtering the 3D models before applying the alignment step using the class 'bicycle.' This class contains very few true positives. As such, without prefiltering, the dataset is very noisy, making it difficult to find a consistent co-alignment across models. After prefiltering the performance is increased by an order of magnitude. The same can be observed for small data sizes in general, as shown for class 'helicopter.' When only taking 50 models, re-aligning them in the unfiltered state yields an error 50% higher than when filtering first.

Table 3.1: Accuracy of 3D alignment. Each value represents the mean pairwise angular offset between models in that specific scenario. For the 'bicycle' class, there are not many models left after filtering, resulting in an underconstrained optimization. This is reflected by the lower number of perfect alignments. 'Airplane' models often have a $180°$ error, resulting in a relatively high MPAO.

Class	Before	After	Perfect
airplane	$82.5°$	$41°$	64%
bicycle	$91.2°$	$7.72°$	23%
bicycle, no filtering	$91.2°$	$83.32°$	23%
car	$90.3°$	$5.79°$	95%
chair	$87.4°$	$0.9°$	95%
helicopter	$93.1°$	$17.4°$	95%
helicopter, no filt.	$93.1°$	$19.2°$	95%
helicopter, no filt. (50 models)	$93.1°$	$30.2°$	95%

3.6.3 2D repository view sorting

After filtering and alignment of the 3D repository, we reorder the 2D images from Bing according to view variation (see Section 3.5). We solely use KC-encoded HOG-based feature for this part, as the CNN-based feature was specifically trained not to respond to changes in viewing angle. Figure 3.13 shows, for class 'car,' the assigned views for a number of images, as well as the ROC curves for the ordering based on the view classifier scores of three different views. Note that many of the errors are due to the assignment of images to 180 degree flipped views, which are especially prevalent for the side view of the car, but are also present in other views.

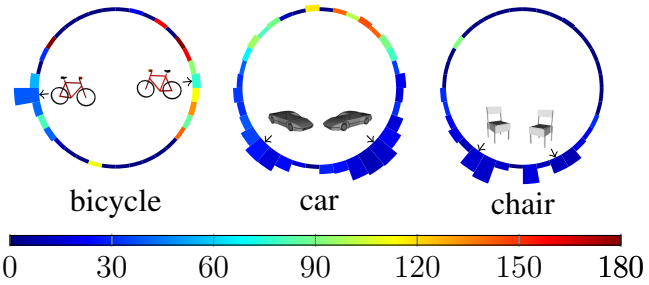


Figure 3.14: R-precision of view classification per view shown for three different classes. For each bar, color signifies the R-precision with respect to ground truth, while height corresponds to the number of Bing images with that view as ground truth. Note that the height gives a measure of confidence to the error (i.e., taller bars indicate more statistically significant), and also shows the distribution of views per class. For all classes, this distribution is very biased towards a number of canonical views (e.g., showroom 3/4th views for cars). Tall blue bars indicate perfect results, while short bars of any color can be ignored due to lack of enough data.

Error per viewing angle. Figure 3.14 visualizes the R-precision (the precision of the classifier at position R, where R is the number of positives available) per viewing angle together with the distribution of viewing angles across the ground truth. Judging only from the ordering of the images themselves, the somewhat high quantitative error for some views seems surprising. We have observed that perceptually two views can look very similar, while still being objectively somewhat further apart. For exploration, this ambiguity works in our favor – even with slightly erroneous view alignment, perceptually the ordering of the images makes sense.

Comparison with regression. In Figure 3.15 we show the distribution of error

magnitude in degrees for two classes, for both our method as well as a comparative regression method. We ran a support vector regressor using 4 different common kernels (linear, polynomial, hypertangent and radial basis function) on the data, using 10-fold cross validation to find the best parameters (both for the kernel and the regularization parameter). The results shown in the figure have highest performance. Note that this approach does not work nearly as well as our classification approach. Finding a kernel capable of handling the circular nature of the data remains an interesting direction for future work.

Effect of background clutter. Figure 3.16 shows representative examples of view classification degradation under background clutter in test images. Often these images exhibit strong directional lines, such as a sharp horizon, or the silhouettes of buildings. Dealing more explicitly with such difficulty, for example by incorporating background in the training stage, a direction which is explored in Chapter 5.

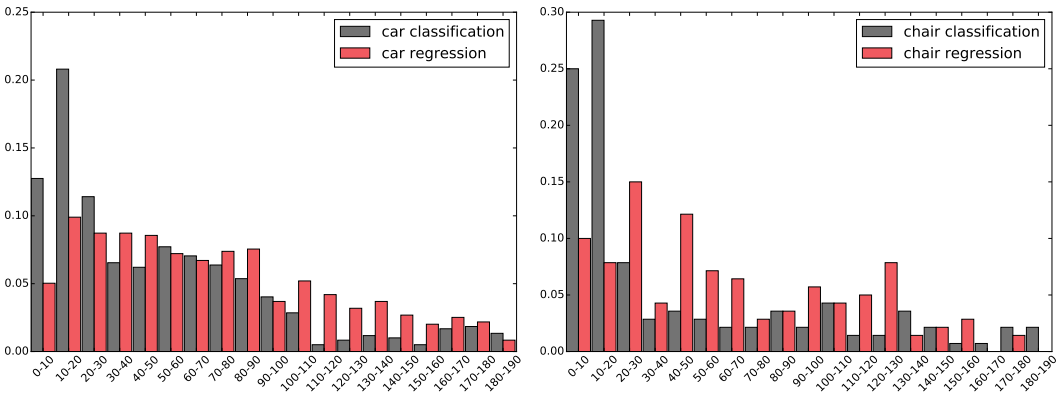


Figure 3.15: Histogram of error magnitudes for the 'car' and the 'chair' class, discretized in bins of 10 degrees, for both classification and non-linear regression. Note that our method results in a large concentration of errors in 0-10 and 10-20 bins.

3.6.4 2D view classifier modeling

To avoid the high cost of training many classifiers, we exploit the regular structure of the classifiers' weight vectors and bias, as described in Section 3.5.2. Figure 3.17 shows for the class 'car' the Kendall-Tau rank correlation between the original view classifiers, trained as normal, and the modeled classifiers. This statistic measures the correlation of two data orderings, being 1 when the orderings are equal, -1

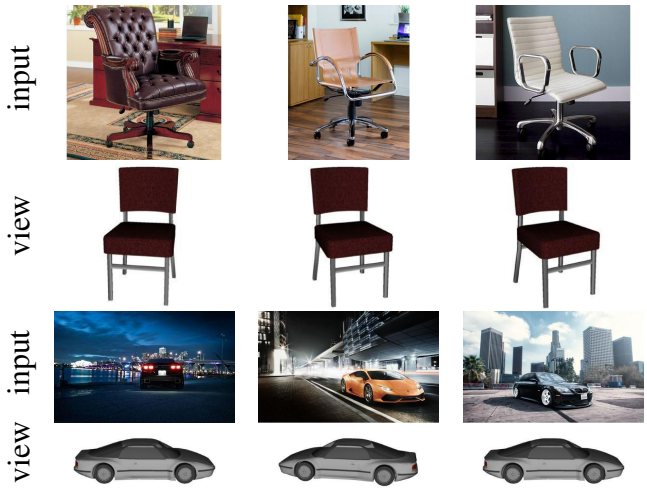


Figure 3.16: View classification of images with significant background clutter. The photographs are input to our view classification pipeline, the renderings are the resulting view classifications. In some cases (bottom row) the background clutter leads to misclassifications, whereas in other cases the system handles the difficulty well.

when they are opposite, and 0 when they are mutually independent. The left-most chart shows that applying PCA to the weight vectors and using only the top 3 dimensions does not change the resultant ordering much, showing that just the top 3 dimensions in feature space are responsible for most of the view classification performance. In the middle chart, we sample only every 30 degrees, reducing the number of classifiers we have to train by a factor of 3. Although the original classification score orderings are not preserved entirely, the performance is still reasonable. Even increasing the step size to 50 degrees does not dwindle performance entirely, although it approaches the limit of what is useful.

Currently, we use cubic interpolation for the estimation of the weight vectors. The curves, as shown in Figure 3.9 and as we observed for other classes as well, have a clear periodic structure, and could possibly be approximated using a sum of sines model. This would possibly constrain the system more, allowing us to use even sparser sampling of the classifier space. We leave this possibility for future work.

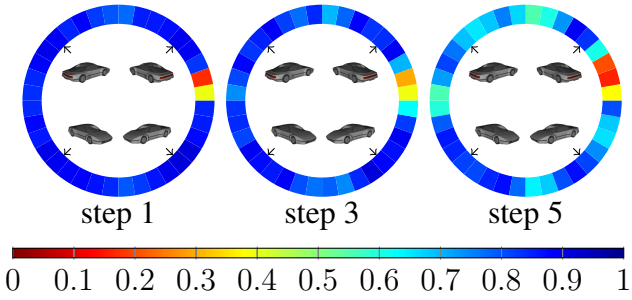


Figure 3.17: Performance of modeled SVM classifiers on synthetic data using 3 different sampling densities. A step size of n means we model the first 3 PCA dimensions using every n^{th}th classifier from the set of normally trained classifiers. The color signifies the Kendall-Tau ranking correlation between the modeled SVM of the corresponding view and the respective normally trained SVM. A score of 1 signifies perfect correlation (equal ordering, so zero loss), a score of 0 means uncorrelated orderings. To show the effect of the PCA we also show step size 1 (taking all classifiers).

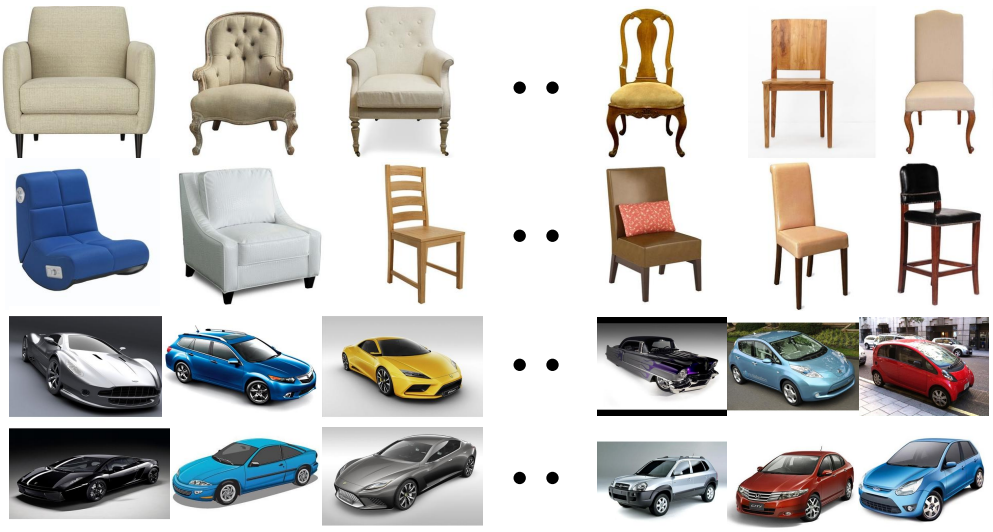


Figure 3.18: Attribute-sorted view-classified Bing images for two classes. Each row shows a different view sorting. For each row, as we go from left-to-right, the height-to-width ratio increases, i.e., the objects turn from short-and-wide to tall-and-narrow.

3.6.5 2D repository shape sorting

We test the method described in Section 3.5.3 for estimating a certain shape property from images in the 2D repository. First, we extract the ratio of height over width from the set of aligned 3D models. Then, for each viewing angle a KRR is trained using the method described in Section 3.5.3. To estimate the ratio for a given image from the 2D repository, we apply the KRR corresponding to the image's assigned

viewing angle to the features of that image. Applying this to all images of each view, we can re-sort the images within a view by ratio. We show a sampling of such results for the class 'chair' in Figure 3.18. Figure 3.19 shows for the 3D renderings the Kendall-Tau rank correlation between the ordering by ground truth ratio and the ordering by estimated ratio for each view. Although the score decreases for views from which the width is difficult to judge, the scores across most other views is high.

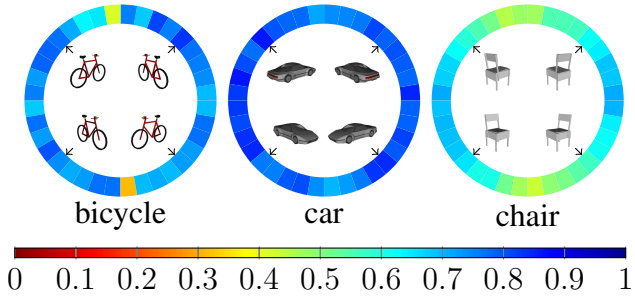


Figure 3.19: Accuracy of shape estimation. Color represents the Kendall-Tau rank correlation coefficient for the ordering of the synthetic models according to the shape ground truth versus the estimated shape. A score of 1 indicates perfect correlation (equal ordering); a score of 0 indicates uncorrelated orderings.

Notes on scalability and performance. So far, we reported evaluations on 20 classes with image and model sets of size 150 per collection. To test scalability, we tested a number of classes on datasets of larger size. (Note that the main bottleneck is to prepare the ground truth for larger sets.) Specifically, we tested the influence of increasing both the size of training data and the number of testing data. The left ROC plot in Figure 3.20 shows that increasing the test set to the top 1000 search results does not decrease performance with respect to the original set of 150 (compare with Figure 3.11). Note that this is not trivial, as we expect the 3D Warehouse search results to increasingly contain more false positives. In contrast, recall that method only relying on 3D data, will perform worse as the fraction of outlier shapes increase. The right plot shows the performance of training with 1000 examples. Note that performance increases only slightly with respect to the original training of only 150 examples.

Our pipeline consists of unoptimized code, with the extraction of the features

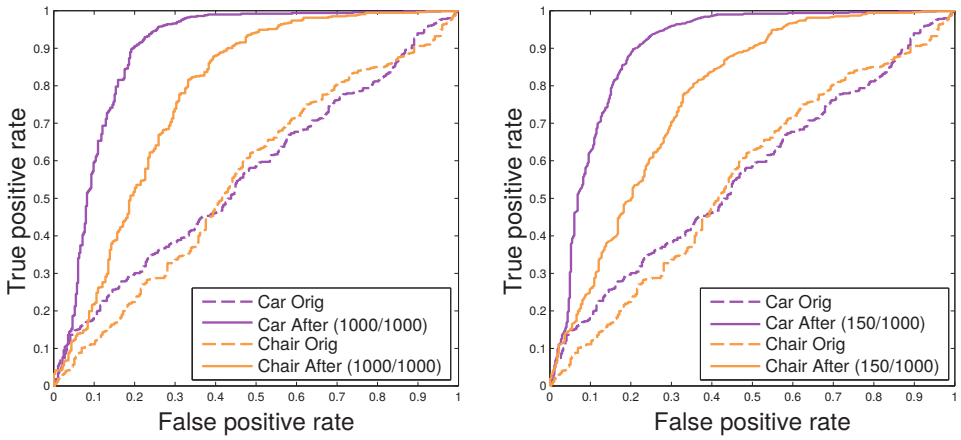


Figure 3.20: Left: ROC curve for HOG-based classifier trained on 1000 2D images with the purpose of filtering 1000 3D models. Right: ROC curve for classifier trained on 150 2D images with the purpose of filtering 1000 3D models. Although results improve slightly, very similar performance is obtained with the smaller training set. This shows that our approach scales to large datasets, as only training the classifiers is expensive.

being the main bottleneck. While timings can be improved in the future, currently in order to process 150 image/model sets, the system takes 1-2 minutes for the image $\rightarrow$ model direction and 3-4 minutes for the image $\leftarrow$ model direction. While linear in complexity with the number of models, the step can be easily be run across multiple threads.

Limitations. We observed two main sources of errors: (i) In case of a class like 'boats,' consistent image background (i.e., water) can easily be learned as a distinguishing feature by view classifier. Although this effect is diminished when using CNN features, it is still an issue. An interesting future direction is to avoid such errors, without explicit background extraction. (ii) In case of a class like 'helicopters,' we observed a consistent difference in the camera pose in the image (looking up to the object) and model renderings (looking horizontally at the object). This leads to higher than usual view estimation error (around 17°).

3.7 Exploring Image and 3D Model Collections

We now describe how to use the output of the jointly analyzed image and 3D model collections for multi-modal data exploration. Figure 3.21 shows the user interface[2].

[2]A video showing this interface is available at `https://youtu.be/m584yqGtlCE`

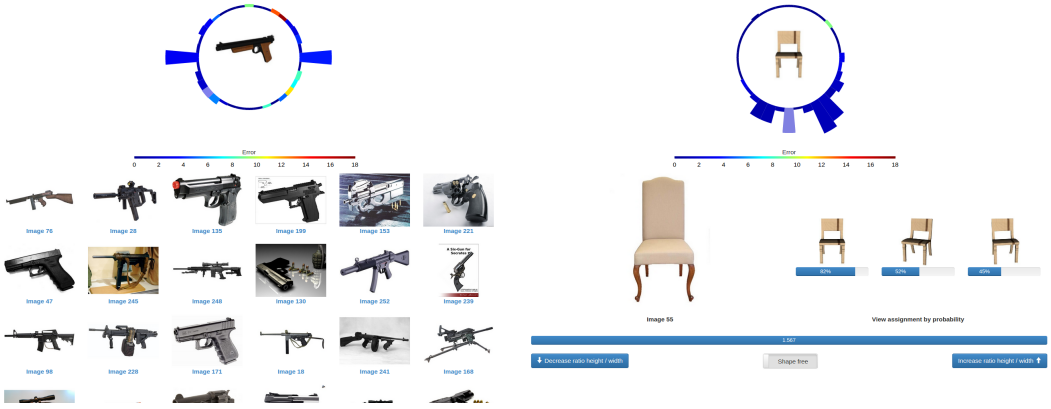


Figure 3.21: The view-shape refactored image collections and the filtered and consistently coaligned model collections enable novel exploration possibilities. (Left) User selects a view by posing the model icon in the view-dial, while the system retrieves the top rated images for the indicated view. (Right) Any selected image can be used to probe for other images in a comparable view or with a comparable shape attribute.

CROSSLINK produces filtered and coaligned model collections, and image collections resorted by view-shape attribute axes. The user can then select a 3D model (shown as an icon) and use the provided view-dial to interactively pose a view (i.e., vary azimuthal angle) as the system retrieves the top rated images for the selected view. The height of the bars indicates the number of images in that view, while the color indicates confidence in the view estimates. The user can click on any image to further probe the confidence in its view estimate. More interestingly, the user can ask for images (from the same view) of objects with higher/lower shape attribute values. This interaction makes use of the discovered links between the two data repositories. Note that while this mode is very natural using our view-attribute reordered images, performing comparable actions using the raw image and/or model collections would be cumbersome and very difficult using existing query interfaces.

3.8 Conclusions and Future Work

We presented a framework for joint processing of image and 3D model collections that exploits the strengths of each data modality to improve tasks in the other. As a key difference to standard image/shape analysis approaches, we investigated how to

factor out *both* shape and (camera) pose variations across such collections, and thus reveal their underlying structure. Our proposed framework is easy to scale, and does not attempt to explicitly compute point- or patch-level correspondences, or background segmentation on the image. Technically, we modeled how pose variation manifests as image-space feature variation, and then demonstrated how to factor out such variations to reveal consistent shape attribute-based reordering on images across multiple views. One important result is that through this framework we now have access to a large, co-aligned set of 3D models of a diverse set of classes. Many shape analysis techniques benefit from this (see related work in Section 2.2). Moreover, this model database will serve as input to the methods of Chapters 4 and 5. There we will show the utility of this database for 2D scene understanding. Interestingly, the cross-modal links are again traversed in both directions: from 2D to 3D for cleaning up the model database, and from 3D to 2D for aiding 2D scene understanding.

Finally, we extensively evaluated our framework to demonstrate that cross-domain processing not only results in cleaner and more consistent image and 3D model search, but also enables novel exploration possibilities.

While we presented a first framework to jointly exploit correlations across across image and 3D model collections, there are many exciting and important questions that need to be investigated:

1. A natural next step will be to investigate how 2-parameter view variations (i.e., include altitude variations) beyond one parameter view variations as investigated here.

2. The domain adaptation technique we used for shape attribute regression can also be used for the other parts of the pipeline. As performance in these other parts was good without this extra layer, we did not perform any domain adaptation there. Performance might still increase by using the geodesic flow kernels throughout the pipeline.

3. The shape attribute regressor currently does not take into account the circular

nature of the data – as with the model alignment, realizing that renderings from very similar views will have similar regressor weights could be used as an extra regularizer. Note that this regularization would need to be carefully combined with the geodesic flow kernel.

4. Finally, as a long term goal, we would expect to use such cross-domain connections along with advances in material modeling and semantic links [134] to eventually unify image and 3D model collection, and thus be able to naturally transition between the two representations.

Chapter 4

Scene Structure Inference through Scene Map Estimation[1]

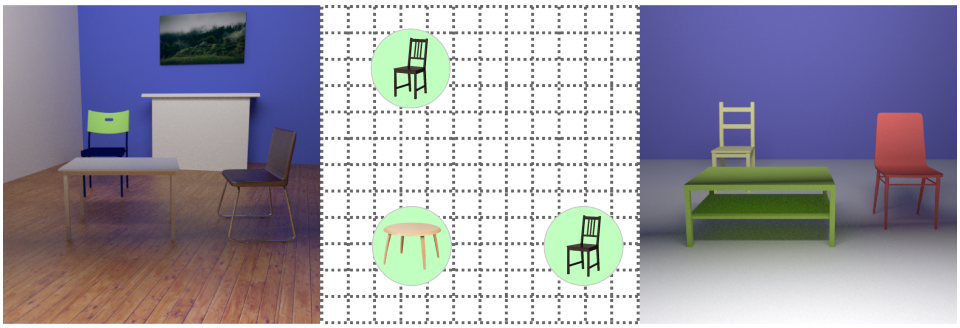


Figure 4.1: Given a single RGB image (left), we propose a pipeline to generate a scene map (middle) in the form of a floor plan with grid locations for the discovered objects. The resulting scene map can then be potentially used to generate a 3D scene mockup (right). Please note that our system does not yet support pose estimation; hence in the mockup the objects are in default front-facing orientations.

4.1 Introduction

In the previous chapter, the analysis of iconic images in-the-wild was discussed through the lens of 2D and 3D retrieval. Most in-the-wild data, however, is not iconic. In the 2D domain, images of our environment are much more common. These images mimic our own visual input in our daily environment. Our ability of reasoning from this visual input is vital, both to the constant and continuous analysis

[1]Published at the International Symposium on Vision, Modeling and Visualization 2016 [55]

of our surroundings, as well as to the formation of a correct response to this analysis. If we are to create intelligent systems capable of navigating the intricacies of the real world as well as human beings, we need to find ways of recreating this impressive mental ability. Hence, not surprisingly, indoor scene understanding has received significant research attention in both computer graphics and vision.

A core subtask of indoor scene understanding is *scene structure inference*, i.e., deducing the presence and the locations of individual objects composing the scene. Given a single image, as humans, we can in most cases tell the class of the objects and their relative positions in the scene. Of course, from this information a lot can be inferred, such as an unobstructed path through the room, scale, and scene type (a room containing a bed and a chair is likely to be a bedroom, while a room containing a desk and an executive chair is likely to be an office). Such floor plan-level information thus provides a compact and useful summary about the nature and the structure of the scene (see Figure 4.1).

One possible solution to inferring such a scene floor plan from a single image is to merge state-of-the-art solutions to both semantic segmentation and depth estimation to define the location of all objects in the scene. Such an approach has several disadvantages. Firstly, it is not trivial to delineate the boundaries of the individual objects in the image, as most of the existing semantic segmentation approaches do not produce instance-aware segmentation [103]. Secondly, both semantic segmentation and depth estimation model each pixel individually. Fundamentally, as we are only interested in the relative location of the objects, the intermediate steps that involve pixel-level labeling can introduce a significant source of error affecting holistic scene understanding.

To circumvent these problems, we propose a novel representation, called a *scene map*. The scene map models the structure of an indoor scene using a collection of grids that mark the location of objects of different classes in a top-down view of the scene (see Figure 4.2). Importantly, as we target directly the global structure of the scene compared to e.g., extracting placement in world coordinates, low resolution grids suffice for this task. This limits the number of variables necessary to

train and estimate in practice to the bare minimum.

In this chapter a pipeline is presented for estimating the scene map from a single image. At its heart lies a convolutional neural network, based on the successful VGG architecture [104]. As training data for scene maps is scarce, we create a rendering pipeline that synthesizes scenes using the model set generated by the method in Chapter 3 and renders them on the fly, supplying the network with a virtually unlimited amount of training data. Using this synthetic training data, the network is trained end-to-end. The pipeline's performance is compelling, with 52% of models being located within one grid cell of their ground truth location.

We compare the method with a baseline that combines state-of-the-art semantic segmentation and single frame depth estimation. Our evaluation shows that the scene map representation gives more accurate results for this task, while needing to solve for a significantly fewer variables than its baseline counterparts. We conclude with discussion of limitations and future work.

The main contributions thus include:

- Introducing the scene map as a representation for holistic scene understanding;
- suggesting a method for synthesizing scenes together with their scene maps by exploiting a 3D model collection, and using this data to train a convolutional neural network; and
- proposing a method for inferring the scene map given a single frame as input, using our learning pipeline.

4.2 Method

Our method infers scene structure from a single RGB image. It does so by learning a mapping from the input to a new representation called a *scene map* through the use of a deep neural network. We will first discuss this new representation, and then detail the network architecture at the core of our method. Finally, we explain our

Figure 4.2: A scene map describes the scene on a per-class basis from a top-down view corresponding to an input RGB image. A white square indicates the presence of an instance of that particular class at that location. Here we show the groundtruth scene map, while our result can be seen in Figure 4.7, bottom row.

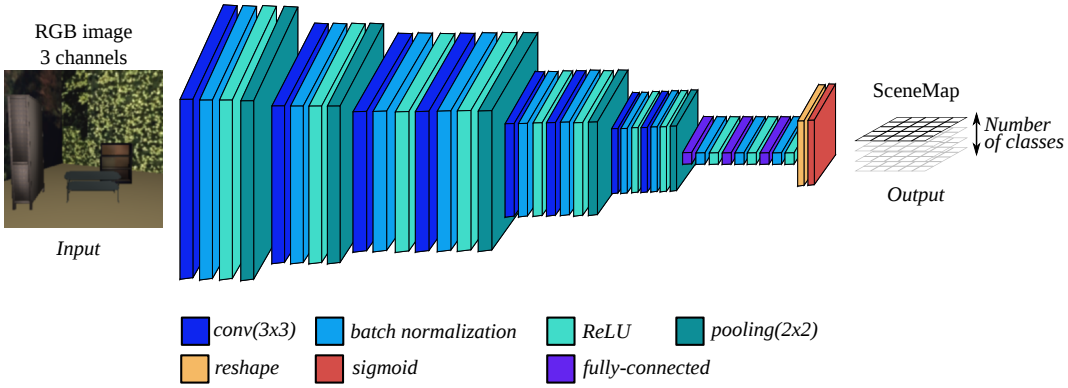


Figure 4.3: Our network architecture, based on VGG11.

synthetic rendering pipeline, which feeds the network with an unlimited supply of training data, which helps to offset the lack of real training data for this purpose.

4.2.1 Scene Map

Our system takes as input a single RGB image, and outputs a top-down view of the scene called a *scene map*. Intuitively, the scene map provides a two-dimensional summary of the objects present in the scene and their relative positions in a way that is similar to a floor-plan. The two coordinates of the scene map correspond to the x and y coordinates of the plane parallel to the floor of the given indoor scene, and the values stored at a particular coordinate correspond to the objects present at that position. Importantly, the scene map completely removes the third coordinate (height), and only represents the floor implicitly by using it as a frame of reference for other objects. As we demonstrate below, such a reduced representation

greatly facilitates the inference and learning tasks, while still providing a very useful summary of the overall scene structure.

More precisely, assuming that the scenes contain objects belonging to N different classes, a scene map $\boldsymbol{S}$ consists of grids $G_i \in \boldsymbol{S}$ of resolution $r \times r$, with $i \in \{1 \dots N\}$, giving one grid per class. Each grid is represented as a binary matrix, which marks the locations of all instances of any class in the scene; see Figure 4.2. This representation is inspired by the popular *occupancy grid* representation commonly used in robotics applications for 3D mapping [112], where the 3D environment is modeled as an evenly-spaced field of binary random variables, taking the value 1 when an obstacle is present at the corresponding location. Thus, a scene map can be considered as a spatial-semantic occupancy grid, with 2 dimensions reserved to spatial coordinates and the third to class identity.

The scene map is of limited resolution by design. As we are interested in the general layout of a scene, a margin of e.g., 30cm in the placement of an object could be acceptable. By assuming such a margin, the number of variables in the placement problem ($r \times r \times N$) is significantly reduced compared to using a fine grid, or to modeling the problem in the original pixel space ($w \times h \times N$). This type of simplification is encountered equally in computer vision applications that convert regression into classification: e.g., in [4] where the aim is to predict ego-motion encoded as a rotation-translation movement. Instead of regressing to precise (continuous) angle and translation values, the problem is converted into a classification task by binning each movement into a fixed (discrete) number of ranges of movement magnitude. This choice results in a sensible trade-off between accuracy and complexity in problems where very precise predictions are not mandatory.

In our setup, the scene map is designed to encode a square area on the floor of the scene in front of the camera of $6\text{m} \times 6\text{m}$ in size. This is large enough to accommodate more than 95% of the scenes in the SUNRGB-D dataset, and can easily fit the average UK room size[126]. We use grids of size 16×16, resulting in a grid cell size of 37.5cm.

4.2.2 Scene Map Inference Overview

Our main goal is to compute the scene map representation from a single input RGB image. For this we follow a data-driven approach that has been shown to be effective for a wide variety of image processing tasks. Namely, we train a Convolutional Neural Network (CNN) that, given a single image, tries to output its scene map representation directly, without estimating any low-level attributes such as depth or pixel-wise class labels. One challenge with adopting this approach, however, is that it requires a large amount of training data to be successful, due in part to the large number of variables that typically need to be estimated. Unfortunately, there is no existing sufficiently large dataset that contains ground truth scene map labelings (e.g., the recent SUNRGB-D dataset [106] contains approximately 10000 images). To overcome this issue, we train our network with scenes that we synthesize on the fly by exploiting an existing 3D model collection [73] and varying the composition of the scene and the appearance of the objects using a large texture dataset [89] using a probabilistic model. In particular, we create a scene synthesis pipeline that uses a rendering approach and a randomized object placement and appearance variation model. This pipeline effectively provides our learning framework with an unlimited source of data that we use to train an adapted CNN for scene map inference. To summarize, our general approach consists of the following key steps:

- Adapting a well-developed CNN architecture for inference of scene maps from single images.

- Constructing a randomized scene synthesis pipeline based on a scene composition model coupled with appearance variation and an efficient rendering method.

- Using our scene synthesis method to train the network by generating a large number of ground truth pairs consisting of an image and its associated scene map.

- Using the trained network to estimate the scene map on a new test image.

Below we describe each of the individual steps of our pipeline and provide the corresponding implementation details.

4.2.3 Network

To learn the mapping from the RGB image space to the scene map representation, we use a deep neural network that builds upon VGG11 [104], with a few modifications (see Figure 4.3). Notably, we added batch normalization after each convolutional and fully-connected layer, resulting in a significant decrease in training time [59]. The original VGG11 maps the input image to a discriminative feature representation in $\mathbb{R}^{1024}$, then uses a classifier to predict a class label for each image. Since our problem requires a spatial representation and not a single class label, we remove the classifier and instead reshape this representation to the desired scene map representation of size $r \times r \times N$. Note that most architectures designed for spatial tasks (e.g. for semantic segmentation) use mirrored encoder-decoder networks, enforcing direct correspondences between the feature maps learnt by the encoder and the decoder at each level [85]. But in our case such architectures are not justified, since the input domain (image pixels) is different in resolution and viewpoint from the output domain (grid cells). However, investigating for more adapted architectures for our task constitutes a direction of future work. The result is passed through a sigmoid layer, so that each cell in the grid reflects the likelihood of an object of a given class being present. The overall architecture has 20 million parameters.

Training. We have implemented the proposed network using Torch. The RMSprop optimiser [113] was used with an initial learning rate of 10^{-3}, and a learning rate decay of 0.8 after every 10000 iterations. Note that the training is done from scratch, since no pre-trained models are available for VGG11. The training was performed on a multi-GPU system (4 GPUs, 12G memory each), with batch size of 32, and took approximately 10 hours to converge.

4.2.4 Non-maximum suppression

Each cell within a class grid shows the confidence of the network about the presence of an object of that class at the corresponding spatial location. However, the

network is often uncertain about the precise location of an object. This uncertainty is expressed by a spreading of the probability across multiple cells in the vicinity of the actual location. Note that this behavior is justified considering that depth estimation from a single image suffers from scale ambiguity, especially when a certain object has not been seen before. Deep learning approaches for depth estimation from single RGB images [29] try to bypass this issue by implicitly learning absolute scale ranges for each object from the large number of training examples. The intra-class scale variability will dictate the range width, and eventually the accuracy that can be obtained; wide range resulting in more uncertainty in the output. A very simple idea to reduce uncertainty and binarize the probability maps would be to use a fixed cut-off value, e.g. 0.5, and deem every cell with an output probability of 0.5 or higher to contain an object of that class. However, we found that performing a max pooling post-processing step, with a 3×3 window, results in sparser, more accurate scene maps than direct thresholding. Hence, we use this approach in our experiments (see Figure 4.4).

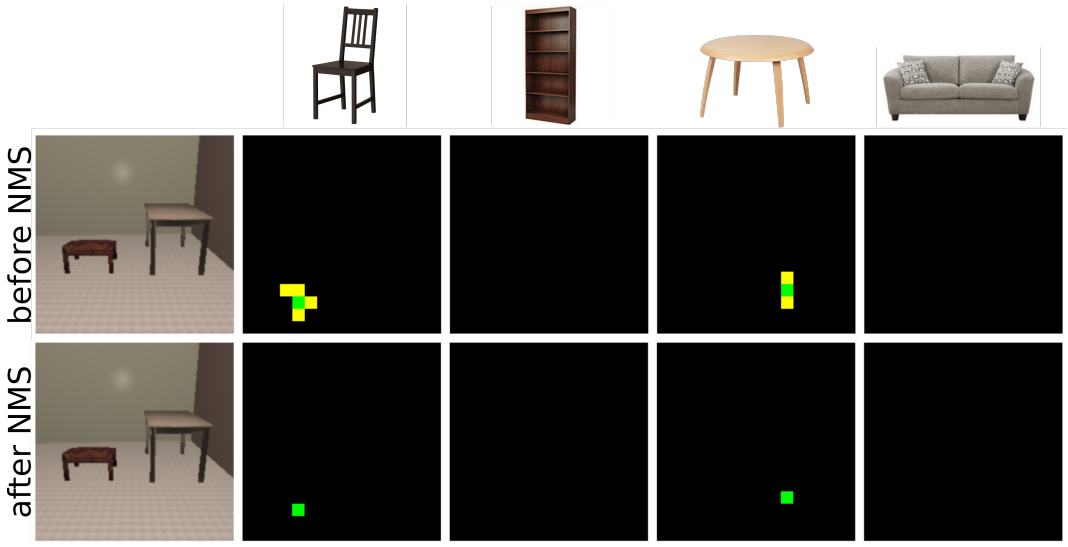


Figure 4.4: Result of non-maximum suppression. Yellow cells represent false positives, green cells true positives. The top row shows the scene map after simple thresholding at 0.5. This results in spurious activations around the true location of each object. After non-maximum suppression, these are removed, with only the local maximum (the true positive) being left.

4.2.5 Rendering pipeline

Training a deep neural network requires large amounts of training data. The largest available dataset for our purpose, SUNRGB-D [106], contains approximately 10000 images with 60000 bounding boxes of 1000 different classes. We have found this not to be enough for training a network that generalizes well. To boost our training data numbers, we set up a synthetic rendering pipeline, which renders training pairs of images with the associated scene maps on the fly. This provides the system with an unlimited stream of essentially unique training data (although theoretically two scenes could be identical, the probability of this is vanishingly low). This is an instance of *online learning*, which has self-regularizing capabilities, limiting the risk of overfitting [14].

Data. The rendering pipeline takes a set of class-labeled objects $\boldsymbol{O}$ and textures $\boldsymbol{T}$ as input. In our experiments, we take $\boldsymbol{O}$ as a subset of the IKEA dataset [73]. This dataset contains objects of four classes *chair, shelf, table* and *sofa*, with 16 objects per class. We manually curated all models to have accurate relative scale and to be centered at world origin, with consistent orientation. The texture set $\boldsymbol{T}$ consists of a subset of 136 textures from the VisTex dataset [89], which we curated manually to be appropriate textures for furniture. Both $\boldsymbol{O}$ and $\boldsymbol{T}$ are separated into training and test sets, using a $75\%/25\%$ split, as illustrated in Figure 4.5.

Scene generation. When the pipeline is queried for a new training example, a new random scene is generated (see Figure 4.6). First, a subset $\boldsymbol{o}$ ranging from 2 to 6 objects from $\boldsymbol{O}$ is sampled at random with replacement. These objects are then randomly placed around the world center in a 6×6 meter square. The objects are randomly rotated around the up-vector in increments of 90 degrees. We make sure the objects placed do not collide with each other. Inspection of 100 publicly available indoor photographs shows that one wall is nearly always visible, with a second perpendicular wall being visible approximately 75% of the time. As such, two perpendicular walls are placed around the scene, with random, but coherent orientation. Finally, the objects and walls are individually textured by randomly sampling and scaling from our texture set $\boldsymbol{T}$. Note that by including small texture

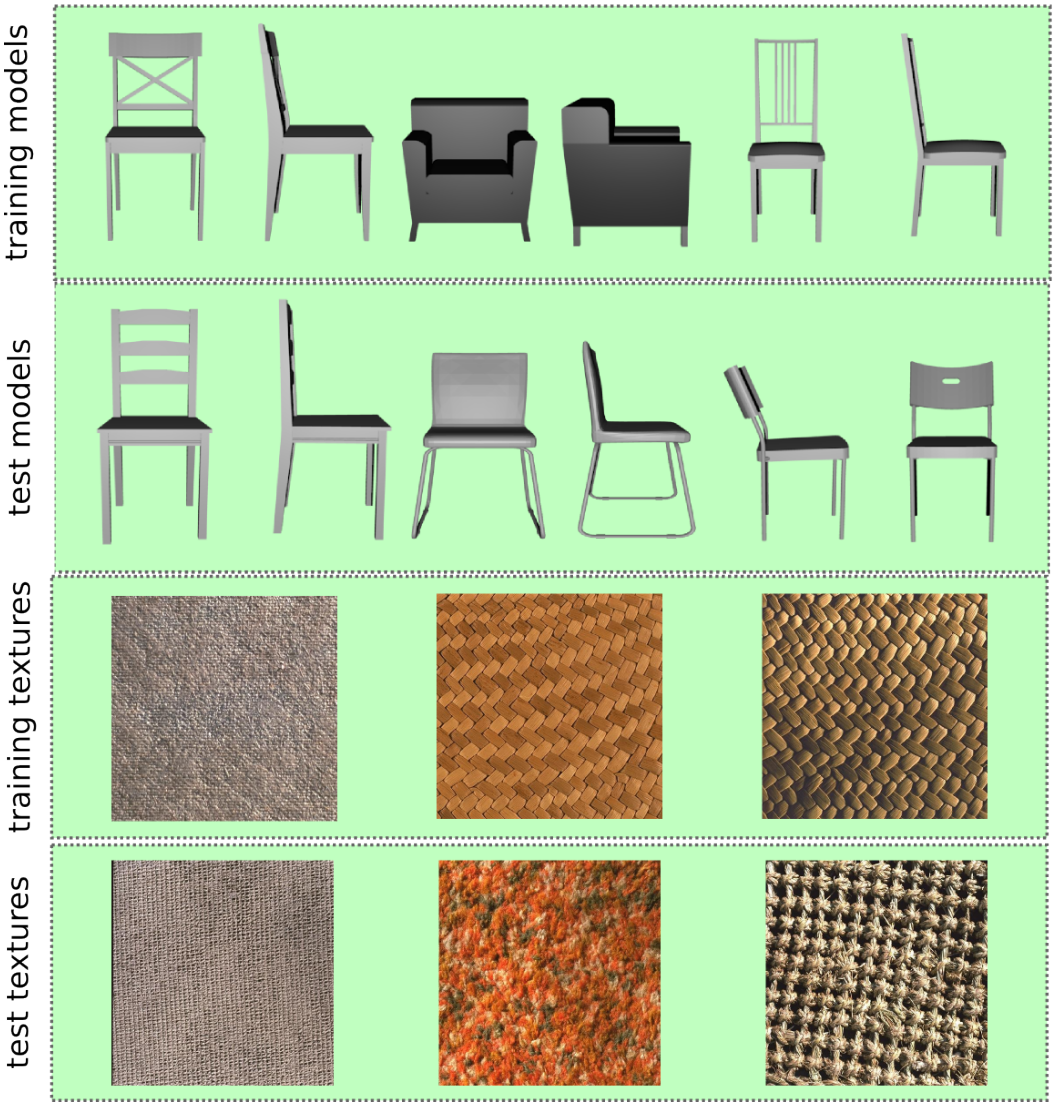


Figure 4.5: We split models and textures into training and test sets at a 75%/25% split. This allows us to test how well the network learns the general shape of each class of object.

scales, we mimic textures with repeating patterns.

The camera is placed at a height drawn randomly in the range between $1m$ and $1.8m$ to mimic the range of heights from which most handheld photographs are taken.

Rendering. The generated scenes are then rendered using an OpenGL setup with a simple Phong shading model. For each scene, we also generate the scene map, semantic labeling, and depth ground truth. The latter two are used only for baseline comparison (see Section 4.3.1). See Figure 4.6 for some samples from the rendering

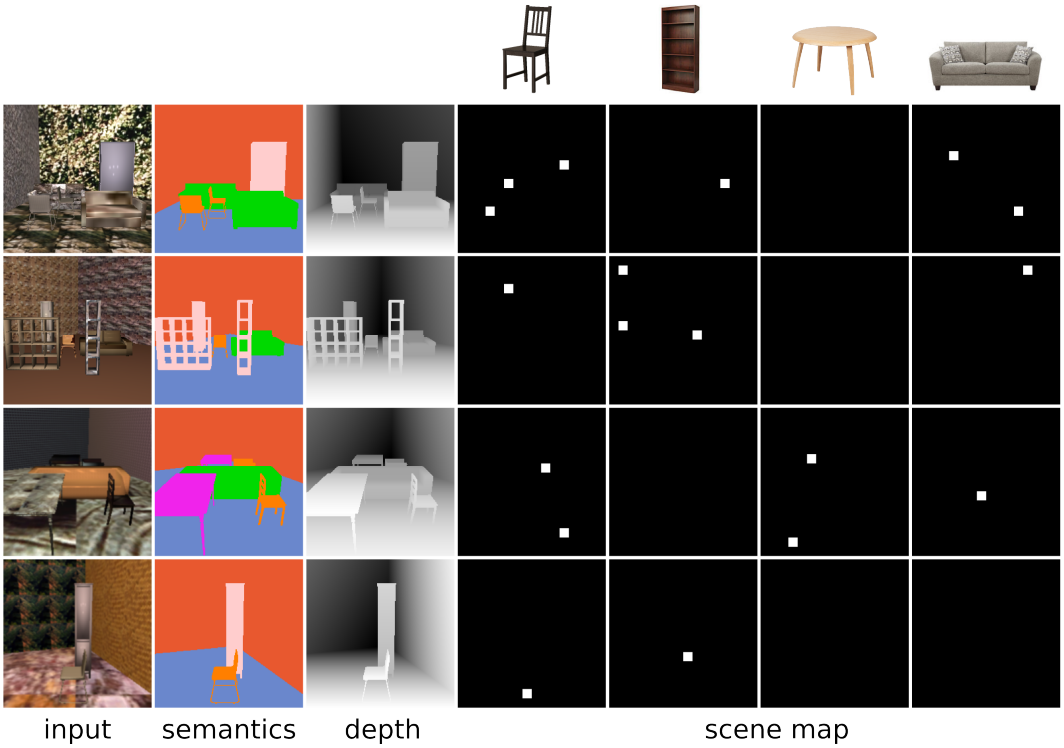


Figure 4.6: An illustration of 4 samples from the data generation pipeline. We render RGB, semantic segmentation, depth, and scene map, resulting in an unlimited stream of fully defined training data.

pipeline.

4.3 Evaluation

In the preceding sections we proposed the scene map as a representation for holistic scene understanding, reasoning that its low dimensionality removes unnecessary variables from the optimization process compared to dense, pixel-based approaches. Below, we compare our scene map estimation method with a dense approach that combines the output of semantic segmentation with depth estimation, both from single frame RGB input.

4.3.1 Baseline

Semantic segmentation. The semantic segmentation pipeline we use is a version of [85], using VGG11 [104] as the basis encoder instead of VGG16 (to be comparable with our pipeline in terms of depth), as well as with the fully connected layers removed. The final layer has 6 output maps, one for each class (i.e., chair, shelf,

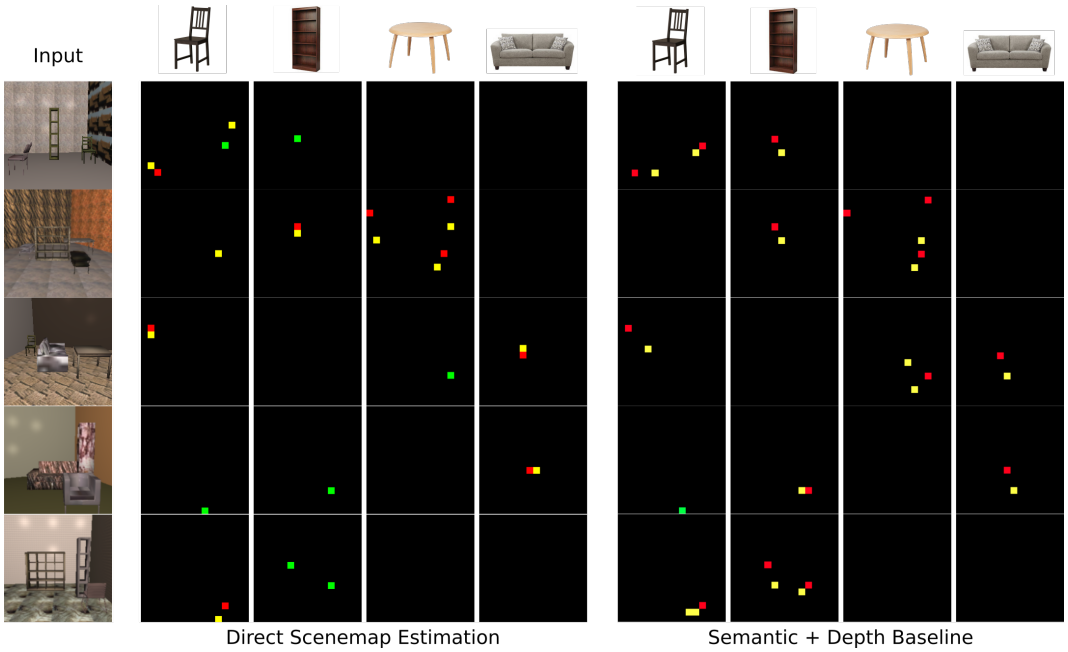


Figure 4.7: Left: generated scene map using our pipeline. Each column represents a specific class, each row is one sample. Ground truth is represented using cell color: green grid cells indicate true positives, yellow grid cells indicate false positives, red grid cells indicate false negatives. Right: scene map generated using the semantic segmentation + depth estimation baseline.

table, sofa) plus two for the wall and floor. We use a spatial cross-entropy loss, classifying each pixel individually. This pipeline is trained from scratch on the same data as our scene map pipeline (see Section 4.2.5).

Depth estimation. Our depth estimation network is similar, but the final layer outputs just a single map instead of the number of classes as before. We use a log mean squared error loss on the output [29].

Combining outputs. We convert the output of the above two networks into a scene map. First, connected components are extracted from the semantic segmentation. Then, for each component, we find the average depth using the estimated depth map. Using the known focal length of the camera we then compute the 3D location of the center of each component, and project this into the scene map.

Performance of networks. By themselves, these networks show high performance in their respective tasks on this data. On the test models and test textures, the semantic segmentation network shows an accuracy of 96.5%, and the depth network

Table 4.1: Comparison of our method with the semantic segmentation + depth estimation baseline. TPR is true positive rate, TPR+1 and TPR+2 are true positive rates when respectively off-by-one and off-by-two errors are allowed. FPR is false positive rate. Note that as most of the grid is empty, false positive rates are very low.

Method	Model set	TPR	TPR+1	TPR+2	FPR
Baseline	Train	0.03	0.20	0.47	0.0037
Ours	Train	**0.24**	**0.66**	**0.82**	**0.0029**
Baseline	Test	0.02	0.19	0.43	0.0043
Ours	Test	**0.15**	**0.52**	**0.71**	**0.0031**

an rMSE of 17cm. These unusually high numbers (for reference, [85] report an accuracy of 72.5% in the original paper) are in part due to the unlimited amount of training data our synthetic rendering pipeline provides the network. Moreover, the data used in [85] likely has higher variability and noise than our data, as they come from real photographs instead of synthetically rendered scenes, and hence are more challenging.

4.3.2 Training vs. test

As discussed in Section 4.2.5, all training data is generated synthetically, resulting in images very unlikely to be seen twice, but the models used for generating the images are seen many times over. We will evaluate the network both on images generated using these same models, as well as images generated using the models in the test set. Both scenarios are plausible: one can imagine the case where the types of models used in the scene are known in advance, or the case where only the classes are known.

4.3.3 Comparison

In Figure 4.7, we show the output of our pipeline, as well as the output of the baseline. Our method shows a clear advantage in performance over the baseline method. Note that although our method does not always find the perfect location, in virtually all cases the presence of an object is detected. The baseline sometimes misses an object entirely, and often activates two cells for a single object.

In Table 4.1 we compare the accuracy of our method with the baseline quan-

titatively. Performance is evaluated on both training models as well as test models. Aside from the true and false positive rates, we also report the performance when "one-off" and "two-off" errors are counted as correct (i.e., an object detection one or two cells away from the ground truth is still counted). Our method significantly outperforms the baseline on all settings. It is interesting to note that for the baseline there is not much difference in performance between the training and test models. For our own method, the decrease in performance from training to test is more significant, while still outclassing the baseline by a compelling margin.

4.3.4 Effect of object density

To test our pipeline's scalability with respect to the number of objects in the scene, we tested two scenarios where the this number was respectively increased to a range of 6 to 9 objects, and 10 to 15 objects. These scenarios generate scenes with objects very close together, making the task of distinguishing objects more difficult. Sample results can be seen in Figure 4.8. Clearly, the network is having increasingly more difficulty placing each object at the right location. Moreover, it more often fuses two objects together, resulting in just a single cell being activated. Table 4.2 shows the quantitative decrease in performance as the number of objects in the scene increases.

This decrease in performance is not unexpected. Indeed, when only a small part of given object is visible, identifying it as well as placing it correctly becomes intrinsically harder. We experience this as humans as well – when objects are heavily occluded we make use of contextual information to make correct inferences about object identity and placement, something that is missing here, as scenes are generated randomly. This insight is one of the foundations of the method in Chapter 5, where we use a model of object co-occurrence to guide an indoor scene reconstruction process, resulting in better performance under occlusion.

Table 4.2: Effect of object density on performance of our pipeline. Performance decreases with increased scene occupation, but does not dwindle as to be unusable.

Number of objects	TPR	TPR+1	TPR+2	FPR
5-9	0.11	0.42	0.58	0.0044
10-15	0.09	0.32	0.44	0.0053

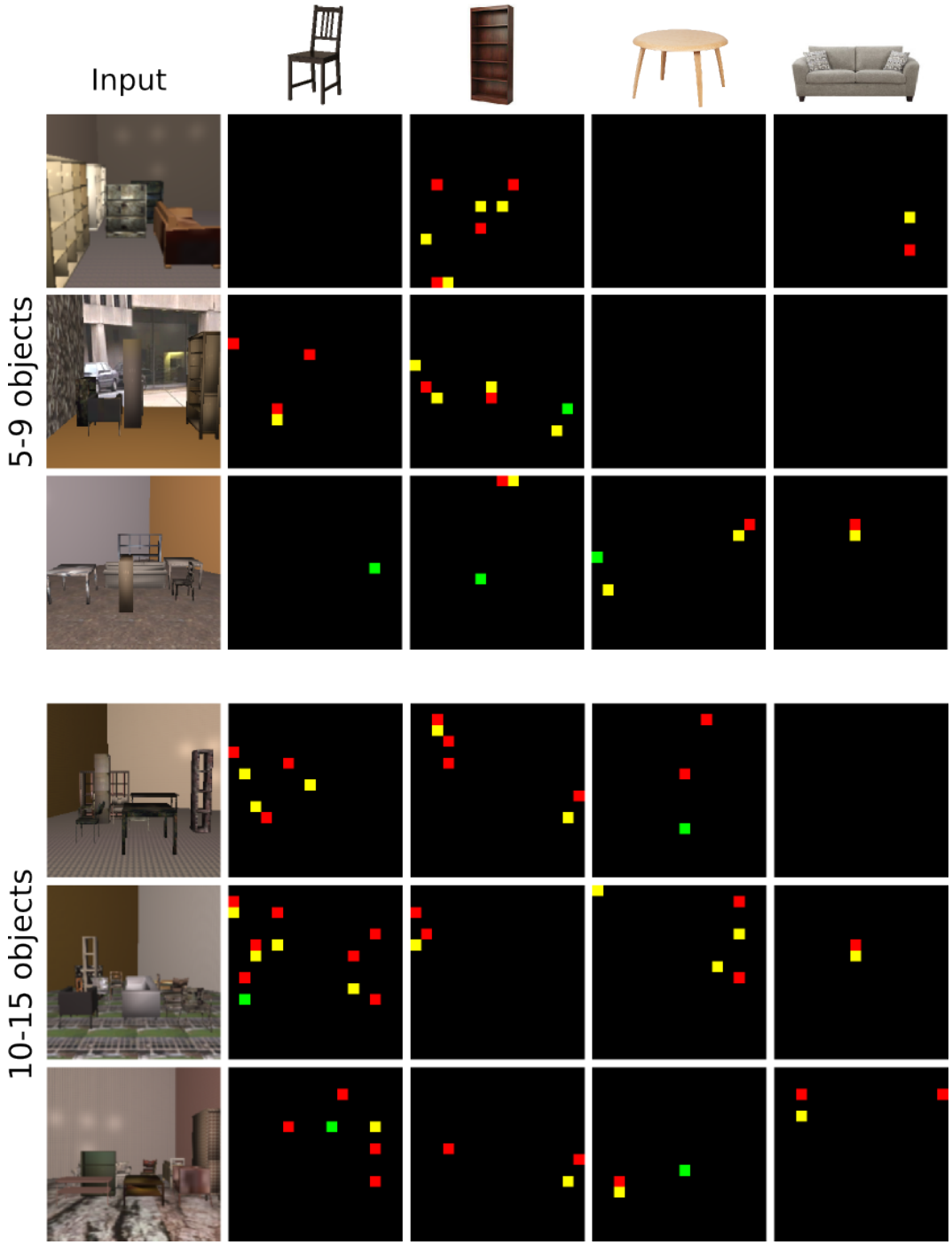


Figure 4.8: Results with increasingly dense scenes. Ground truth is represented using cell color: green grid cells indicate true positives, yellow grid cells indicate false positives, red grid cells indicate false negatives. The precise localization of objects becomes more difficult, but in general the presence of objects is still inferred correctly.

4.4 Discussion and Conclusion

We have presented a new representation for scene structure called the *scene map*. It reduces the number of parameters necessary for representing scene structure to a minimum, thereby reducing the necessary variables to estimate during optimization. Although the accuracy of the proposed method is limited by design through the size of the grid cells, our output can be directly used for a number of tasks, some of which are detailed below. This is opposed to pixel-wise approaches, which are designed to output accurate predictions, but whose output necessitates non-trivial post-processing to become usable in practice, as shown in our evaluation.

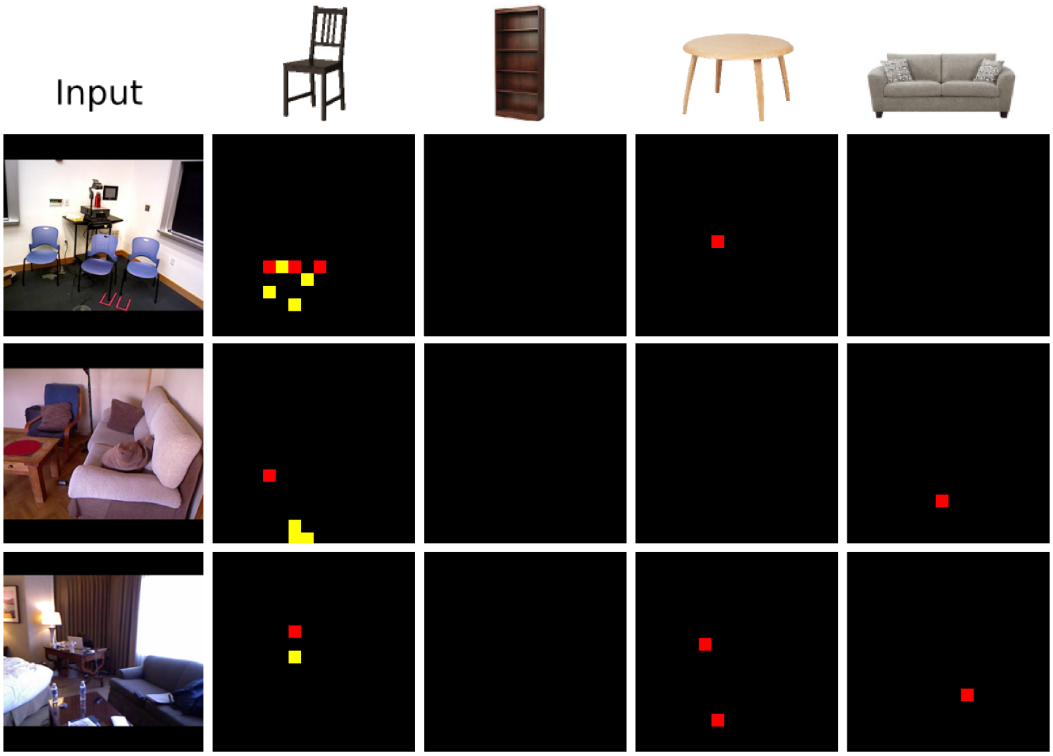


Figure 4.9: Result of our method on real data. The disparity between the feature space of the synthetic training data and that of real photographs unfortunately limits the efficacy of our method here. Improving domain adaptation and training on a large set of real training data could mitigate this limitation.

Future work. While we proposed a first pipeline to extract scene maps from single RGB images, several refinements remain to be explored. As it stands, not enough real data is available to train our network from scratch. The synthetic pipeline does not produce accurate results on real images due to the discrepancy between the

feature response of synthetic renders and photographs (see Figure 4.9). To extend our method to real images, a method for domain adaptation between synthetic and real images (e.g., SUNRGB-D) needs to be devised. Note that the output of the next chapter's pipeline can be converted to scene maps, although the problem being solved is a more complex one. Moreover, comparison with other baseline methods is possible. For example, in the current baseline the semantic segmentation step could be replaced by an object detection pipeline (e.g. [94]). Finally, evaluating on a larger set of classes is needed to show applicability on more varied scenes.

We believe that scene maps extracted from a single image can be directly used for multiple purposes, as they provide a complete summary of the composition and structure of the scene. For example, they open the possibility of automatic retrieval of images with specific scene configurations. In a complementary task, scene maps can help to automatically extract statistics of space utilization from large image datasets [102, 106]. Such statistical models could be used for different tasks such as improved scene synthesis and scene type classification. Finally, when combined with in-class model retrieval and a pose estimation pipeline, scene mockups can be potentially generated from scene maps, which in turn can be helpful for architectural visualization and scene relighting. Such a method is the scope of the next chapter (Chapter 5).

Chapter 5

Finding Chairs in Indoor Scenes under Heavy Occlusion using Scene Statistics

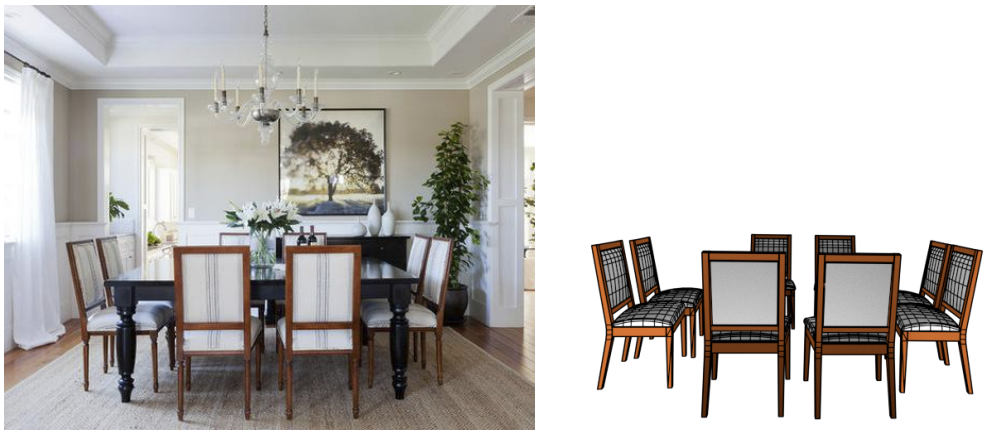


Figure 5.1: We propose a method for generating mockups from a single input photograph of an indoor scene.

5.1 Introduction

Large sets of 3D indoor scenes are useful for purposes ranging from architecture and product design to virtual reality content and game asset creation. Aside from being used directly as a resource for rendering or interactive purposes, statistics extracted from them can be used to gain insight into how objects are commonly

used, and how they are commonly placed with respect to each other. However, such sets are unfortunately hard to come by and expensive to manually create. In contrast, 2D photographs of such indoor scenes are widely and freely available, as are large databases of individual 3D models. It makes sense, then, to try and convert 2D photographs to 3D scenes making use of 3D models. This gives rise to an essential problem in computer vision and graphics, which we will henceforth call the *mockup problem*: given a single 2D photograph and a database of 3D models, place instances from this database into a 3D scene as to reconstruct the photograph as accurately as possible.

The problem is inherently ill-posed, as photographs are the result of the projection of many complex attributes (e.g. geometry, material, illumination). Indeed, we are faced with reconstructing an entire dimension that was lost when the photograph was taken. Additionally, inter- and intra-object occlusions limit the amount of visual information available for certain objects, making the reconstruction process more difficult still. It is possible for computer vision algorithms to make reasonable inferences from a single natural image by relying on relevant *prior knowledge* about the image in question. Even so, the complexity of the problem, together with the difficulty of gathering large amounts of training data, makes the mockup problem a highly challenging one.

Recent advances have addressed parts of the goal by looking at simpler problems, such as object recognition [46], localization [94], and pose prediction [128]. Unfortunately, these techniques are designed for objects that are almost fully visible, and fail under moderate to severe occlusions, making them useful only for the simplest of scenes. Moreover, they work on a single object basis, discarding any more high-level information that might be present. Simply combining these methods thus yields limited success (see Section 5.4).

In this chapter, we suggest that in order to improve beyond this baseline, we need to reason about the scene on a more global basis, and inject deeper knowledge of the domain into the optimization process. Our key insight is that scenes typically exhibit significant regularity in terms of co-occurrence of objects, which can be

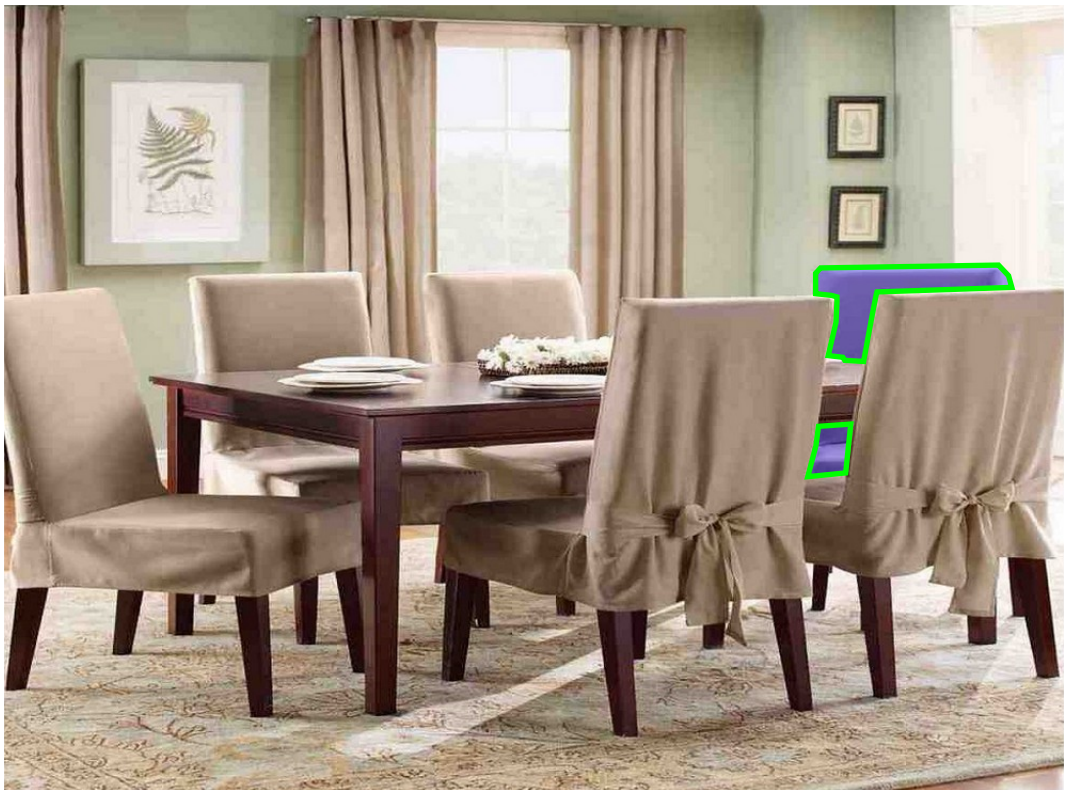


Figure 5.2: The chair marked in blue can easily be distinguished as being a chair through its context, even though most of the object is occluded.

exploited as explicit prior information to make predictions about object identity, placement and orientation, even when such objects are in highly occluded regions, and thus single-object based methods fail.

Intuitively, this approach makes sense – it matches the way we as humans reason under similar noisy conditions. A heavily occluded chair is still easily distinguishable as such due to the presence of other chairs and a table (see Figure 5.2), as we have a good understanding of typical chair-table arrangements. By explicitly modeling this type of knowledge, we can find placements that would otherwise carry too little visual information for accurate recognition.

This insight is captured in our method by combining reprojection error of known keypoints with pairwise object co-occurrence costs in the objective function. Candidate placements are generated and tested on the one hand based on semantic keypoint maps from a newly trained deep neural network, and on the other hand based on the pairwise agreement between instances according to a model of object

co-occurrence statistics, gleaned from a database of pre-existing 3D scenes.

We tested our approach quantitatively on 100 hand-annotated images and show a marked improvement of recognition over baseline methods. Although our current implementation is focused on the *chair* class, the method itself is not inherently limited to this, and could be extended to other classes with appropriate data annotation effort.

The contributions of this chapter are:

- a keypoint estimation network for estimating relevant keypoints of multiple instances of chairs in a single image,
- a pairwise co-occurrence model capturing likelihood of co-occurring chair instances,
- an end-to-end pipeline for finding chairs in single images that beats the state-of-the-art,
- a ground-truth dataset of 100 scenes for testing performance of similar methods.

5.2 Motivation and overview

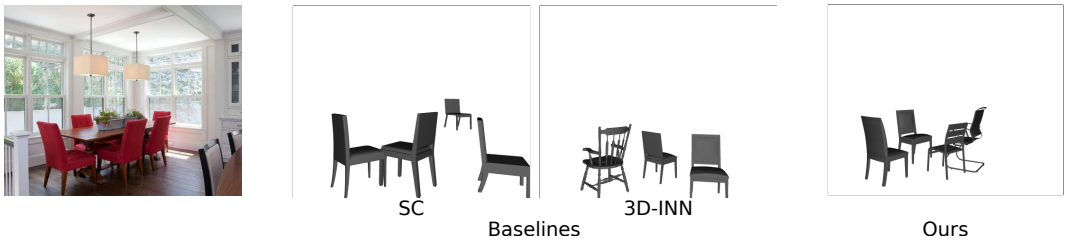


Figure 5.3: Methods based only on the image quickly fail in the presence of less than ideally visible chairs. Our method deals with this situation much better.

To understand the motivation for our approach, we will consider the problem from a high level. Summarily spoken, we are constructing a method that converts a 2D photograph to a 3D scene. The most straightforward and classical way of doing so would be to train some machine learning method on some feature representation

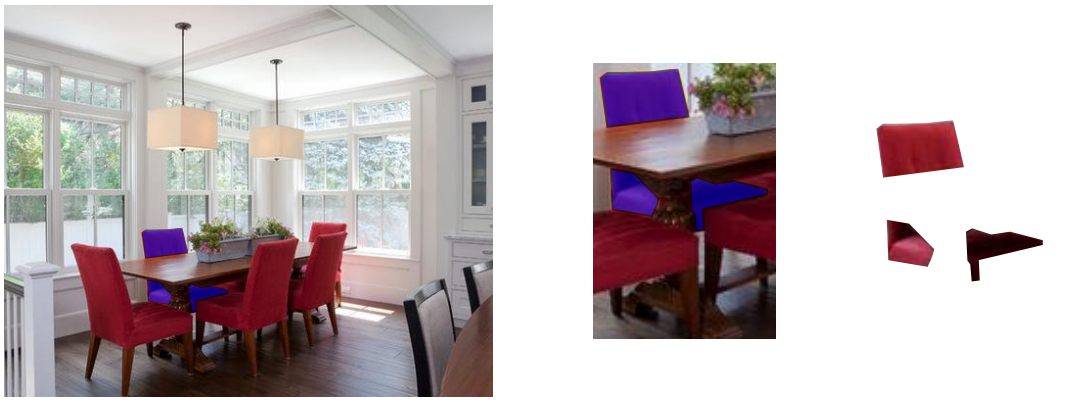


Figure 5.4: As humans, our understanding of scenes is heavily predicated on the context. From left to right, less global information is available, making the classification of the marked object as "chair" harder

of many examples of 2D photograph / 3D scene pairs and use the resulting classifier as our mockup black box. Such an approach can be easily constructed from a combination of existing methods. It turns out, however, that such methods fail badly when confronted with all but the simplest of scenes. In fact, in our evaluation (Section 5.4) we compare our method with two baselines that follow this approach. Foreshadowing some of their results in the left side of Figure 5.3 shows that chairs that are obviously visible get placed correctly, but any instances that are a little harder to see fail to be selected.

To understand this failure, and more importantly how to circumvent it, it is useful to consider how we as humans are capable of understanding these kind of scenes. Looking at Figure 5.4, we see 2 chairs, one heavily occluded and one clearly visible, in 3 different conditions. In the first condition, we see the full scene, in the second only the local context and in the last condition we only see the pixels that belong to the chair itself. It is clear to see that for the recognition of the unoccluded chair the environment is not important – the shape of the object is clearly visible, and we immediately recognize the chair. However, in the heavily occluded case, the task of recognizing the chair becomes easier as more context gets added. For the last column, we might hypothesize that the image regions belong to a chair, but we have no way of confirming this for certain – unless the context is restored.

Clearly, the addition of context gives us extra information in classifying and

posing the objects in a scene. But this is not the whole story. Importantly, the extra information we are given when seeing the entire image is only useful given prior knowledge we have built up over previous experiences. In this particular example, the added context helps us only because we know that chairs often occur together with other chairs and tables. Given this prior knowledge and the global context of the object, our recognition efficacy is enhanced.

This insight is what we capture in our approach to the scene mockup problem: to maximize performance on the mockup task, we need to consider both local information and the context the objects are placed in. Furthermore, to understand this context we need to tell the system what usual scenes look like.

We express these notions in our method as follows: we will extract *local* information from the input image using a keypoint detection network (Section 5.3.2), then *model* the prior knowledge about how scenes are usually built up (Section 5.3.4.1), finally combining this model with the keypoints to find chair instances from a *global* perspective (Section 5.3.4.2). This added high level information allows us to push performance past that of the previously mentioned approach of using only the input data itself (see Figure 5.3, right). In the next section, we will go through each of these steps in detail.

5.3 Method

Our pipeline (Figure 5.5) takes as input a photograph $\boldsymbol{x}$ and a database of 3D chair models $\boldsymbol{M}$, and outputs a mocked up 3D scene $\boldsymbol{S}$, such that the reprojection of $\boldsymbol{S}$ with the separately estimated camera C results in an image as similar as possible to $\boldsymbol{x}$ (see Figure 5.6).

As a preprocessing step, the scene camera C is estimated using an off-the-shelf technique ([47]), giving us focal length and camera orientation (Section 5.3.1). We then enter the main pipeline, which consists of three stages. In the first stage, the image is passed through a keypoint estimation network that outputs a set of *keypoint probability maps*, representing at each pixel the probability of the presence of a certain semantically meaningful keypoint (Section 5.3.2). In the second stage, these

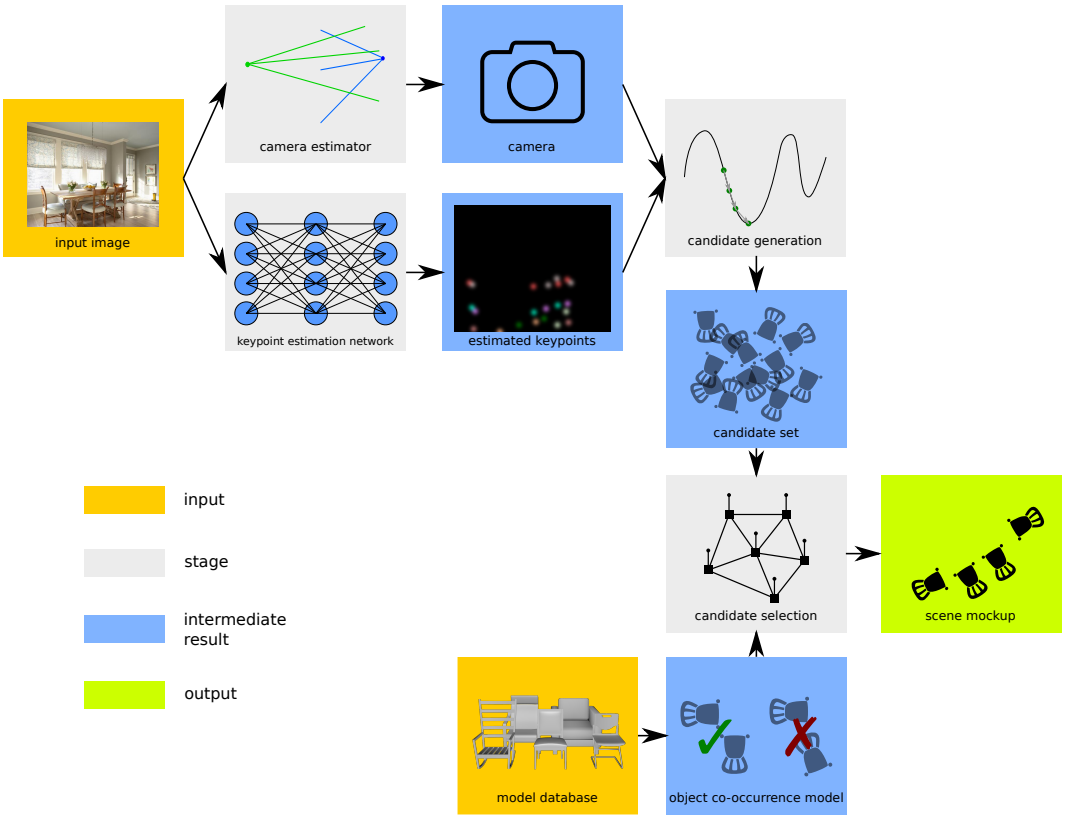


Figure 5.5: The full pipeline of our method.

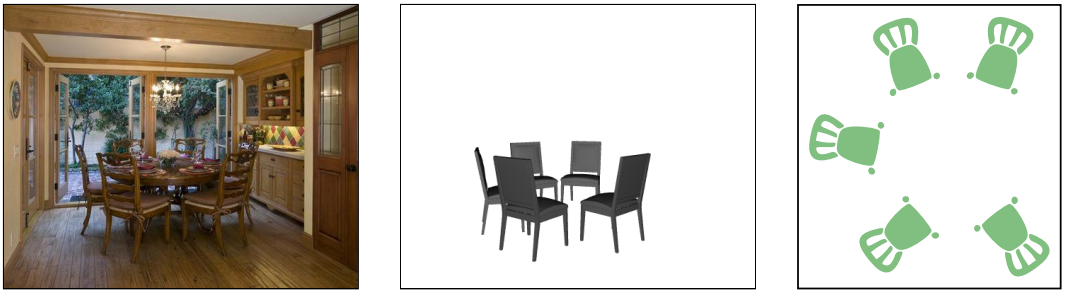


Figure 5.6: Intended working of our method: we take a single image of a structured indoor scene as input, and output a 3D scene with the constituent chairs recovered in the right location and pose, as well as the camera parameters that reproject this scene as close as possible to the original input.

keypoint maps are combined with the estimated camera C to generate candidate object placements (Section 5.3.3). In the third stage, a selection is made among these candidates by optimizing an objective function which combines object-to-keypoint-map matching with pairwise placement agreement according to a pre-trained object co-occurrence model (Section 5.3.4). The second and third stages are then iterated,

this time taking into account the previously found objects during candidate generation as a strong prior (Section 5.3.5). This process is iterated until convergence. We now discuss each of these stages in turn.

5.3.1 Camera estimation

To convert sets of 2D keypoints to possible 3D locations we need the intrinsic and extrinsic parameters of the camera with which photo $\boldsymbol{x}$ was taken. Specifically, for a good reconstruction, we need the orientation of the camera with respect to the ground plane in the form of rotation matrix C_R, the focal length C_f, and a measure of the scale of the room C_s. However, estimating the scale of the room without prior information is not possible – even if we know the 2D location of a chair, it still might be 1 meter or 100 meters tall. There is no way of deciding this without some prior knowledge about chairs and their dimensions. We thus fix our scale parameter and only estimate C_f and C_R, and replace C_s with individual scale parameters for each object in the optimization later on. Most methods for camera parameter estimation indeed focus on C_f and C_r, and to do so rely on automatically estimating vanishing points (see Figure 5.7). We employ the method from Hedau et al. [47]. In summary, their method uses structured learning from Tsochantaridis et al. [116] to rank multiple room layout candidates, which are generated from estimated vanishing points. We refer to the paper from Hedau et al. for more information.

To complete our camera parameters, we pick meters as unit in our world coordinate system (the same coordinate system used by our model set), and set the camera's location C_t as being at eye height (1.8m) on world origin. This altogether yields our camera C.

5.3.2 Keypoint maps

Our goal is to find location and pose of as many chairs in the scene as possible. We aim to do this by finding all instances of a predefined set of semantically meaningful keypoints in the image, and then use the estimated camera together with a 3D chair template consisting of those same keypoints to reconstruct the 3D location and pose of the chairs.

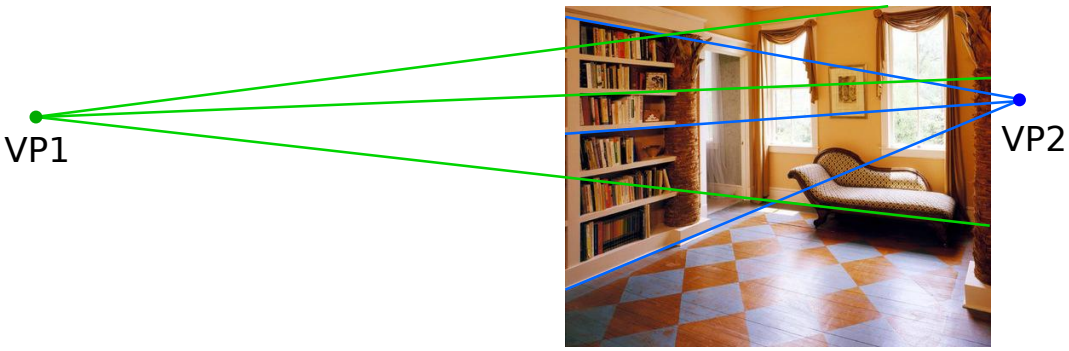


Figure 5.7: By estimating vanishing points in the image, the camera rotation matrix and focal length can be detected. Detecting scale a priori is not possible.

We start by defining a set of general keypoint types for the chair object class. Each keypoint type represents one or more keypoints that should be present in each (reasonable) chair instance. We selected 8 keypoint types, each of which is uniquely identifiable on every reasonable chair. These keypoint types are shown in Figure 5.8.

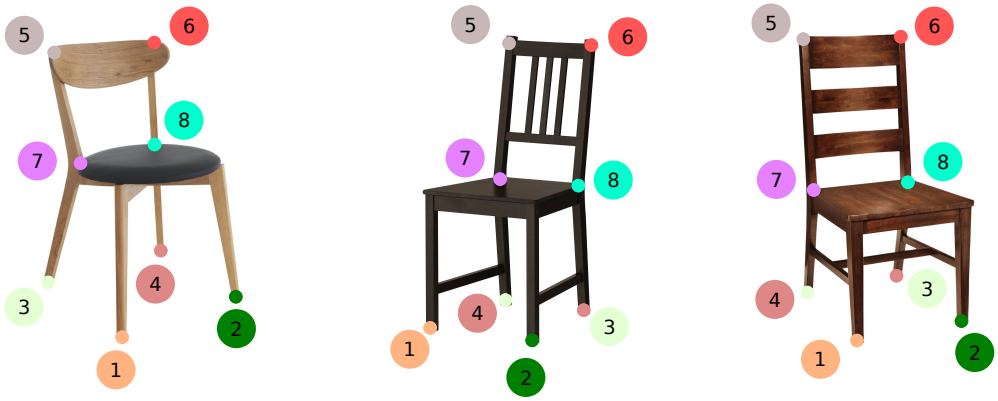


Figure 5.8: Selected keypoint types.

5.3.2.1 Keypoint location map

A keypoint location map is a 2D map whose domain is the input image $\boldsymbol{x}$, and represents belief about the presence of a specific keypoint type at a specific pixel of $\boldsymbol{x}$. It is represented as a $r \times r$ single-channel matrix, with values between 0 and 1. In the case of perfect information, the matrix will have value 0 everywhere except for those locations where a keypoint of the corresponding type is present, where it would have value 1. However, as we will employ an L2 loss function, such step-function keypoint maps would result in an extremely discontinuous error land-

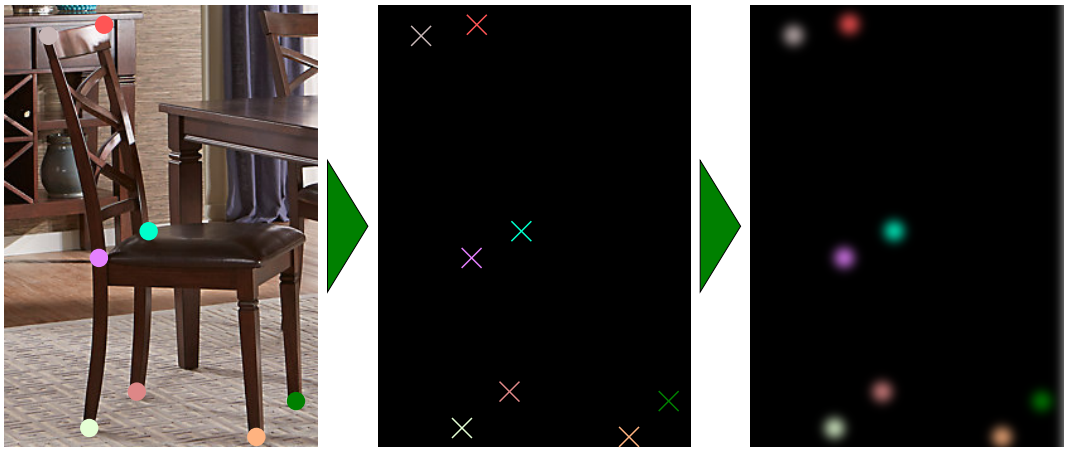


Figure 5.9: To facilitate the training process the keypoints are represented as Gaussian lobes around their location.

scape, destabilizing the training process. Instead, we represent each keypoint using a Gaussian lobe centered around its true location, resulting in a much smoother loss function (see Figure 5.9).

5.3.2.2 Keypoint estimation network

To extract keypoint location maps for each keypoint type from an input image, we employ a deep learning architecture. This network takes our image $\boldsymbol{x}$ as input and outputs a set of keypoint location maps $m_1, \ldots, m_{N_k}$, where $N_k = 6$ is the total number of predefined semantic keypoints.

The network architecture was selected through experimentation. We tried 2 different architectures:

- The convolutional pose machines (CPM) [124] architecture, whose task of human pose estimation through keypoint localization closely resembles our own, and
- ResNet-50 [46], a general purpose network with high performance on a number of image understanding tasks, such as object detection and semantic segmentation.

In both cases, we trained the network using an L2 loss function on the difference between the output and ground truth keypoint location maps.

Table 5.1: Performance of the two tried architectures on our task. ResNet-50's advantage of being pretrained on ImageNet gives it the edge over CPM.

architecture	MSE
ResNet-50 [46]	3.24×10^{-5}
CPM [124]	1.02×10^{-4}

Perhaps surprisingly, ResNet-50 resulted in the highest test accuracy (see Table 5.1). Although the task that CPM was meant for (keypoint detection) more closely resembles our own, it cannot compete with the fact that ResNet-50 was pretrained on ImageNet, the data distribution of which is more similar to ours.

We employed the TensorFlow implementation of ResNet-50. By using an input image size of 512×512 and a bottleneck stride of 8 we get a final keypoint map size of $r = 64$. The full architecture can be seen in Table 5.2. The training data we used is discussed in Section 5.3.8.

Table 5.2: ResNet-50 based architecture used for keypoint estimation.

layer name	output size	node type
input	512×512	
conv_1	256×256	7×7, stride 2
max_pool	128×128	Max pooling, stride 2
block_1	64×64	Bottleneck units with shortcuts, $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$, last 3×3 stride 2
block_2	64×64	Bottleneck units with shortcuts, $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$, all stride 1
block_3	64×64	Bottleneck units with shortcuts, $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$, all stride 1
block_4	64×64	Bottleneck units with shortcuts, $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$, all stride 1

5.3.3 Candidate generation

Now that the camera parameters and keypoint locations have been estimated, we move on to the candidate generation stage. In this part, predefined object templates

are fit to different subsets of the estimated keypoint locations, and scored by their agreement with the entire keypoint map. First, we will describe how we get specific keypoint locations from the estimated keypoint maps. Then, we will discuss how we construct the object templates from our set of 3D models. Finally, we describe the actual candidate generation process.

5.3.3.1 Keypoint locations from keypoint location maps

The keypoint estimation network's output consists of N_k single channel keypoint location maps $m_1, \ldots, m_{N_k}$. For our candidate generation process, these maps need to be converted to concrete keypoint locations. We cannot simply take all locations with a value above a certain threshold, as the maps spread the probability of a found keypoint across multiple pixels (Figure 5.9). One way of dealing with this is to find all local maxima in each map. The issue with this is that large regions of very low probability still have many local maxima. To discount these, we first pass each map m_i through a thresholding operation with threshold τ_m, discarding all pixels below that value. Then, we find all 8-neighbourhood local maxima in each map m_i, and store them as our candidate keypoint locations. We denote the found keypoint locations of type k as $\boldsymbol{Q}_k$, and the full set $\boldsymbol{Q} = \{\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{N_k}\}$. See Figure 5.10.

Figure 5.10: Keypoint candidate locations are found by thresholding the output of the neural network and then finding local maxima

5.3.3.2 Object templates

From the keypoint candidates $\boldsymbol{Q}$, we want to find actual chair candidates. As all chairs are slightly different in shape, and fitting each chair model in our dataset individually is prohibitively expensive, we make use of a chair template model.

Specifically, we create this chair template model by fitting a Principal Com-

ponent Analysis (PCA) basis to the 3D coordinates of all 8 keypoints of all chair models in our database $\boldsymbol{M}$. By analysing the cumulative percentage of variance of each resulting PCA dimension, we conclude that the top 3 PCA dimensions are responsible for $> 85\%$ of variance in the shape of all chairs. These top 3 PCA dimensions represent our chair template model T, and the deviation from the mean $\boldsymbol{p} \in \mathbb{R}^3$ represents a variable for our optimization. See Figure 5.11.

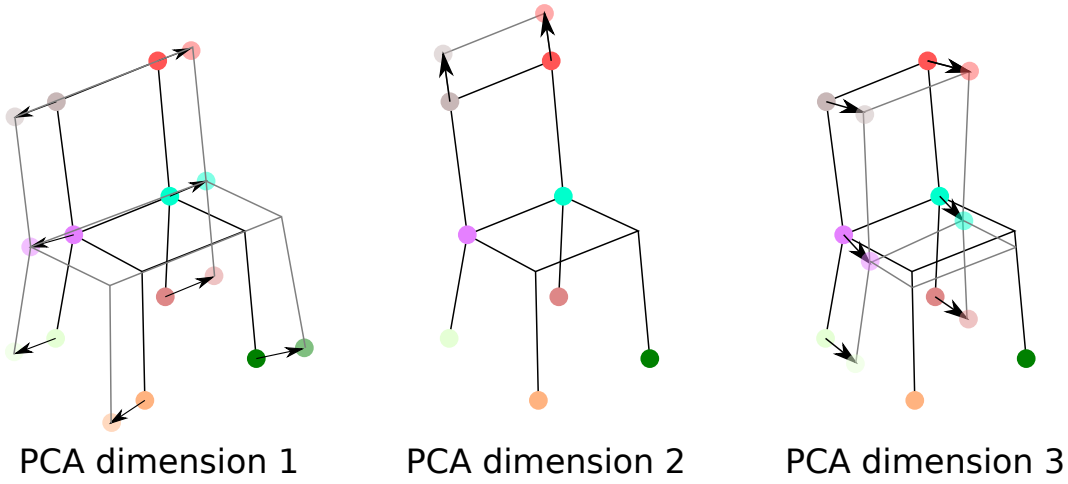


Figure 5.11: Visualization of the top 3 PCA dimensions of our chair template, with respect to the mean chair. They approximately correspond to respectively chair width, back height and chair depth.

We define $T(\boldsymbol{p})$ as the reprojection of PCA parameters $\boldsymbol{p}$ to 3D world space, i.e. the instantiated coordinates of one particular instance of the chair template.

5.3.3.3 Candidate keypoint sets

Finally, we will fit the generated chair template T to the found keypoint locations $\boldsymbol{Q}$. Unfortunately, we do not have any correspondences between the keypoint locations of different types – for example, we do not know which "top-left" keypoint belongs with which "front-right-leg" keypoint. As such, we generate the exhaustive set of candidates by fitting a candidate chair placement to each minimum set of 2D keypoint locations that results in a well-defined fitting problem. A single keypoint correspondence is not enough, as any candidate placement can then be rotated around its up-axis indiscriminately. As we know the camera and thus the ground plane, and work under the assumption that the chair models can change only scale

and azimuth (i.e. are placed flat on the ground), we can suffice with 2 keypoint correspondences. Although this does leave some ambiguities due to overlap between the scale dimension and the template parameters, due to regularization on both of these parameter sets the resulting problem is well-defined. We thus create our set of 2D keypoint candidate pairs as

$$\boldsymbol{K} = \bigcup_{\boldsymbol{Q}_i \in \boldsymbol{Q}} \bigcup_{\boldsymbol{Q}_j \in \boldsymbol{Q} \setminus \boldsymbol{Q}_i} \boldsymbol{Q}_i \times \boldsymbol{Q}_j,$$

where $\times$ represents the Cartesian product.

5.3.3.4 Template fitting

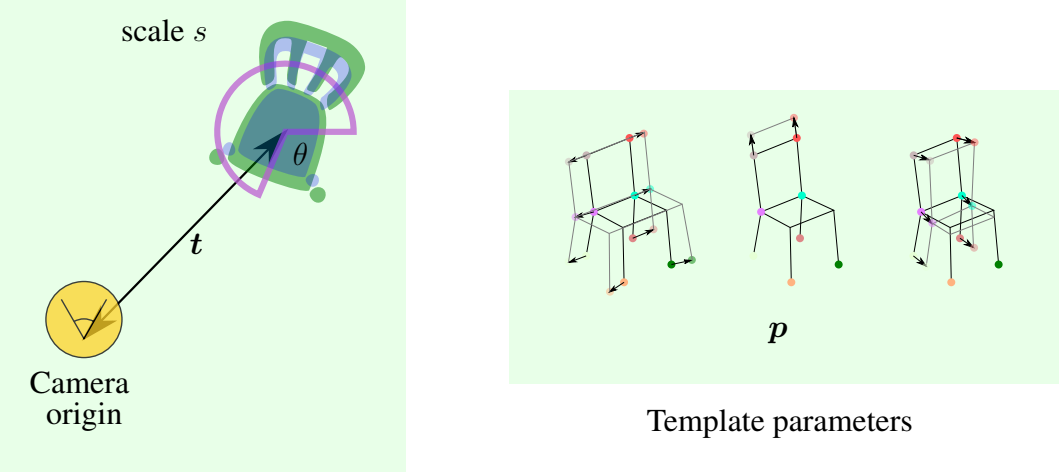


Figure 5.12: Parameters estimated during the candidate fitting process.

Then, we will generate one candidate chair placement for each $\boldsymbol{K}_i \in \boldsymbol{K}$ by finding the optimal parameters that yield a reprojection of the template's keypoints in line with $\boldsymbol{K}_i$, as well as the full keypoint location maps $\boldsymbol{m}$. These parameters consist of:

- a 2D translation across the ground plane $\boldsymbol{t}$,
- 1D azimuth θ,
- 1D scale s,
- 3D chair template parameters $\boldsymbol{p}$.

See Figure 5.12 for clarification. This optimization is split into two stages. In the first stage, we will optimize specifically for the reprojection of the 3D keypoints corresponding to $k_u, k_v \in \boldsymbol{K}_i$. In the second stage, we will incorporate our knowledge of the other keypoint location maps in $\boldsymbol{m}$ and further finetune the parameters to match with them as closely as possible as well. We now describe each stage in turn.

First stage – optimization w.r.t. 2 keypoints In the first stage, we find the optimal parameters such that the reprojection of the chair template's keypoints line up with $\boldsymbol{K}_i$. We define the reprojection z_i of each keypoint $k_i, i \in \{u, v\}$ as

$$z_i = P(R(s[T(\boldsymbol{t})]_i, \theta) + \boldsymbol{t}, C),$$

where R represents rotation, and P represents camera projection.

The objective function is then simply the summed mean squared error of these reprojections w.r.t. the data:

$$L = \sum_{i \in \{u,v\}} ||z_i - k_i||^2$$

We initialize the parameters as $\boldsymbol{t} = \boldsymbol{0}, \theta = 0, s = 1, \boldsymbol{p} = \boldsymbol{0}$. Furthermore, we add an L2 regularization term to both the norm of the template parameters $\boldsymbol{p}$ as well as the scale s. This non-linear least squares optimization problem is then solved using Ceres [3].

Second stage – optimization w.r.t. all keypoints Now that the parameters have been optimized w.r.t. our keypoint pair $\boldsymbol{K_i}$, we finetune the parameters by also taking into account the other keypoint location maps in $\boldsymbol{m}$. Note that we now go back to using the keypoint location maps themselves instead of the extracted local maxima – we do not optimize for exact location anymore, and allow the final reprojection to deviate from the maxima in each individual keypoint location map. Instead, we maximize the *total probability* over all keypoint location maps. Our

objective function thus becomes:

$$L = \sum_{i \in \{1,\dots,N_k\}} ||1 - m_i(z_i)||^2,$$

where $m_i(z_i)$ represents the value of keypoint location map m_i at reprojected keypoint z_i. The same L2 regularizations as in the first stage apply, and we again solve our problem using Ceres [3].

If the final loss of the second stage is lower than a threshold τ_u we add the final parameters as a candidate placement to our candidate placement set $\boldsymbol{O}$. This candidate placement set is then passed on to the candidate selection stage.

5.3.4 Candidate selection

In the final stage of our pipeline, we incorporate the key insight of this method, as discussed in the introduction, which states that we need to use higher level scene statistics to maximize our mockup performance. Specifically, we take the candidate placements $\boldsymbol{O}$ from the previous stage and employ a combination of the keypoint location maps and a model of object co-occurrence statistics to select the final subset of chairs that constitutes our scene mockup.

5.3.4.1 Scene statistics

To model these higher level scene statistics, we employ a pairwise object co-occurrence model. It models the probability of two chairs occurring at a given relative orientation and translation from each other. To create this model, we fit a Gaussian Mixture Model over the relative orientation δ_θ and translation $\boldsymbol{\delta}_t$ of pairs of chairs in the synthetic scene dataset PBRS (see Section 5.3.8). We only take into account chairs that are within a distance $\delta_r = 1.5m$ from each other, reasoning that chairs that are farther apart are more likely to belong to entirely different groups of chairs, making it imprudent to base our reconstruction on their relationship. See Figure 5.13 for clarification.

Fitting the GMM was done using Expectation-Maximization. As the models in PBRS tend to be aligned exactly, we regularize the resulting mixture model by adding a small bias (0.01) to the diagonal of the fitted covariance matrices. The

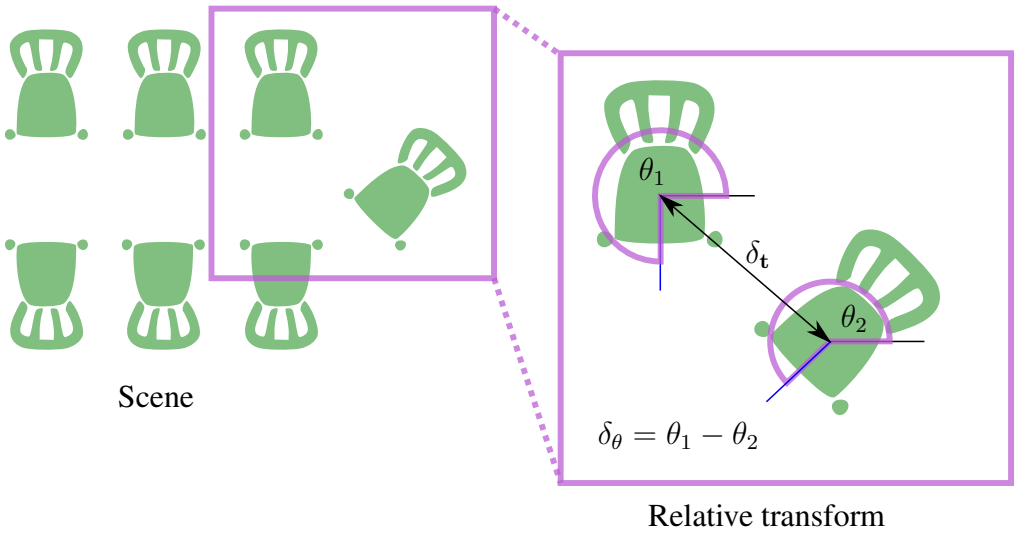


Figure 5.13: We extract relative transformations of pairs of chairs from the PBRS dataset and fit a GMM to these datapoints.

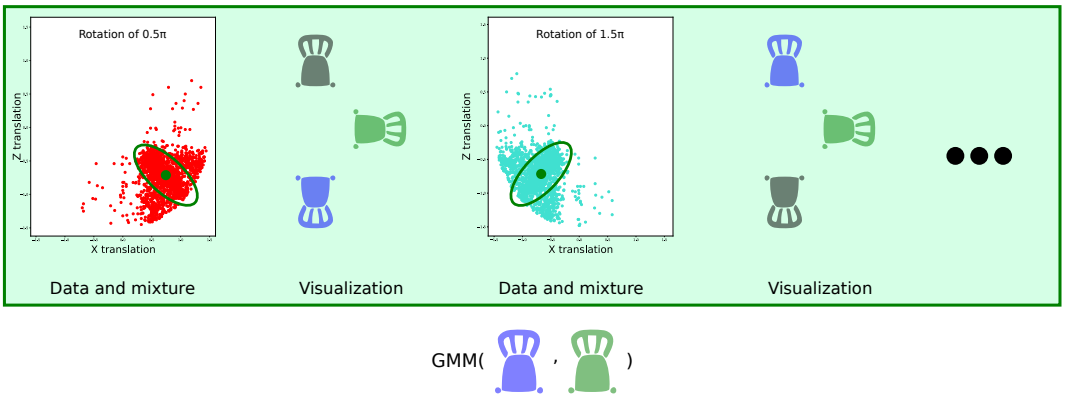


Figure 5.14: A visualization of two of the mixture components resulting from fitting the GMM to the relative transformations of pairs of chairs in the PBRS dataset. The means and standard deviational ellipses are plotted in green.

number of mixture components N_m was found by experimentation, and was set to 5. A visualization of some of the resulting mixture components can be found in Figure 5.14.

5.3.4.2 Graph optimization

We now need to prune our over-complete set of candidate placements using the trained object co-occurrence model. We represent this task as a graph labeling problem. Each candidate placement represents a node in the graph, and takes on a binary label representing whether or not that candidate placement is present in the

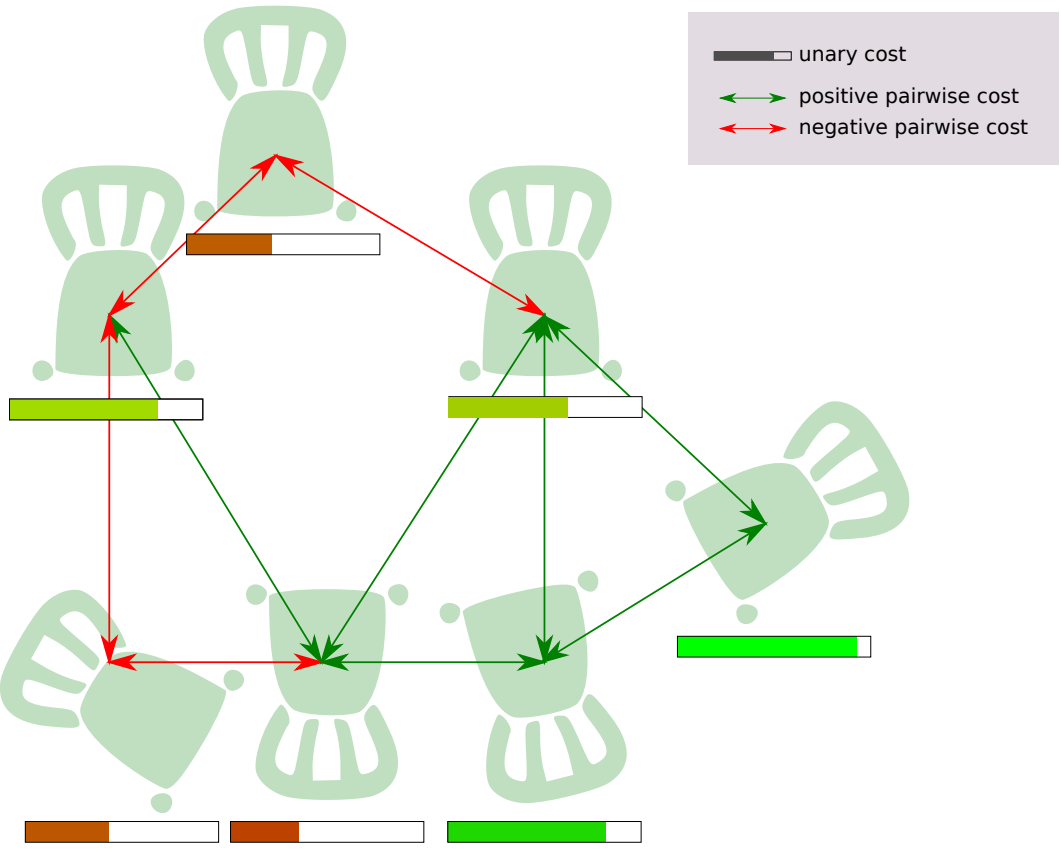


Figure 5.15: We model our candidate selection problem as a graph labeling problem, where the unary costs are based on the keypoint location maps, and the pairwise costs on the scene statistics GMM.

final mockup. Unary costs for each label stem from the keypoint location maps, and pairwise costs stem from the scene statistics GMM. See Figure 5.15.

Unary cost To compute the unary score of a candidate placement $o_i \in \boldsymbol{O}$, we *generate* the keypoint location map $\boldsymbol{n}$ of o_i (in the same way we would do for creating a ground truth keypoint map) and compare it with the keypoint location map $\boldsymbol{m}$ of the input image $\boldsymbol{x}$. As we do not expect a single placement to explain the entire keypoint location map, we setup the score as a multiplicative one, with the value only being dependent on the agreement of the actual keypoints the placement o_i exhibits:

$$u_i = \frac{||\boldsymbol{n} \odot \boldsymbol{m}||_F}{\boldsymbol{n} \odot \boldsymbol{n}},$$

where $||\cdot||_F$ represents the Frobenius norm, and $\odot$ represents the Hadamard

product.

The normalization factor ensures that a candidate that perfectly matches the keypoint location map of our input image $\boldsymbol{x}$ gets a score of 1. Finally, for a specific candidate $o_i \in \boldsymbol{O}$, interpreting u_i as a probability we get unary costs based on the log odds of u_i:

$$U_i(0) = 0 \tag{5.1}$$

$$U_i(1) = -\log(\frac{u_i^\alpha}{1 - u_i^\alpha}) \tag{5.2}$$

where α is a scaling parameter to set the sensitivity of optimization to the value in the keypoint maps. Our choice for the log odds means that a (scaled) score of higher than 0.5 results in a candidate unary cost that *decreases* the score of the total cost when selected, and otherwise *increases* it.

Pairwise cost The pairwise cost is based entirely on the fitted GMM. We extract the relative translation $\boldsymbol{\delta}_t$ and orientation δ_θ, and evaluate the trained GMM to get our raw pairwise score:

$$p_{ij} = GMM(o_i, o_j)$$

The final pairwise score is then again based on the log odds corresponding to p_{ij}. It only applies when two objects co-occur:

$$P_{ij}(0,0) = P_{ij}(1,0) = P_{ij}(0,1) = 0 \tag{5.3}$$

$$P_{ij}(1,1) = -\log(\frac{p_{ij}^\beta}{1 - p_{ij}^\beta}) \tag{5.4}$$

with β a scaling parameter similar to α.

Finally, we add an infinite pairwise cost to all candidate placement pairs that intersect. These intersections are precomputed based on triangle-triangle intersections.

We solve the final problem setup using OpenGM [5] by converting it to a linear program and feeding it to CPLEX [2].

5.3.5 Iterative optimization

After the optimization from Section 5.3.4 is complete, we could stop and pass on the candidate placements with label 1 to the model selection stage (Section 5.3.6). However, now that some objects have been definitely placed, we can use this information to improve our candidate generation step, and by extension our candidate selection step. In other words, we iterate the process of candidate generation and selection, using the newly selected candidates in each iteration as a strong prior for the candidate generation process of the next generation.

5.3.5.1 Added pairwise cost in generation step

To take into account the already selected placements during the candidate generation phase, we keep our original non-linear least squares optimization, but to the loss function of each stage of the two stage process (see Section 5.3.3.4) we add a term that represents the GMM. Incorporating all mixture components in this term is hard, as it is challenging to define a well-behaved objective function to minimize that represents them. As noted by Olson et al. [86], the structure of the negative log-likelihood (NLL) of a GMM does not lend itself to non-linear least squares optimization. Instead, they propose to approximate the NLL of the full GMM by considering it as a Max-Mixture, reducing the NLL to the weighted distance to the closest mixture mean (see Figure 5.16 and [86] for details). In fact, in our case it makes sense to only optimize with respect to the closest mean, and not all means: a chair should either be encouraged to be next to another chair, or opposite, but never both. This replaces the original GMM likelihood function

$$p_{\mathrm{GMM}}(\boldsymbol{\delta}) = \sum_i w_i N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

with the Max-Mixture likelihood function

$$p_{\mathrm{Max}}(\boldsymbol{\delta}) = \max_i w_i N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where $\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_t \\ \delta_\theta \end{bmatrix}$ is the relative translation and orientation of the new candidate

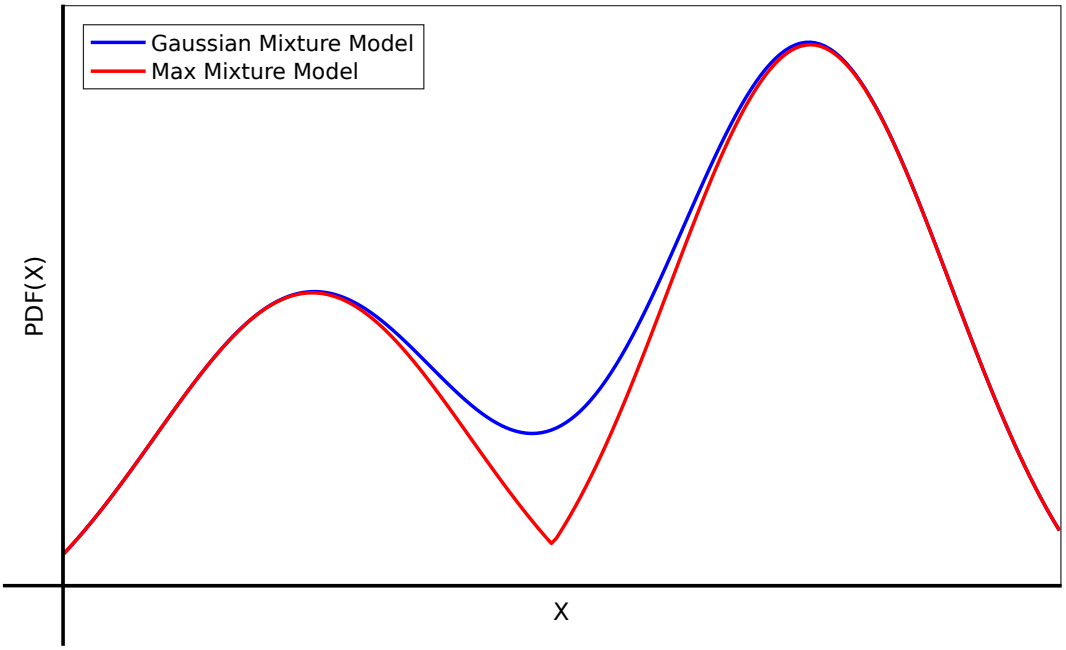


Figure 5.16: We approximate the GMM using a Max-Mixture Model from Olson et al., 2013 [86]. Due to the simplified negative log likelihood of this model we can then use it in our non-linear least squares optimization.

w.r.t. the already placed object, and w_k is the weight of the kth mixture in the model.

Taking the negative log likelihood gives

$$-\log(p_{\text{Max}}(\boldsymbol{\delta})) = \min_k \frac{1}{2}(\boldsymbol{\delta} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\delta} - \boldsymbol{\mu}_k) - \log(w_k \eta_k),$$

where $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the normal distribution, and η_k is the Gaussian normalization factor for the kth mixture. At optimization time, during each step we find the mixture component k^* that minimizes this function, and then optimize w.r.t. the negative log likelihood of the Gaussian of that component alone, resulting in the following term to be added to the objective function:

$$\frac{1}{2}(\boldsymbol{\delta} - \boldsymbol{\mu}_{k^*})^T \boldsymbol{\Sigma}_{k^*}^{-1}(\boldsymbol{\delta} - \boldsymbol{\mu}_{k^*})$$

By decoupling the component selection from the optimization step, we've restored the nice properties of the single Gaussian negative log likelihood. This term is added for each already placed object.

5.3.5.2 Added unary cost in selection step

As the already selected placements are not part of the optimization during later iterations, the influence of the GMM on a new candidate placement w.r.t. already selected placements becomes a unary cost. So, for each candidate placement in the second iteration, we add a term to $U_i(1)$ w.r.t. each of the already selected placements:

$$-\log(\frac{GMM(o_i, o_j^*)^\beta}{1 - GMM(o_i, o_j^*)^\beta})$$

With these modifications, the candidate generation step and candidate selection step are iterated until convergence, i.e. until no new objects are added to the scene.

5.3.6 Model selection

The set of all selected placements still only consist of template parameters, not actual chair models. As a final step, we find the chair g^* in our database $\boldsymbol{M}$ that best fits the template. To do so, we reproject the 3D keypoint coordinates of each chair in the database to the PCA coordinate space, and find the chair whose PCA coordinates are closest to the PCA coordinates of our template:

$$g^* = \arg\min_{g \in \boldsymbol{M}} ||[\mathrm{PCA(g)}]_0^3 - \boldsymbol{p}||^2,$$

where $\boldsymbol{p}$ are the PCA coordinates of the candidate's template.

The resulting chair models together with their transform constitute our final scene mockup.

5.3.7 Hyper parameters

Our optimization pipeline depends on a number of hyper parameters. We optimized these using HyperOpt [56], which employs a Tree of Parzen Estimators (Bergstra et al., 2013 [11]). As our objective function we used the PercCorrectFull measure (see Section 5.4.2). As ground truth data we used 10 scenes we annotated specifically for this purpose, in the same way as the data used for evaluation (see Section 5.4.1). See Table 5.3 for a list of resulting hyper parameter values.

Table 5.3: Hyper parameters of optimization, found by HyperOpt [56]

Name	Description	Value
α	Sensitivity of keypoint maps	0.61
β	Sensitivity to object co-occurrence model	0.14
τ_m	Lower threshold of keypoint location map	0.25
τ_u	Maximum cost for selecting candidate	0.21

Figure 5.17: Example images from our scraped HOUZZ dataset.

5.3.8 Data

5.3.8.1 Image data

For purposes of qualitative evaluation, we scraped the interior design website [1] for the top 1000 results of the search query "dining room". We denote this dataset HOUZZ. These images are high quality and represent difficult but fair scenarios on which we expect our method to perform well. Some examples of these images can be seen in Figure 5.17.

5.3.8.2 Network training data

Traditionally, training a deep neural network requires a large amount of training data. To our knowledge, there is no known large dataset of photographs accurately annotated with object keypoints. As such, we resort to creating our own training data. Ideally, the training data should be from the same distribution as our intended testing data, i.e. photographs of indoor scenes. However, creating a large-scale dataset of this type is extremely time-consuming and expensive. On the other hand, synthetic data in the form of realistic 3D indoor scenes along with physically-based renders is already available in high numbers [138]. Still, despite the high quality of the renders, there is still a significant discrepancy between the feature distribution of the renders and that of the photographs. As such, we augment the synthetic

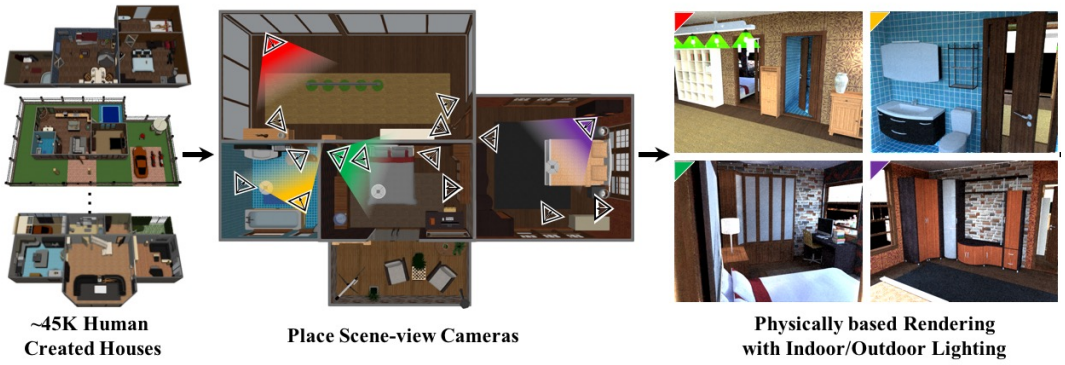


Figure 5.18: For the training setup of our network with synthetic data, we use renders from the PBRS dataset [138], which provides ±45K houses with ±400K high quality renders. Figure from [138].

dataset with a subset of real photographs from HOUZZ annotated through Amazon Mechanical Turk. We now discuss each data type in turn.

Synthetic data The dataset provided by Zhang et al. [138] provides 45K realistic indoor scenes, and 400K physically-based renders of these scenes (see Figure 5.18). We denote this dataset as PBRS. These scenes consist of a fixed set of 2500 different models across 60 classes. Among these models there are ± 250 chairs. We took a subset of 100 of these chairs and annotated them with our previously selected keypoint types. We then took all renders that contain at least 1 of the annotated chairs and reprojected the keypoint locations into these renders, yielding one image/keypoint map pair as training data per render. This resulted in a set of ± 8000 image/keypoint map pairs in total.

Real data Unfortunately, the synthetic data alone does not result in good performance on real data. Two distinct reasons can be identified. First, even though the renders in PBRS are of high quality, their feature distribution is both distinct from real photographs as well as less diverse. Secondly, at the time of writing, the set of renders and the set of scenes available for PBRS had some discrepancies between them, resulting in a small but significant set of renders that do not agree with the automatically generated keypoint maps.

To address both of these issues, we annotated a subset of 500 images from the HOUZZ dataset through Amazon Mechanical Turk. We asked 3 workers per

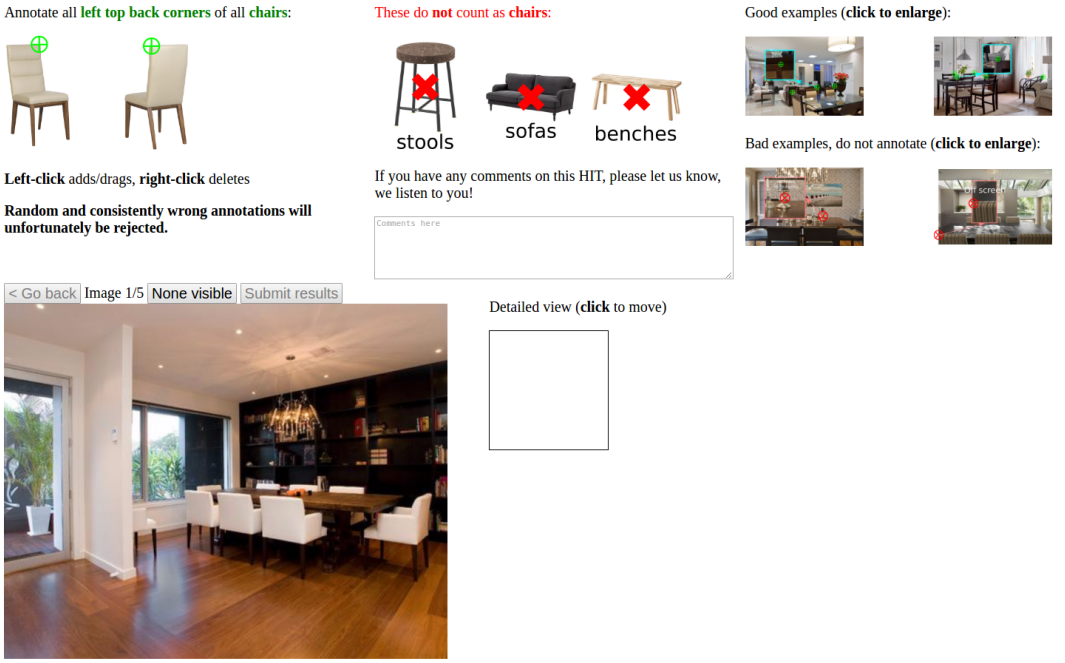


Figure 5.19: The Amazon MTurk interface we used to annotate 500 photographs with keypoints.

image to annotate all keypoints in the image through a drag-and-drop interface (see Figure 5.19), and averaged the resulting 3 keypoint maps per image. This resulted in a training set of 500 hand-annotated photographs, which was then used to train our keypoint estimation network.

Final training set We experimented with 3 different training setups. In the first setup, we trained the network only with synthetic data. In the second setup, we only trained the network with real data. Finally, in the third setup, we first trained the network until convergence with the synthetic data, and then finetuned the network using the smaller set of real data.

Surprisingly, the best performance on the test set resulted from setup 2, i.e. training only with the real data. Apparently, the shortcomings of the synthetic data mentioned above were of higher importance than expected. One likely explanation is the fact that training the network with the synthetic data first steers away the network weights from those that were the result of the ImageNet pretraining, which already encompass a high general understanding of real photographs. The numbers show that this initial information is more valuable than the extent of the synthetic

data as well as its structural similarity to our test data.

5.3.8.3 Model data

The models annotated for the purpose of generating synthetic network training data also immediately function as our model set $\boldsymbol{M}$.

5.4 Evaluation

We thoroughly evaluated our method, investigating the importance of each part of our pipeline as well as comparing our results with other methods. We will first discuss the creation of a set of ground truth annotated scenes for the purpose of quantitative evaluation (Section 5.4.1). We then define a set of diverse performance measures (Section 5.4.2), after which we introduce two baseline methods for comparison purposes (Section 5.4.3). We evaluate our method with the ground truth set, and compare the numbers with two distinct baseline methods (Section 5.4.4). Finally, we perform an ablation study to show the influence of each on the final performance (Section 5.4.5). Both quantitative and qualitative results will be shown along the way.

5.4.1 Ground truth annotation

In order to quantitatively measure the performance of both the baseline methods and our own, we need a set of ground truth annotated scenes, i.e. images for which all objects have been placed manually. We setup an application in which an object can be placed by clicking and dragging, as well as by annotating a number of keypoints of the object and optimizing for its location and scale. Moreover, objects can be copied and translated along their local coordinate axes, allowing for quick and precise annotation (see Figure 5.20). We use the automatically estimated camera parameters, making sure we discard any scenes for which the camera estimation is completely off. We used this tool to fully annotate 100 scenes, which were randomly selected from our HOUZZ dataset of 1000 images.

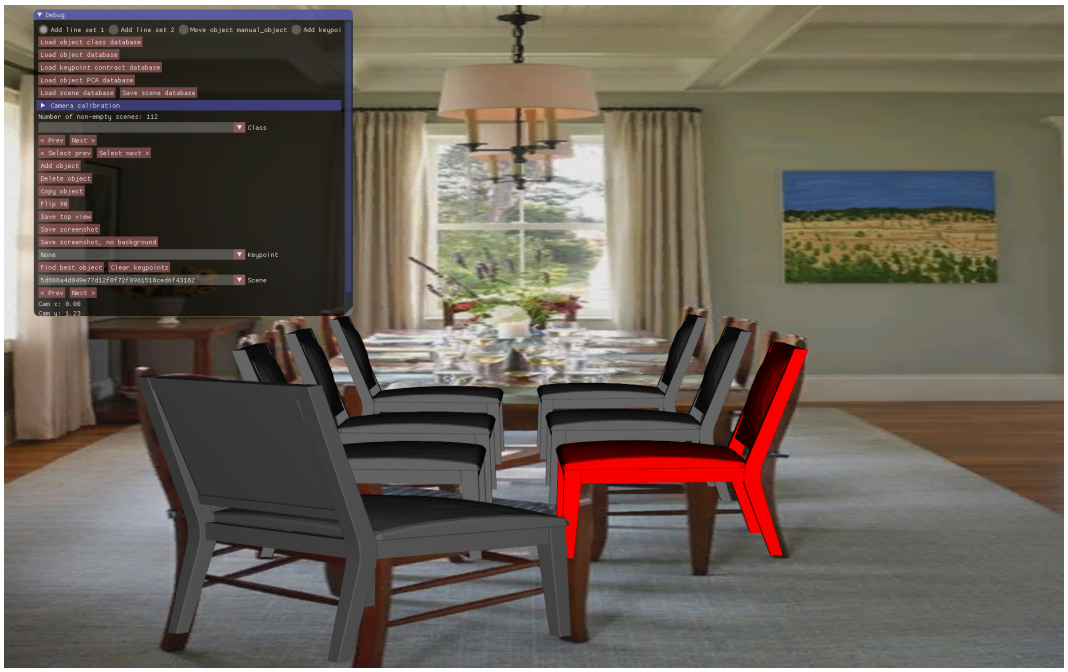


Figure 5.20: We created a ground truth annotation tool for quickly creating ground truth scene mockup examples.

5.4.2 Performance measures

A scene mockup method can be quantitatively evaluated in many different ways. As no single measure tells the full story, we have opted for a number of different ones.

Notation We will use the concept of "source" and "target" to denote the two scenes between which some measure is computed. We specifically do not use "result scene" and "ground truth scene", because they can act as either source or target scene in most measures. We denote the objects in the source and target scene as $o_S \in \boldsymbol{S}$, $o_T \in \boldsymbol{T}$ respectively. $J_3(o_S, o_T)$ and $J_2(o_S, o_T)$ represent the Jaccard index or *intersection-over-union* (IoU) of the bounding boxes of o_S and o_T in 3D world space and 2D screen space respectively (see Figure 5.21). Finally, given an object o_S we define the "J_i^* correspondence" with $\boldsymbol{T}$ as the object in T with the maximum Jaccard index with o_S:

$$J_i^*(o_S, \boldsymbol{T}) = \arg\max_{o_T \in \boldsymbol{T}} J_i(o_S, o_T)$$

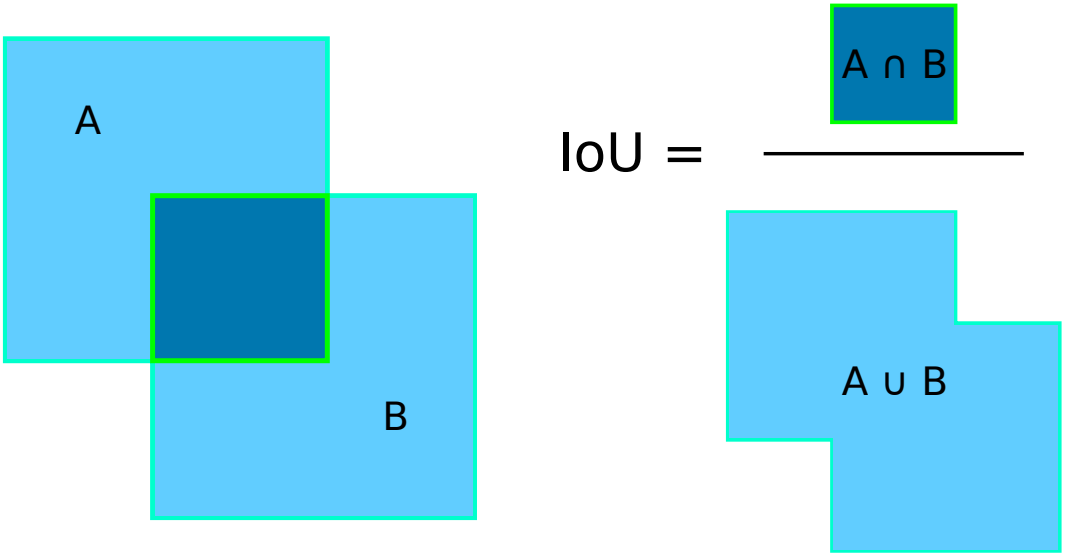


Figure 5.21: Visualization of the intersection-over-union measure in 2D.

Intuitively, this returns, for a given object, the "best matching" object from the other scene in terms of overlap.

Average Max IoU This measure takes a source scene and a target scene, and records the accuracy with which the volumes of the objects in the source scene agree with the objects in the target scene. Specifically, for each object in the source scene, we record the IoU of the object with its MaxIoU correspondence. This measure is averaged over all objects in the source scene to produce the final measure.

$$\text{AvgMaxIoU}(\boldsymbol{S}, T) = \frac{1}{|\boldsymbol{S}|} \sum_{o_S \in \boldsymbol{S}} J_3(o_S, J_3^*(o_S, \boldsymbol{T}))$$

We measure in both directions, i.e. with the ground truth as source and result as target, as well as vice versa. The former can be thought of as a form of "recall" and the latter as a form of "precision". This measure is angle-agnostic and captures the location similarity of objects in the source scene w.r.t. those in the target scene.

Percentage correct location This measure takes a source scene and a target scene, and records the percentage of objects in the source scene that have a J_3^* cor-

respondence over a certain threshold τ_J. To define it, we first set

$$\mathrm{CorrectLoc}(\boldsymbol{S}, \boldsymbol{T}) = \{o_S \in \boldsymbol{S} \mid J_3(o_S, J_3^*(o_S, \boldsymbol{T})) > \tau_J\}.$$

Then,

$$\mathrm{PercCorrectLoc}(\boldsymbol{S}, \boldsymbol{T}) = \frac{|\mathrm{CorrectLoc}(\boldsymbol{S}, \boldsymbol{T})|}{|S|}.$$

We again measure in both directions, yielding recall (ground truth is source, result is target) and precision (vice versa) measures.

Percentage correct As the previous measure, but with the added constraint that the angle difference is under a threshold τ_θ. So,

$$\mathrm{CorrectFull}(\boldsymbol{S}, \boldsymbol{T}) = \{o_S \in \mathrm{CorrectLoc}(\boldsymbol{S}, \boldsymbol{T}) \mid \angle(o_S, J_3^*(o_S, \boldsymbol{T})) < \tau_\theta\}.$$

Then,

$$\mathrm{PercCorrectFull}(\boldsymbol{S}, \boldsymbol{T}) = \frac{|\mathrm{CorrectFull}(\boldsymbol{S}, \boldsymbol{T})|}{|\boldsymbol{S}|}.$$

Angle difference This measures the average angle difference for the objects that have correct location. This measure is symmetrical.

$$\mathrm{AngleDiff}(\boldsymbol{S}, \boldsymbol{T}) = \frac{1}{|\mathrm{CorrectLoc}(\boldsymbol{S}, \boldsymbol{T})|} \sum_{o_S \in \mathrm{CorrectLoc}(\boldsymbol{S}, \boldsymbol{T})} \angle(o_S, J_3^*(o_S, \boldsymbol{T}))$$

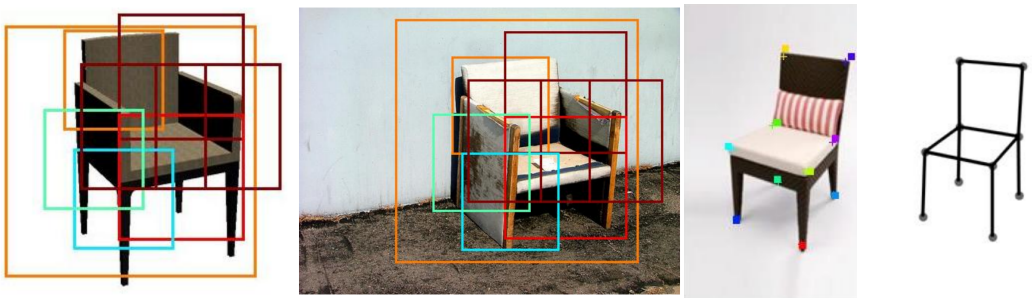
Average Max 2D IoU This measures the average maximum IoU of the bounding boxes of each projected object in the source scene with the bounding boxes of the projected objects in the target scene.

$$\mathrm{AvgMax2DIoU}(\boldsymbol{S}, \boldsymbol{T}) = \frac{1}{|\boldsymbol{S}|} \sum_{o_S \in \boldsymbol{S}} J_2(o_S, J_2^*(o_S, \boldsymbol{T}))$$

5.4.3 Baseline methods

We compare our method with two baselines from the literature. As the exact problem formulation we employ has to our knowledge not been attempted, we convert

the output of each baseline (in both cases 3D pose but 2D, image space locations of chairs) to the 3D scene mockup format that our method produces.



Seeing Chairs [6] 3D-INN [128]

Figure 5.22: Example of raw output of the two baseline methods.

Seeing chairs [6] This method from Aubry et al. finds chairs by matching so-called "discriminative visual elements" or DVEs from a set of rendered views of 1000+ chair models with the input image. These DVEs are linear classifiers over HOG features [25] learnt from the rendered views in a discriminative fashion. They are learned at multiple scales, and only the most discriminative ones are kept for matching purposes. At test time, a patch-wise matching process finds the best-matching image patch/rendered patch pairs, and then finds sets of pairs that come from the same rendered view (see Aubry et al.'s paper for details [6]).

This method outputs scored image space bounding boxes together with a specific chair model and pose. See Figure 5.22, left. For the 3D performance measures (Section 5.4.2) we need the output in the form of a 3D scene. To this end we convert each set of bounding box, pose, and chair model to a 3D scene. As the camera is known (Section 5.3.1), we can optimize the location (in the X-Z plane) of the 3D model without changing its pose, such that the 2D bounding box of the projected model matches as closely as possible with the detected bounding box. This can be formulated as a least-squares optimization problem, which we solve using Ceres [3].

FasterRCNN [94] + 3D-INN [128] This baseline is a combination of a convolutional neural network (CNN) trained for object detection (FasterRCNN) and another CNN trained for 3D object interpretation (3D-INN). We use FasterRCNN to

extract bounding boxes of chairs from the input image, and then feed these regions of interest to 3D-INN, which produces a templated chair model consisting of a set of predefined 3D keypoints as well as a pose estimate (azimuth and elevation). See Figure 5.22, right. The set of keypoint types we have chosen for our method is a subset of the keypoints produced by 3D-INN, and thus we can use the candidate generation part of our pipeline (see Section 5.3.3) to convert the extracted keypoints to a 3D chair.

5.4.4 Comparison

Table 5.4: Quantitative performance of our method vs. the two baseline methods. We outperform the baseline significantly across all measures.

	AvgMaxIoU (precision)	AvgMaxIoU (recall)	AvgMaxIoU (F1)	
3D-INN [128] + FasterRCNN [94]	0.316	0.150	0.198	
SeeingChairs [6]	0.195	0.128	0.149	
Ours	**0.386**	**0.250**	**0.293**	
	PercCorrect (precision)	PercCorrect (recall)	PercCorrect (F1)	
3D-INN [128] + FasterRCNN [94]	0.263	0.124	0.165	
SeeingChairs [6]	0.071	0.043	0.052	
Ours	**0.298**	**0.167**	**0.207**	
	PercCorrectFull (precision)	PercCorrectFull (recall)	PercCorrectFull (F1)	
3D-INN [128] + FasterRCNN [94]	0.04	0.015	0.021	
SeeingChairs [6]	0.013	0.007	0.009	
Ours	**0.285**	**0.161**	**0.198**	
	AvgMax2DIoU (precision)	AvgMax2DIoU (recall)	AvgMax2DIoU (F1)	AngleDiff (in degrees)
3D-INN [128] + FasterRCNN [94]	0.526	0.336	0.401	55.8
SeeingChairs [6]	0.372	0.325	0.341	11.4
Ours	**0.628**	**0.470**	**0.525**	**7.3**

We ran our pipeline and the two baseline methods on the full ground truth annotated scene set (Section 5.4.1). A sampling of results can be seen in Figure 5.23. The same visualization for all 100 scenes in our ground truth set can be found in Appendix B.

The baseline methods perform well when there is no occlusion in the scene. Chairs that are clearly visible are reconstructed reliably, as the visual information directly available is enough for these methods to make a reasonable inference about the object's pose and identity. However, when a chair is partly occluded, these methods break down quickly. In contrast, our method is more often able to recover from these situations, due to the incorporation of the object co-occurrence model.

This difference in performance is also reflected in the quantitative results. We extracted the performance measures listed in Section 5.4.2 from each method, and list them in Table 5.4. Our method outperforms the baselines on all counts. More-

over, in Figure 5.24 we show how the PercCorrectFull measure changes under varying thresholds of IoU and angle (see Section 5.4.2).

5.4.5 Ablation study

Finally, we evaluated the importance of each of our pipeline's optional steps to the final performance. Specifically, we ran our pipeline on the full test set under two weakening conditions. In the first condition, we disable all pairwise costs, and run the entire pipeline based solely on the keypoint location maps. In the second

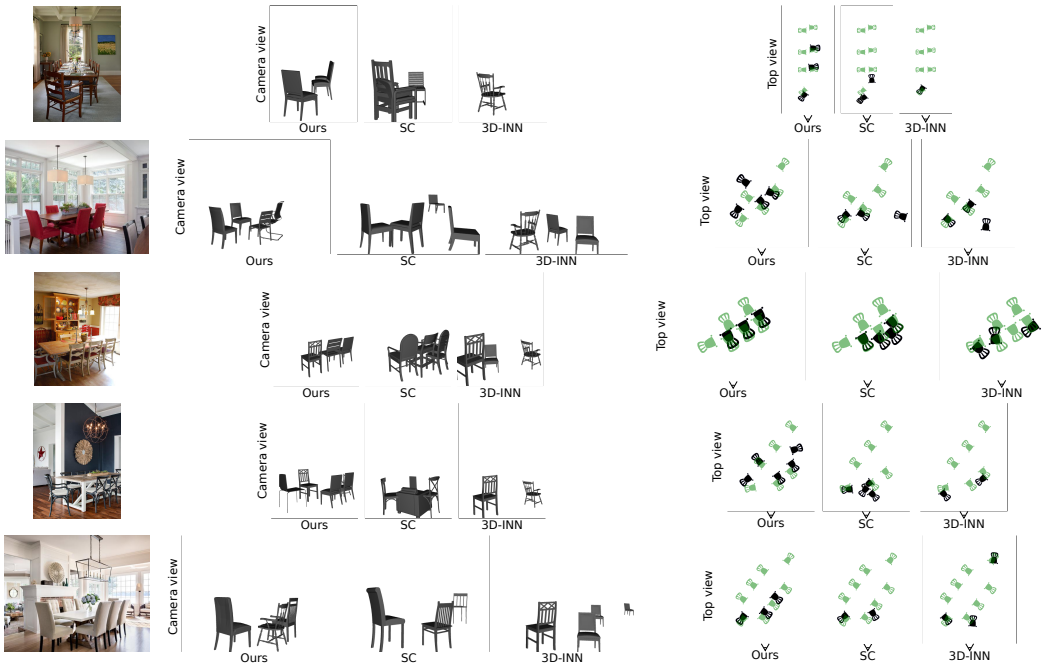
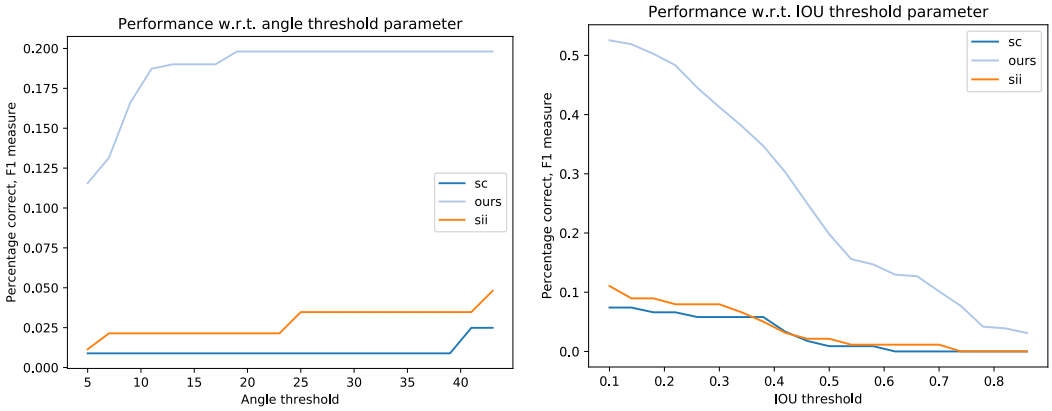


Figure 5.23: Qualitative results for our method vs. the baseline methods.



(a) Performance under varying τ_θ **(b)** Performance under varying τ_J

Figure 5.24: Changes in performance under varied angle and IoU thresholds.

Table 5.5: Ablation study showing the importance of using scene statistics and multiple iterations for best performance.

	AvgMaxIOU (precision)	AvgMaxIOU (recall)	AvgMaxIOU (F1)	PercCorrectFull (precision)	PercCorrectFull (recall)	PercCorrectFull (F1)
Full pipeline	0.386	0.250	**0.293**	0.285	**0.161**	**0.198**
No scene stats	0.296	**0.265**	0.267	0.174	0.151	0.154
Single iteration	**0.421**	0.190	0.251	**0.346**	0.123	0.175

condition, we only run the second and third stage once, removing the possibility of the candidate generation stage benefiting from previously placed objects. Results are found in Table 5.5.

There are some things to note. First, although AvgMaxIOU recall increases when disabling scene statistics, the precision goes down significantly. This makes sense, as the pairwise costs by themselves do not propose new objects – they only make output mockups more precise by pruning objects that do not agree with others. Second, using only a single iteration increases precision, but recall takes a significant hit. Again, this is logical, as in later iterations the keypoint location maps have decreased influence relative to the pairwise costs. This means that objects with weaker keypoint response get found more easily, but also that false positives are somewhat more likely. Overall, the combined AvgMaxIOU F1 measure is highest for the full pipeline, and perhaps most importantly the PercCorrectFull F1 measure as well.

5.5 Discussion

We proposed a method for automatically finding chairs in a photograph of a structured scene. Our key insight which gives us an advantage over other methods is the incorporation of higher level scene statistics, allowing us to reason more accurately about objects that are highly occluded. Through quantitative and qualitative evaluation, we have shown a considerable increase in performance across multiple measures. Nevertheless, some limitations of our method remain:

- Our method is currently only suited to chairs. However, this is not a limitation of the method, and with a proper data annotation effort it could be extended to arbitrary other classes. Note that adding more classes will likely improve the accuracy of finding chairs by themselves as well, as there will be more

scene information to draw from.

- The keypoint network is currently trained with 500 sample images. This is a very small set of data, and the performance of the network has clear room for improvement through the addition of more training data. However, gathering such data is expensive. Finding a better way to incorporate large amounts of synthetic training data into the pipeline is an interesting avenue for future work.

- After candidate selection, we do not reoptimize the position and orientation of each object. As we now have the added information of the location of the other objects, this could result in more accurate object placements.

- We do not explicitly model style. Although the use of the chair template does have some influence on the outer shape of the chair being used, there are many more properties that could be modelled for a more convincing mockup.

Chapter 6

Discussion and Future Work

In this thesis, three distinct methods regarding the analysis of uncontrolled visual data, or "in-the-wild" visual data, were discussed. Working our way from iconic images and 3D models to photographs of indoor scenes and their resulting 3D scene mockups, we showed a number of ways in which we can combine 2D and 3D information to beat single-modal methods. Through thorough evaluation of each method, we showed significant improvement over such baseline approaches on all accounts.

6.1 Summary

In Chapter 3 we looked at the problem of retrieving and exploring collections of 2D iconic images and 3D models. We started from the fact that typical 3D model collections and 2D iconic image collections have significantly different intrinsic properties – 2D collections are usually of much higher quality than 3D collections, while 3D collections implicitly provide viewing angle and shape information. Our insight was that these differences can be used to improve or enable tasks in each dimensionality by analyzing the collections jointly. Specifically, by exploiting the advantages of each dimensionality concurrently, we improved retrieval performance of the 3D collection, reducing the number of false positives, while enabling exploration through pose and shape of the 2D collection. Through quantitative and qualitative evaluation we demonstrated clear improvement of each stage of our pipeline over standard baseline methods.

Next, in Chapter 4, we shifted our focus to images of indoor scenes. Using a

deep neural network, we extract so-called *scene maps* from single images of indoor scenes – single channel maps that specify a top-down view of the location of different types of objects in scene coordinates. To offset the lack of existing training data, we incorporated a 3D scene generation and rendering step into the pipeline based on the clean model sets extracted in Chapter 3, supplying the training process with a virtually infinite amount of training data. On synthetic data, we showed significant improvements over baseline methods based on depth estimation and semantic segmentation. We also showed that under severe occlusion this method does not perform well, as in this case the visual data alone is too ambiguous to perform any reasonable inference.

In the final Chapter 5 we took this issue into account while expanding the objective into the full scene mockup problem: given a single image, find as many objects as possible in the image including their 3D location and pose, such that when reprojecting these objects into 2D with the separately estimated camera we get an image that's close to the input. We approached this in three stages, first extracting semantic keypoints using a deep neural network, then fitting a learned object template to all possible minimum sets of these keypoints, and finally pruning this candidate set using a learned object co-occurrence model. The use of keypoints and the object co-occurrence model allows us to make inferences about objects that are heavily occluded, by not requiring all keypoints to be visible for making an inference, and by relying on the contextual information provided by the pose and location of more clearly visible objects. By splitting up the reconstruction into a candidate generation and candidate selection phase, we kept the problem tractable, and by iterating these two phases we optimally made use of information as it became available. We compared our method with two separate baseline techniques and showed increased performance on all benchmarks, as well as the importance of each stage of our pipeline.

6.2 Future work

There are still many interesting avenues of research in cross-dimensional in-the-wild analysis that are left unexplored. Here we identify several key challenges that remain, and specify the long-term aspirations of this growing field.

An interesting addition to the problems tackled in Chapter 3 would be to not only sort and filter, but also to synthesize or edit properties in each dimensionality using information gleaned from the other. Indeed, we could imagine a system where the appearance and shape distributions learned over the "table" class assist the user in modifying a photograph of a 2D table in a geometrically meaningful way. In the other direction, the much richer material information of the 2D domain could be used for the automatic material assignment of the models in the 3D collection. Some work in this domain has already been done (i.e. Chen et al., 2015 [19]), but automating this process fully would constitute a valuable new addition to the 3D model creation toolkit.

Expanding the problem of joint 2D iconic image and 3D model retrieval and exploration, one interesting yet lofty goal is to unify their representation. A feature space in which both these cross-dimensional data types co-exist would allow for seamless exploration and retrieval of 2D images and 3D models, unifying all steps in Chapter 3 into one "basis-changing" operation. One could even imagine such a space to assist in the synthesis of new 3D models based on 2D images in this joint feature space, or vice versa.

In the context of indoor scene mockups, possible future research spans multiple orthogonal directions. As a first clear follow-up to Chapter 5 the pipeline can be extended to multiple classes. The resulting scene mockups will be richer and useful in themselves for multiple purposes, including game asset creation and virtual reality. Moreover, the accuracy of the method will increase with each added class, as ever more contextual information becomes available. Furthermore, it would be interesting to go beyond pose and location and infer other properties, such as object material, object style and scene lighting. Our hypothesis is that the added properties will again help each other: it is more than conceivable that the style of the chairs in

a room have some influence on the probability of the tables' materials. The statistical models that are needed to capture the relationships of these object properties could be useful for more than just this task – product designers and architects could look at these models directly to learn their audience's preferences, such as favored modes of room structure conditioned on object style.

On multiple occasions in this thesis we dealt with co-occurring synthetic images and real photographs. A reoccurring problem is the stark difference in feature space between the two types of images. An exciting and extremely useful research direction would be the development of a general domain adaptation method that mitigates this difference. Such a method would be of essential importance for many research directions where lack of real training data is the current main obstacle, including our work in Chapter 4.

Considering all these directions together reaffirms the main insight on which this dissertation is built, and expands it further: analyzing the visual data in our world and making its information content accessible can benefit from not only cross-dimensional links but from cross-modal ones in general. As we want to extract more information from the 2D photographs and 3D models and scenes we produce on a daily basis, it will be useful to consider these problems in a joint fashion. We have shown this maxim to be true for one subset of problems, and are looking forward to the exciting applications that will result from its continued exploration.

Appendix A

Full Results for Chapter 3

Results start on next page.

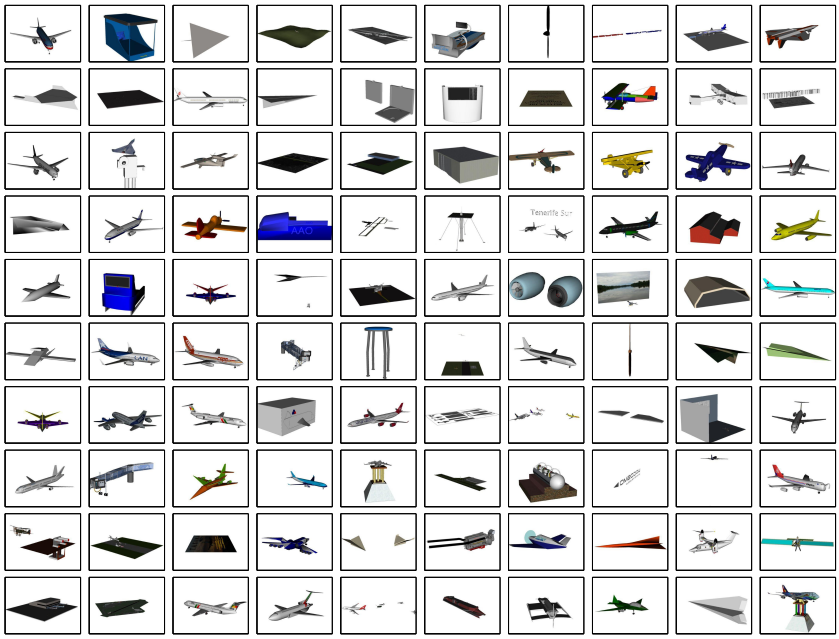


Figure A.1: First 100 models in original 3D set for class “airplane”.

Figure A.2: First 100 models in resorted and realigned 3D set for class “airplane”.

Figure A.3: First 100 images in original 2D set for class “airplane”.

Figure A.4: Top 6 images for 5 view classifiers of class “airplane”. Each column represents one view classifier.

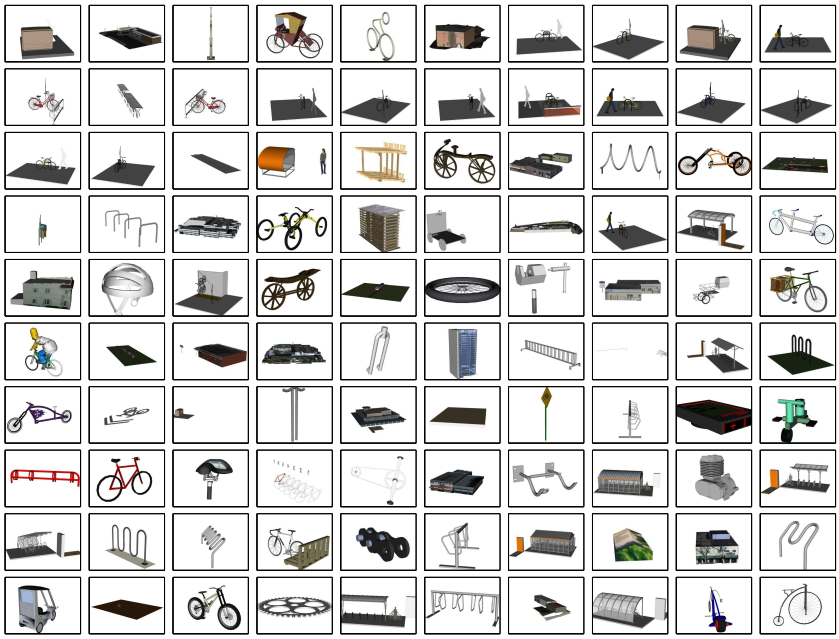


Figure A.5: First 100 models in original 3D set for class “bicycle”.

Figure A.6: First 100 models in resorted and realigned 3D set for class “bicycle”.

Figure A.7: First 100 images in original 2D set for class “bicycle”.

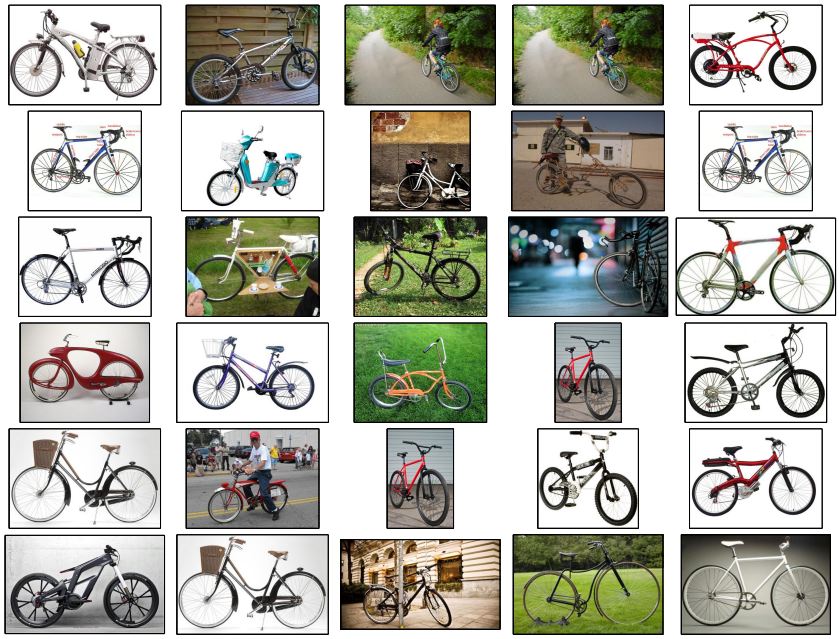


Figure A.8: Top 6 images for 5 view classifiers of class “bicycle”. Each column represents one view classifier.

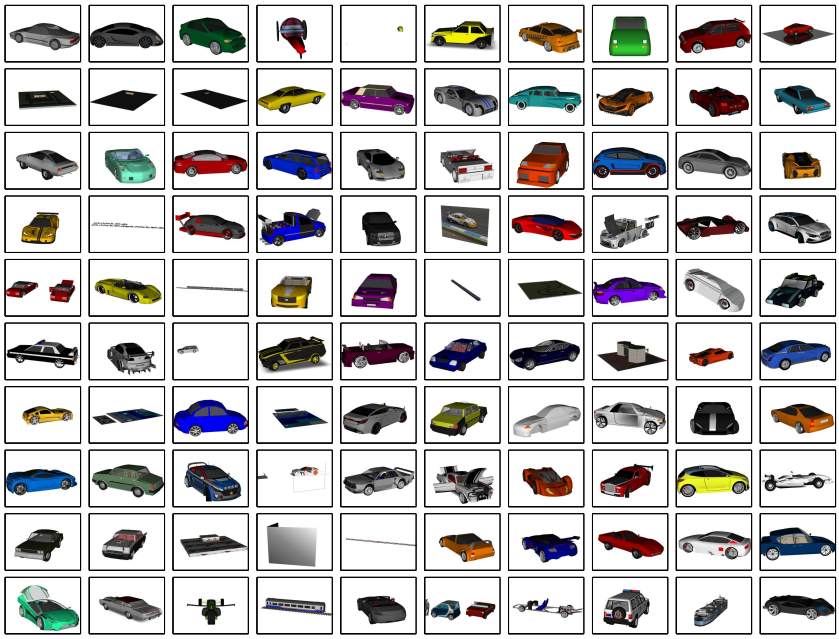


Figure A.9: First 100 models in original 3D set for class “car”.

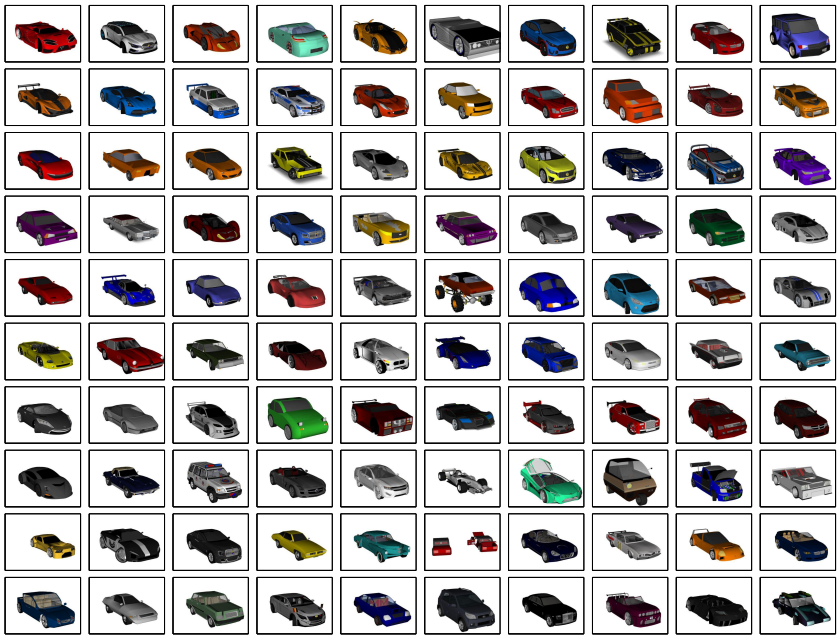


Figure A.10: First 100 models in resorted and realigned 3D set for class “car”.

Figure A.11: First 100 images in original 2D set for class “car”.

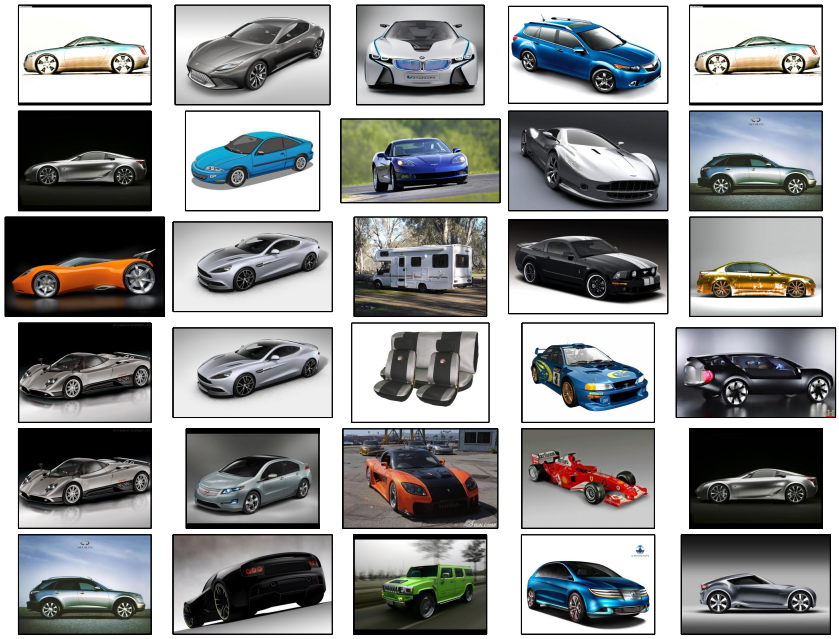


Figure A.12: Top 6 images for 5 view classifiers of class “car”. Each column represents one view classifier.

Figure A.13: First 100 models in original 3D set for class “couch”.

Figure A.14: First 100 models in resorted and realigned 3D set for class “couch”.

Figure A.15: First 100 images in original 2D set for class “couch”.

Figure A.16: Top 6 images for 5 view classifiers of class “couch”. Each column represents one view classifier.

Figure A.17: First 100 models in original 3D set for class “helicopter”.

Figure A.18: First 100 models in resorted and realigned 3D set for class “helicopter”.

Figure A.19: First 100 images in original 2D set for class “helicopter”.

Figure A.20: Top 6 images for 5 view classifiers of class “helicopter”. Each column represents one view classifier.

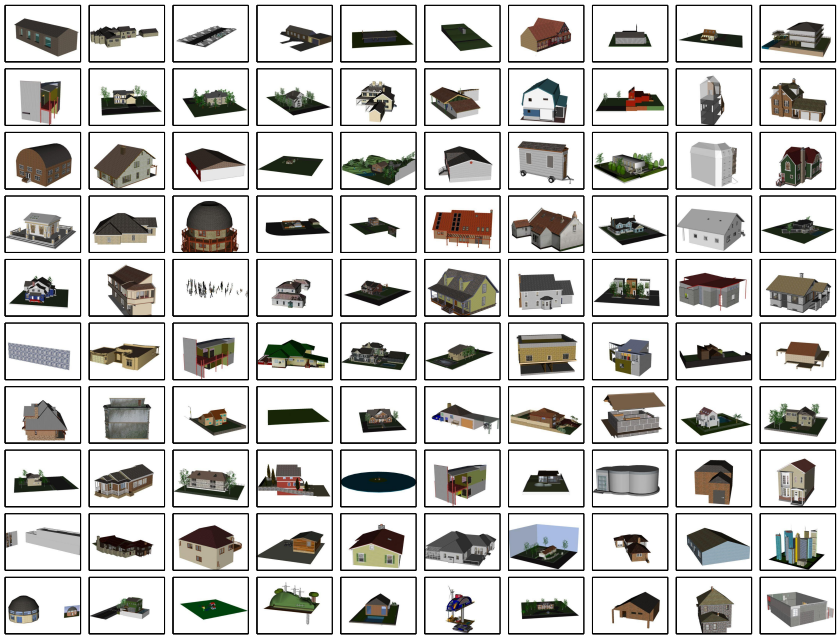


Figure A.21: First 100 models in original 3D set for class “house”.

Figure A.22: First 100 models in resorted and realigned 3D set for class “house”.

Figure A.23: First 100 images in original 2D set for class “house”.

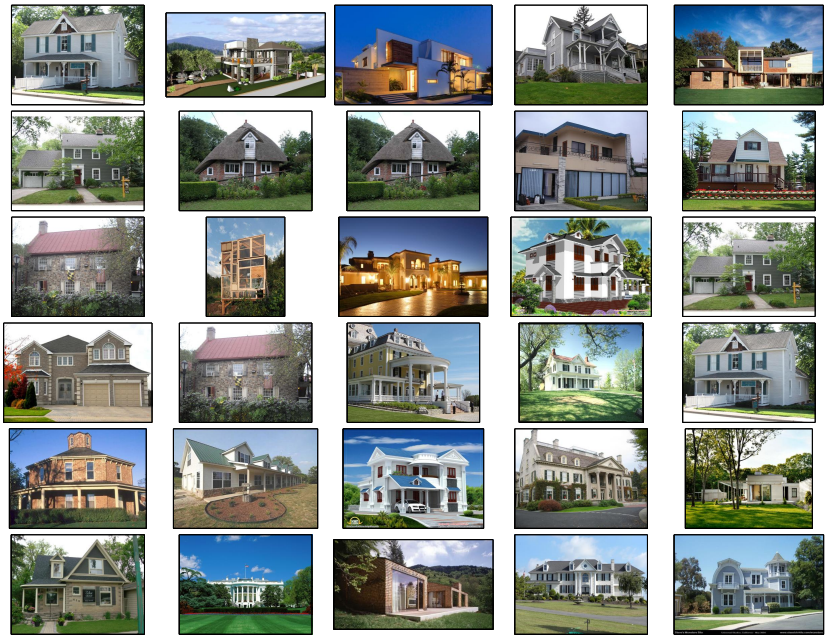


Figure A.24: Top 6 images for 5 view classifiers of class “house”. Each column represents one view classifier.

Appendix B

Full Results for Chapter 5

Results start on next page.

Camera view
Ours
SC
3D-INN
Top view

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN

Camera view
Ours
SC
3D-INN
Top view

Top view
Ours
SC
3D-INN
Top view
Ours
SC
3D-INN
Camera view
Ours
SC
3D-INN
Camera view
Ours
SC
3D-INN

Appendix C

List of publications

The work in this thesis appears in the following publications:

- HUETING, M., OVSJANIKOV, M., AND MITRA, N. J. Crosslink: joint understanding of image and 3d model collections through shape and camera pose variations. *Proc. ACM/SIGGRAPH Asia 34*, 6 (2015), 233

- HUETING, M., PĂTRĂUCEAN, V., OVSJANIKOV, M., AND MITRA, N. J. Scene structure inference through scene map estimation. In *Proc. Vision, Modeling & Visualization* (2016)

Bibliography

[1] houzz.
urlhttps://www.houzz.co.uk/.

[2] IBM ILOG CPLEX Optimizer.
urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[3] AGARWAL, S., MIERLE, K., AND OTHERS. Ceres solver. `http://ceres-solver.org`.

[4] AGRAWAL, P., CARREIRA, J., AND MALIK, J. Learning to see by moving. In *Proc. International Conference on Computer Vision* (2015), pp. 37–45.

[5] ANDRES, B., BEIER, T., AND KAPPES, J. OpenGM: A C++ library for discrete graphical models. *CoRR abs/1206.0111* (2012).

[6] AUBRY, M., MATURANA, D., EFROS, A. A., RUSSELL, B. C., AND SIVIC, J. Seeing 3d chairs: Exemplar part-based 2d-3d alignment using a large dataset of CAD models. In *Proc. Conference on Computer Vision and Pattern Recognition* (2014), pp. 3762–3769.

[7] AUBRY, M., RUSSELL, B. C., AND SIVIC, J. Painting-to-3d model alignment via discriminative visual elements. *ACM Trans. Graph. 33*, 2 (2014), 14:1–14:14.

[8] AVERKIOU, M., KIM, V. G., AND MITRA, N. J. Autocorrelation descriptor for efficient co-alignment of 3d shape collections. *Computer Graphics Forum 35*, 1 (2016), 261–271.

[9] AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum 33*, 2 (2014), 125–134.

[10] BANSAL, A., RUSSELL, B., AND GUPTA, A. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proc. Conference on Computer Vision and Pattern Recognition* (2016).

[11] BERGSTRA, J., YAMINS, D., AND COX, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. International Conference on Machine Learning* (2013), pp. 115–123.

[12] BLANZ, V., AND VETTER, T. A morphable model for the synthesis of 3d faces. *Proc. ACM/SIGGRAPH* (1999), 187–194.

[13] BOSCAINI, D., MASCI, J., MELZI, S., BRONSTEIN, M. M., CASTELLANI, U., AND VANDERGHEYNST, P. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Proc. Symposium on Geometry Processing 34*, 5 (2015), 13–23.

[14] BOTTOU, L. Online learning and stochastic approximations. In *Online Learning in Neural Networks*. Cambridge University Press, 1998, ch. 2, pp. 9–42.

[15] BOUREAU, Y., PONCE, J., AND LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In *Proc. International Conference on Machine Learning* (2010), pp. 111–118.

[16] CARNEIRO, G., CHAN, A. B., MORENO, P. J., AND VASCONCELOS, N. Supervised learning of semantic classes for image annotation and retrieval. *Proc. Pattern Analysis and Machine Intelligence 29*, 3 (2007), 394–410.

[17] CGTRADER. CGTrader, 2017.

[18] CHEN, D., TIAN, X., SHEN, Y., AND OUHYOUNG, M. On visual similarity based 3d model retrieval. *Computer Graphics Forum 22*, 3 (2003), 223–232.

[19] CHEN, K., XU, K., YU, Y., WANG, T., AND HU, S. Magic decorator: automatic material suggestion for indoor digital scenes. *Proc. ACM/SIGGRAPH Asia 34*, 6 (2015), 232.

[20] CHEN, L., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *Arxiv abs/1412.7062* (2014).

[21] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

[22] CHEN, W., WANG, H., LI, Y., SU, H., TU, C., LISCHINSKI, D., COHEN-OR, D., AND CHEN, B. Synthesizing training images for boosting human 3d pose estimation. *Arxiv abs/1604.02703* (2016).

[23] CORSINI, M., DELLEPIANE, M., PONCHIO, F., AND SCOPIGNO, R. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. *Computer Graphics Forum 28*, 7 (2009), 1755–1764.

[24] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297.

[25] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *Proc. Conference on Computer Vision and Pattern Recognition* (2005), pp. 886–893.

[26] DAMBREVILLE, S., SANDHU, R., YEZZI, A. J., AND TANNENBAUM, A. Robust 3d pose estimation and efficient 2d region-based segmentation from

a 3d shape prior. In *Proc. European Conference on Computer Vision* (2008), pp. 169–182.

[27] DATTA, R., JOSHI, D., LI, J., AND WANG, J. Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys 40*, 2 (2008).

[28] DENG, J., DONG, W., SOCHER, R., LI, L., LI, K., AND LI, F. Imagenet: A large-scale hierarchical image database. In *Proc. Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255.

[29] EIGEN, D., AND FERGUS, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proc. International Conference on Computer Vision* (2015), pp. 2650–2658.

[30] EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. Sketch-based shape retrieval. *Proc. ACM/SIGGRAPH 31*, 4 (2012), 31:1–31:10.

[31] EVGENIOU, T., AND PONTIL, M. Regularized multi–task learning. In *Proc. ACM/SIGKDD* (2004), pp. 109–117.

[32] FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D. A., AND RAMANAN, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 9 (2010), 1627–1645.

[33] FELZENSZWALB, P. F., MCALLESTER, D. A., AND RAMANAN, D. A discriminatively trained, multiscale, deformable part model. In *Proc. Conference on Computer Vision and Pattern Recognition* (2008), pp. 1–8.

[34] FISHER, M., RITCHIE, D., SAVVA, M., FUNKHOUSER, T., AND HANRAHAN, P. Example-based synthesis of 3d object arrangements. *Proc. ACM/SIGGRAPH Asia* (2012).

[35] FISHER, M., SAVVA, M., LI, Y., HANRAHAN, P., AND NIESSNER, M. Activity-centric scene synthesis for functional 3d scene modeling. *Proc. ACM/SIGGRAPH 34*, 6 (2015), 179.

[36] GIRDHAR, R., FOUHEY, D. F., RODRIGUEZ, M., AND GUPTA, A. Learning a predictable and generative vector representation for objects. *Arxiv abs/1603.08637* (2016).

[37] GODARD, C., MAC AODHA, O., AND BROSTOW, G. J. Unsupervised monocular depth estimation with left-right consistency. In *CVPR* (2017).

[38] GOLDFEDER, C., AND ALLEN, P. Autotagging to improve text search for 3d models. In *Proc. ACM/IEEE-CS joint conference on Digital libraries* (2008), ACM, pp. 355–358.

[39] GONG, B., SHI, Y., SHA, F., AND GRAUMAN, K. Geodesic flow kernel for unsupervised domain adaptation. In *Proc. Conference on Computer Vision and Pattern Recognition* (2012), pp. 2066–2073.

[40] GOOGLE. Google Official Blog, 2010.

[41] GOOGLE. Project Tango, 2014.

[42] GOULD, S., AND HE, X. Scene understanding by labeling pixels. *Communications of the ACM 57*, 11 (2014), 68–77.

[43] GUPTA, S., ARBELÁEZ, P. A., GIRSHICK, R. B., AND MALIK, J. Aligning 3d models to RGB-D images of cluttered scenes. In *Proc. Conference on Computer Vision and Pattern Recognition* (2015), pp. 4731–4740.

[44] HANDA, A., PATRAUCEAN, V., BADRINARAYANAN, V., STENT, S., AND CIPOLLA, R. Scenenet: Understanding real world indoor scenes with synthetic data. *Arxiv abs/1511.07041* (2015).

[45] HARTLEY, R., AND ZISSERMAN, A. *Multiple view geometry in computer vision*. @Cambridge university press, 2003.

[46] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proc. Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[47] HEDAU, V., HOIEM, D., AND FORSYTH, D. A. Recovering the spatial layout of cluttered rooms. In *Proc. International Conference on Computer Vision* (2009), pp. 1849–1856.

[48] HOUGEN, D. R., AND AHUJA, N. Estimation of the light source distribution and its use in integrated shape recovery from stereo and shading. In *Proc. International Conference on Computer Vision* (1993), pp. 148–155.

[49] HU, R., VAN KAICK, O., WU, B., HUANG, H., SHAMIR, A., AND ZHANG, H. Learning how objects function via co-analysis of interactions. *Proc. ACM/SIGGRAPH 35*, 4 (2016), 47.

[50] HUANG, Q., SU, H., AND GUIBAS, L. J. Fine-grained semi-supervised labeling of large shape collections. *Proc. ACM/SIGGRAPH Asia 32*, 6 (2013), 190:1–190:10.

[51] HUANG, Q., WANG, F., AND GUIBAS, L. J. Functional map networks for analyzing and exploring large shape collections. *Proc. ACM/SIGGRAPH 33*, 4 (2014), 36:1–36:11.

[52] HUANG, Q., WANG, H., AND KOLTUN, V. Single-view reconstruction via joint analysis of image and shape collections. *Proc. ACM/SIGGRAPH Asia 34*, 4 (2015), 87.

[53] HUANG, S., SHAMIR, A., SHEN, C., ZHANG, H., SHEFFER, A., HU, S., AND COHEN-OR, D. Qualitative organization of collections of shapes via quartet analysis. *Proc. ACM/SIGGRAPH 32*, 4 (2013), 71:1–71:10.

[54] HUETING, M., OVSJANIKOV, M., AND MITRA, N. J. Crosslink: joint understanding of image and 3d model collections through shape and camera pose variations. *Proc. ACM/SIGGRAPH Asia 34*, 6 (2015), 233.

[55] HUETING, M., PĂTRĂUCEAN, V., OVSJANIKOV, M., AND MITRA, N. J. Scene structure inference through scene map estimation. In *Proc. Vision, Modeling & Visualization* (2016).

[56] HYPEROPT. HyperOpt, 2017.

[57] INFOTRENDS. How long does it take to shoot 1 trillion photos?, 2016.

[58] INTEL. RealSense, 2017.

[59] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning* (2015), pp. 448–456.

[60] IZADINIA, H., SHAN, Q., AND SEITZ, S. M. IM2CAD. *CoRR abs/1608.05137* (2016).

[61] JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H. Material memex: automatic material suggestions for 3d objects. *Proc. ACM/SIGGRAPH Asia 31*, 6 (2012), 143.

[62] JARRETT, K., KAVUKCUOGLU, K., RANZATO, M., AND LECUN, Y. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision* (2009), pp. 2146–2153.

[63] JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R. B., GUADARRAMA, S., AND DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM International Conference on Multimedia* (2014), pp. 675–678.

[64] KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. A. Learning part-based templates from large collections of 3d shapes. *Proc. ACM/SIGGRAPH 32*, 4 (2013), 70:1–70:12.

[65] KLEIMAN, Y., FISH, N., LANIR, J., AND COHEN-OR, D. Dynamic maps for exploring and browsing shapes. *Proc. Symposium on Geometry Processing 32*, 5 (2013), 187–196.

[66] KOENDERINK, J. J., AND PONT, S. C. Irradiation direction from texture. *Journal of the Optical Society of America 20*, 10 (2003), 1875–1882.

[67] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems* (2012), pp. 1106–1114.

[68] LEARNED-MILLER, E. G. Data driven image models through continuous joint alignment. *Proc. Pattern Analysis and Machine Intelligence 28*, 2 (2006), 236–250.

[69] LEE, D. C., HEBERT, M., AND KANADE, T. Geometric reasoning for single image structure recovery. In *Proc. Conference on Computer Vision and Pattern Recognition* (2009), pp. 2136–2143.

[70] LI, B., LU, Y., LI, C., GODIL, A., SCHRECK, T., AONO, M., BURTSCHER, M., CHEN, Q., CHOWDHURY, N. K., FANG, B., FU, H., FURUYA, T., LI, H., LIU, J., JOHAN, H., KOSAKA, R., KOYANAGI, H., OHBUCHI, R., TATSUMA, A., WAN, Y., ZHANG, C., AND ZOU, C. A comparison of 3d shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding 131* (2015), 1–27.

[71] LI, Y., SU, H., QI, C. R., FISH, N., COHEN-OR, D., AND GUIBAS, L. J. Joint embeddings of shapes and images via CNN image purification. *Proc. ACM/SIGGRAPH Asia 34*, 6 (2015), 234:1–234:12.

[72] LI, Y., ZHENG, Q., SHARF, A., COHEN-OR, D., CHEN, B., AND MITRA, N. J. 2d-3d fusion for layer decomposition of urban facades. In *Proc. International Conference on Computer Vision* (2011), pp. 882–889.

[73] LIM, J. J., PIRSIAVASH, H., AND TORRALBA, A. Parsing IKEA objects: Fine pose estimation. In *Proc. International Conference on Computer Vision* (2013), pp. 2992–2999.

[74] LIU, Z., ZHANG, Y., WU, W., LIU, K., AND SUN, Z. Model-driven indoor scenes modeling from a single image. In *Proc. Graphics Interface Conference* (2015), pp. 25–32.

[75] LOPEZ-MORENO, J., GARCES, E., HADAP, S., REINHARD, E., AND GUTIERREZ, D. Multiple light source estimation in a single image. *Proc. Eurographics 32*, 8 (2013), 170–182.

[76] LOPEZ-MORENO, J., SUNDSTEDT, V., SANGORRIN, F., AND GUTIERREZ, D. Measuring the perception of light inconsistencies. In *Proc. Symposium on Applied Perception in Graphics and Visualization* (2010), pp. 25–32.

[77] MANDAL, D., CHAUDHURY, K. N., AND BISWAS, S. Generalized semantic preserving hashing for n-label cross-modal retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017).

[78] MASCI, J., BRONSTEIN, M. M., BRONSTEIN, A. M., AND SCHMIDHUBER, J. Multimodal similarity-preserving hashing. *Proc. Pattern Analysis and Machine Intelligence 36*, 4 (2014), 824–830.

[79] MASSA, F., RUSSELL, B. C., AND AUBRY, M. Deep exemplar 2d-3d detection by adapting from real to rendered views. *Arxiv abs/1512.02497* (2015).

[80] MICROSOFT. Microsoft Kinect, 2013.

[81] MIN, P., KAZHDAN, M. M., AND FUNKHOUSER, T. A. A comparison of text and shape matching for retrieval of online 3d models. *Proc. Research and Advanced Technology for Digital Libraries* (2004), 209–220.

[82] MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. Structure-aware shape processing. In *Eurographics 2013 State of the Art Reports* (2013), pp. 175–197.

[83] MONSZPART, A., MELLADO, N., BROSTOW, G. J., AND MITRA, N. J. Rapter: rebuilding man-made scenes with regular arrangements of planes. *Proc. ACM/SIGGRAPH 34*, 4 (2015), 103.

[84] NGUYEN, C. H., RITSCHEL, T., MYSZKOWSKI, K., EISEMANN, E., AND SEIDEL, H. 3d material style transfer. *Proc. Eurographics 31*, 2 (2012), 431–438.

[85] NOH, H., HONG, S., AND HAN, B. Learning deconvolution network for semantic segmentation. In *Proc. International Conference on Computer Vision* (2015), pp. 1520–1528.

[86] OLSON, E., AND AGARWAL, P. Inference on networks of mixtures for robust robot mapping. *I. J. Robotics Res. 32*, 7 (2013), 826–840.

[87] OVSJANIKOV, M., LI, W., GUIBAS, L. J., AND MITRA, N. J. Exploration of continuous variability in collections of 3d shapes. *Proc. ACM/SIGGRAPH 30*, 4 (2011), 33.

[88] PEREIRA, J. C., COVIELLO, E., DOYLE, G., RASIWASIA, N., LANCKRIET, G. R. G., LEVY, R., AND VASCONCELOS, N. On the role of correlation and abstraction in cross-modal multimedia retrieval. *Proc. Pattern Analysis and Machine Intelligence 36*, 3 (2014), 521–535.

[89] PICARD, R., GRACZYK, C., MANN, S., WACHMAN, J., PICARD, L., AND CAMPBELL, L. Vistex texture dataset. `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html`, 1995. Accessed: 2016-05-01.

[90] PLATT, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. MIT Press, 1999, ch. 5, pp. 61–74.

[91] POWELL, M. W., SARKAR, S., AND GOLDGOF, D. B. A simple strategy for calibrating the geometry of light sources. *Proc. Pattern Analysis and Machine Intelligence 23*, 9 (2001), 1022–1027.

[92] PRISACARIU, V. A., AND REID, I. D. PWP3D: real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision 98*, 3 (2012), 335–354.

[93] RAFIEE, G., DLAY, S. S., AND WOO, W. L. A review of content-based image retrieval. In *Proc. International Symposium on Communication Systems Networks and Digital Signal Processing* (2010), pp. 775–779.

[94] REN, S., HE, K., GIRSHICK, R. B., AND SUN, J. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Information Processing Systems* (2015), pp. 91–99.

[95] ROTHER, C. A new approach to vanishing point detection in architectural environments. *Image Vision Computing 20*, 9-10 (2002), 647–655.

[96] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M. S., BERG, A. C., AND LI, F. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252.

[97] RUSSELL, B. C., SIVIC, J., PONCE, J., AND DESSALES, H. Automatic alignment of paintings and photographs depicting a 3d scene. In *Proc. International Conference on Computer Vision* (2011), pp. 545–552.

[98] SAUNDERS, C., GAMMERMAN, A., AND VOVK, V. Ridge regression learning algorithm in dual variables. In *Proc. International Conference on Machine Learning* (1998), pp. 515–521.

[99] SHAO, T., MONSZPART, A., ZHENG, Y., KOO, B., XU, W., ZHOU, K., AND MITRA, N. J. Imagining the unseen: stability-based cuboid arrangements for scene understanding. *Proc. ACM/SIGGRAPH 33*, 6 (2014), 209:1–209:11.

[100] SHILANE, P., MIN, P., KAZHDAN, M. M., AND FUNKHOUSER, T. A. The princeton shape benchmark. In *Proc. International Conference on Shape Modeling and Applications* (2004), pp. 167–178.

[101] SHRIVASTAVA, A., MALISIEWICZ, T., GUPTA, A., AND EFROS, A. A. Data-driven visual similarity for cross-domain image matching. *Proc. ACM/SIGGRAPH Asia 30*, 6 (2011), 154.

[102] SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. Indoor segmentation and support inference from RGBD images. In *Proc. European Conference on Computer Vision* (2012), pp. 746–760.

[103] SILBERMAN, N., SONTAG, D., AND FERGUS, R. Instance segmentation of indoor scenes using a coverage loss. In *Proc. European Conference on Computer Vision* (2014), pp. 616–631.

[104] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[105] SKETCHFAB. Sketchfab, 2017.

[106] SONG, S., LICHTENBERG, S. P., AND XIAO, J. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proc. Conference on Computer Vision and Pattern Recognition* (2015), pp. 567–576.

[107] SONG, S., AND XIAO, J. Sliding shapes for 3d object detection in depth images. In *Proc. European Conference on Computer Vision* (2014), pp. 634–651.

[108] STATISTA. Number of smartphone users worldwide from 2014 to 2020, 2017.

[109] SU, H., HUANG, Q., MITRA, N. J., LI, Y., AND GUIBAS, L. J. Estimating image depth using shape collections. *Proc. ACM/SIGGRAPH 33*, 4 (2014), 37:1–37:11.

[110] SU, H., QI, C. R., LI, Y., AND GUIBAS, L. J. Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proc. International Conference on Computer Vision* (2015), pp. 2686–2694.

[111] TANGELDER, J. W., AND VELTKAMP, R. C. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications 39*, 3 (2008), 441.

[112] THRUN, S., BURGARD, W., AND FOX, D. *Probabilistic Robotics*. MIT Press, 2005.

[113] TIELEMAN, T., AND HINTON, G. E. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[114] TOME, D., RUSSELL, C., AND AGAPITO, L. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proc. Conference on Computer Vision and Pattern Recognition* (July 2017).

[115] TRIMBLE. 3D Warehouse, 2017.

[116] TSOCHANTARIDIS, I., JOACHIMS, T., HOFMANN, T., AND ALTUN, Y. Large margin methods for structured and interdependent output variables. *Journal of machine learning research 6*, Sep (2005), 1453–1484.

[117] TURBOSQUID. Turbosquid, 2017.

[118] VAN GEMERT, J. C., GEUSEBROEK, J., VEENMAN, C. J., AND SMEULDERS, A. W. M. Kernel codebooks for scene categorization. In *Proc. European Conference on Computer Vision* (2008), pp. 696–709.

[119] VICENTE, S., CARREIRA, J., AGAPITO, L., AND BATISTA, J. Reconstructing PASCAL VOC. In *Proc. Conference on Computer Vision and Pattern Recognition* (2014), pp. 41–48.

[120] WANG, F., KANG, L., AND LI, Y. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition* (2015), pp. 1875–1883.

[121] WANG, K., HE, R., WANG, L., WANG, W., AND TAN, T. Joint feature selection and subspace learning for cross-modal retrieval. *IEEE transactions on pattern analysis and machine intelligence 38*, 10 (2016), 2010–2023.

[122] WANG, Y., GONG, M., WANG, T., COHEN-OR, D., ZHANG, H., AND CHEN, B. Projective analysis for 3d shape segmentation. *Proc. ACM/SIGGRAPH Asia 32*, 6 (2013), 192.

[123] WEI, L., HUANG, Q., CEYLAN, D., VOUGA, E., AND LI, H. Dense human body correspondences using convolutional networks. *Arxiv abs/1511.05904* (2015).

[124] WEI, S., RAMAKRISHNA, V., KANADE, T., AND SHEIKH, Y. Convolutional pose machines. In *Proc. Conference on Computer Vision and Pattern Recognition* (2016), pp. 4724–4732.

[125] WESTON, J., BENGIO, S., AND USUNIER, N. WSABIE: scaling up to large vocabulary image annotation. In *Proc. International Joint Conference on Artificial Intelligence* (2011), pp. 2764–2770.

[126] WILSON, S. Dwelling Size Survey, 2010.

[127] WU, B., YANG, Q., ZHENG, W.-S., WANG, Y., AND WANG, J. Quantized correlation hashing for fast cross-modal search. In *IJCAI* (2015), pp. 3946–3952.

[128] WU, J., XUE, T., LIM, J. J., TIAN, Y., TENENBAUM, J. B., TORRALBA, A., AND FREEMAN, W. T. Single image 3d interpreter network. In *European Conference on Computer Vision* (2016), pp. 365–382.

[129] WU, Z., SONG, S., KHOSLA, A., YU, F., ZHANG, L., TANG, X., AND XIAO, J. 3d shapenets: A deep representation for volumetric shapes. In *Proc. Conference on Computer Vision and Pattern Recognition* (2015), pp. 1912–1920.

[130] XIE, J., DAI, G., ZHU, F., AND FANG, Y. Learning barycentric representations of 3d shapes for sketch-based 3d shape retrieval. In *Proc. Conference on Computer Vision and Pattern Recognition* (July 2017).

[131] XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z. Style-content separation by anisotropic part scales. *Proc. ACM/SIGGRAPH Asia 29*, 6 (2010), 184.

[132] XU, K., MA, R., ZHANG, H., ZHU, C., SHAMIR, A., COHEN-OR, D., AND HUANG, H. Organizing heterogeneous scene collections through contextual focal points. *ACM Trans. Graph. 33*, 4 (2014), 35:1–35:12.

[133] XU, K., ZHENG, H., ZHANG, H., COHEN-OR, D., LIU, L., AND XIONG, Y. Photo-inspired model-driven 3d object modeling. *Proc. ACM/SIGGRAPH 30*, 4 (2011), 80.

[134] YÜMER, M. E., CHAUDHURI, S., HODGINS, J. K., AND KARA, L. B. Semantic shape editing using deformation handles. *Proc. ACM/SIGGRAPH 34*, 4 (2015), 86.

[135] YÜMER, M. E., AND KARA, L. B. Co-constrained handles for deformation in shape collections. *Proc. ACM/SIGGRAPH Asia 33*, 6 (2014), 187:1–187:11.

[136] ZHANG, L., AND RUI, Y. Image search—from thousands to billions in 20 years. *ACM Trans. Multimedia Comput. Commun. Appl. 9*, 1s (2013), 36:1–36:20.

[137] ZHANG, Y., BAI, M., KOHLI, P., IZADI, S., AND XIAO, J. Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *Arxiv abs/1603.04922* (2016).

[138] ZHANG, Y., SONG, S., YUMER, E., SAVVA, M., LEE, J., JIN, H., AND FUNKHOUSER, T. A. Physically-based rendering for indoor scene under-

standing using convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition* (2017).

[139] ZHAO, Y., AND ZHU, S. Scene parsing by integrating function, geometry and appearance models. In *Proc. Conference on Computer Vision and Pattern Recognition* (2013), pp. 3119–3126.

[140] ZHOU, W., AND KAMBHAMETTU, C. Estimation of illuminant direction and intensity of multiple light sources. In *Proc. European Conference on Computer Vision* (2002), pp. 206–220.

[141] ZHU, J., LEE, Y. J., AND EFROS, A. A. Averageexplorer: interactive exploration and alignment of visual data collections. *Proc. ACM/SIGGRAPH 33*, 4 (2014), 160:1–160:11.