Learning Scene CAD Recomposition

Hamid Izadinia

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Steven M. Seitz, Chair

Alexei A. Efros

Brian Curless

Byron Boots

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Learning Scene CAD Recomposition

Hamid Izadinia

Chair of the Supervisory Committee:
Professor Steven M. Seitz
Computer Science and Engineering

Humans perceive the world in three dimensions and can interpret the hidden part of objects and scenes despite partial observations and occlusions. This level of understanding comes from imagining the hidden surfaces based on the knowledge of object shapes, scene arrangement, and high-level inference based on common scene patterns. Understanding the 3D scene with the object-level composition of its components is essential for meaningful interaction with surrounding objects in the real world physical environments.

In this thesis, first, we consider the problem of scene understanding where we show that if we can decompose a scene into its prominent objects, then we can start analyzing the scene. This enables us to infer high-level information about the scene structure, such as scene recognition and scene completion. Second, we introduce a novel perspective of recomposing the CAD model of the scene. We propose transforming raw visual sensory observations in order to re-create the scene with corresponding 3D object-level components. Towards this goal, we take advantage of large databases of object CAD models and leverage learning methods to solve this problem for real-world scenes at scale. Our *scene CAD recomposition* re-creates the scene by matching, placing, and aligning the objects from a database of thousands of CAD models. We also propose learning-based methods to automatically recompose the scene from a single-view RGB image as well as a sequence of RGB-D images for whole scene recomposition. Finally, we incorporate scene re-composition to solve a robotic object interaction problem. By means of technical analysis and

experimental studies on real-world scenes, we validate that our novel object-level scene recomposition perspective provides a useful and yet concise representation that can facilitate accomplishing downstream tasks such as object manipulation in robotics. This thesis takes first steps towards developing a fully automatic system for recomposing scenes, and we hope our work inspires future research both on 3D learning and recomposition applications.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Steve Seitz for his patience, invaluable advice and unwavering support during my Ph.D. and providing freedom for me and encouraging me to pursue ideas that excited me and challenging me with thought-provoking discussions during the course of my Ph.D. research.

I deeply thank and appreciate my committee members Alexei (Alyosha) Efros, Byron Boots and Brian Curless. Thanks Alyosha for great enthusiasm and encouragement in my research work. Thanks Byron for valuable feedback and fruitful discussions. Thanks Brian for support and insightful comments during my dissertation.

I'd like to thank Ed Lazowska and Hank Levy for constant and tireless efforts in providing a supportive culture and vivid environment in the UW CSE department. Also, thanks Lindsay Michimoto and Elise deGoede Dorough for making various administrative processes so smooth and worry-free such that sometimes I didn't even feel it during my research.

During my Ph.D. studies, I have greatly benefited from interactions with many excellent UW faculty, bright affiliate researchers and my lab mates at the GRAIL. Many thanks to Larry Zitnick, Ross Girshick and Eli Shechtman for many illuminating research discussions. Thanks Qi Shan and Ali Farhadi for productive collaborations. Thanks Steve Tanimoto for the teaching mentorship I received from him during my time as teaching assistant for the Artificial Intelligence course. I am deeply grateful and honored to learn from Ben Taskar during the very short and yet invaluable period of time that I was fortunate to interact with him.

I am very fortunate and grateful for the opportunity of doing research internships at Facebook Reality Labs and Adobe Research. I would like to thank my mentors Aaron Hertzmann, Bryan Russell, Matthew Hoffman, Christopher Twigg and Robert Wang who hosted me during my in-

ternships and from whom I learned a great many things.

Thanks Seattle for hosting me during my Ph.D. studies by serving me with rich, fresh and wild-caught Alaskan Salmon whose great taste and deep red color motivated me to learn to cook Salmon with a large variety of recipes and be honored to become a salmon master chef with a Ph.D.

I started my Ph.D. adventure with a long road trip, with my always compassionate companion Fereshteh Sadeghi. The trip that allowed me to get off the beaten path and see the natural wonders. The trip that made me feel strong during the ups and downs in my journey at Ph.D. With all my heart, I am thankful of Fereshteh. Fereshteh was my light when things were dark and my genuine treasure that no success can be compared with. Her genuine smile and unique perspective have always brought me a wealth of insightful and sweet moments.

Last but not least, I would like to thank my parents and my family for their love, support and encouragement throughout my Ph.D. career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

We humans perceive the world as a 3D structure composed of parts, pieces, objects, and spaces. To interact and live in our world, we constantly rely on our perceived 3D cues such as shape, depth, size, and distances to objects. Similarly, an artificially intelligent agent needs to understand our 3D world in order to be a helpful tool and a good human companion. The major goal of this thesis is to pose 3D scene understanding as a learning problem. We propose novel scene understanding methods and techniques for object-based scene analysis and object-level *3D scene CAD recomposition* as well as application to a real-world robotics problem. We introduce the 3D Scene CAD recomposition problem from raw sensory data. 3D Scene CAD recomposition is the object-level scene recreation using thousands of object CAD models such that the recreated 3D scene matches the real scene as close as possible. This inevitable need to write programs that see and describe the 3D world [118] has challenged *computer vision* scientists since its emergence in the 1960s.

The perception of the visual world is a hallmark of artificial intelligence. Perhaps, the earliest perceptual psychology work that influenced computer science had been the Helmholtz machine. Helmholtz coined "*unconscious* inference" in 1867 to describe an essential mechanism in the formation of visual impressions. Later, in 1950, James J. Gibson suggested that the nervous system actively constructs *conscious* visual perception. Building upon the assumption that humans utilize depth cues from perceiving an array of objects, Gibson pointed out the importance of studying the visual world of *objects* and *surfaces* in his "Perception of the Visual World" book [49]. Being influenced by Gibson's depth cues about the effects of perspective, texture gradient, and illumination variations [49], Lawrence Roberts proposed 3D reconstruction from 2D images. Lawrence argued that *"The process of creating an object list from a picture is mainly mathematical, based on the*

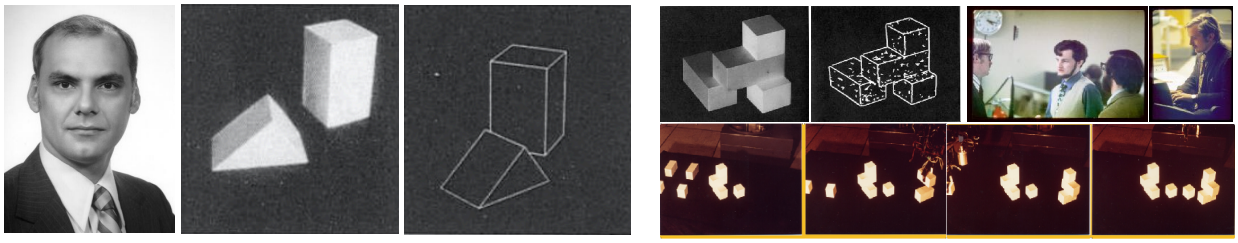


Figure 1.1: (a) Lawrence Roberts's 1963 system took an input photo and computed a 3D scene, rendered to a novel viewpoint. (b) The MIT AL Lab Copy Demo that integrated 3D blocks perception and control to replicate a block structure with a real robot.

natural laws of the space around us.". In his 1963 Ph.D. thesis, Lawrence Roberts processed the edge information in a photograph aiming to obtain a three-dimensional description. His idea was to carefully utilize the natural three-dimensional properties in projective transformation to mathematically identify objects and compute their poses and orientations in the space [129] (see Figure 1.1 part (a)).

Roberts's vision illuminated a path that resulted in seminal research works for robot vision. During the 1970s and 1980s, three-dimensional interpretation of 2D images gained interest, and "block-piling automation" was explored as one of the major research projects in Artificial Intelligence [14, 17, 63]. One of the earliest, and yet remarkably beautiful, manifestations of a complete "closed loop" AI system that integrates 3D perception and robotic control appeared in "MIT AI Lab Copy Demo" lead by Patrick Winston in the 1970s [174]. With a group of researchers at MIT, Winston created a robot with all three major components of sensing, planning, and actuation that could build a copy of a block structure. With the help of his team, Winston presented this breakthrough demo to his advisor Marvin Minsky via a 15-minute film titled "Eye of the Robot" [1] (see Figure 1.1 part (b)).

Followed by those efforts, a whole field of geometric computer vision grew during the 1990s, devoted to studying objects' appearance change as a function of different viewpoints and camera parameters [56, 42, 168, 38, 147, 82]. With the prevalence of digital cameras and the abundance of photos over the Internet in the 2000s, computer vision researchers became interested in data-driven approaches for analyzing the visual data and *re*discovering the elements of the world distilled through the filter of a camera lens. For example, data-driven spatial layout estimation was studied

in [60]. In [180], object cuboids were estimated from single images. Later, single object 3D pose estimation from RGB images [8, 95] and depth [142] were developed using hand-crafted features.

The boom of deep learning and the astonishing success of convolutional neural networks in pattern recognition [133, 50, 109, 59, 51], motivated researchers to develop neural networks specialized to different aspects of 3D scene analysis such as depth and surface normal prediction, single object pose estimation, etc. [35, 176, 169]. While those works are valuable and relevant to our work, we propose quite a complementary approach of elevating classic 3D reconstruction to a new level of object-level 3D *scene recomposition* by means of 3D deep learning and Computer Aided Design (CAD). Our scene CAD recomposition is inspired by the Roberts's Ph.D. thesis [129] and motivated by the "MIT AI Lab Copy Demo" [1], we worked to deploy it in a real robotic task. While previously CAD recomposition was not feasible at scale, with the advent of large CAD databases like ShapeNet [20] this scene recomposition becomes tractable for real-world scenes. We believe that 3D recomposition, by providing a semantic object-level scene description, opens up new research directions for a range of applications.

In this thesis, we study *3D scene learning* problem by investigating several research questions: (1) How can we learn to analyze a scene and decompose it into its major components? (2) Suppose we decomposed a scene into a set of objects, can we recompose the scene into an accurate 3D representation? (3) How can scene recomposition facilitate solving major applications of computer vision such as robotics? Figure 1.2 highlights our proposed approaches for tackling the above research questions.

The organization of this thesis is as follows. We start by studying how we can decompose a scene into its major components and how we can learn the structure of natural scenes based on its prominent object components. Learning scene structure enables making solid assumptions about scene layout, such as completing an incomplete scene using major objects. We then focus on 3D scene re-generation. In particular, we introduce the problem of 3D scene recomposition from incomplete visual sensory data. We provide solutions for object-level 3D scene recomposition using single RGB images or sequences of RGB-D images. When single RGB images are used, we can recompose a scene based on visible objects in an image. Having access to a sequence of RGB-

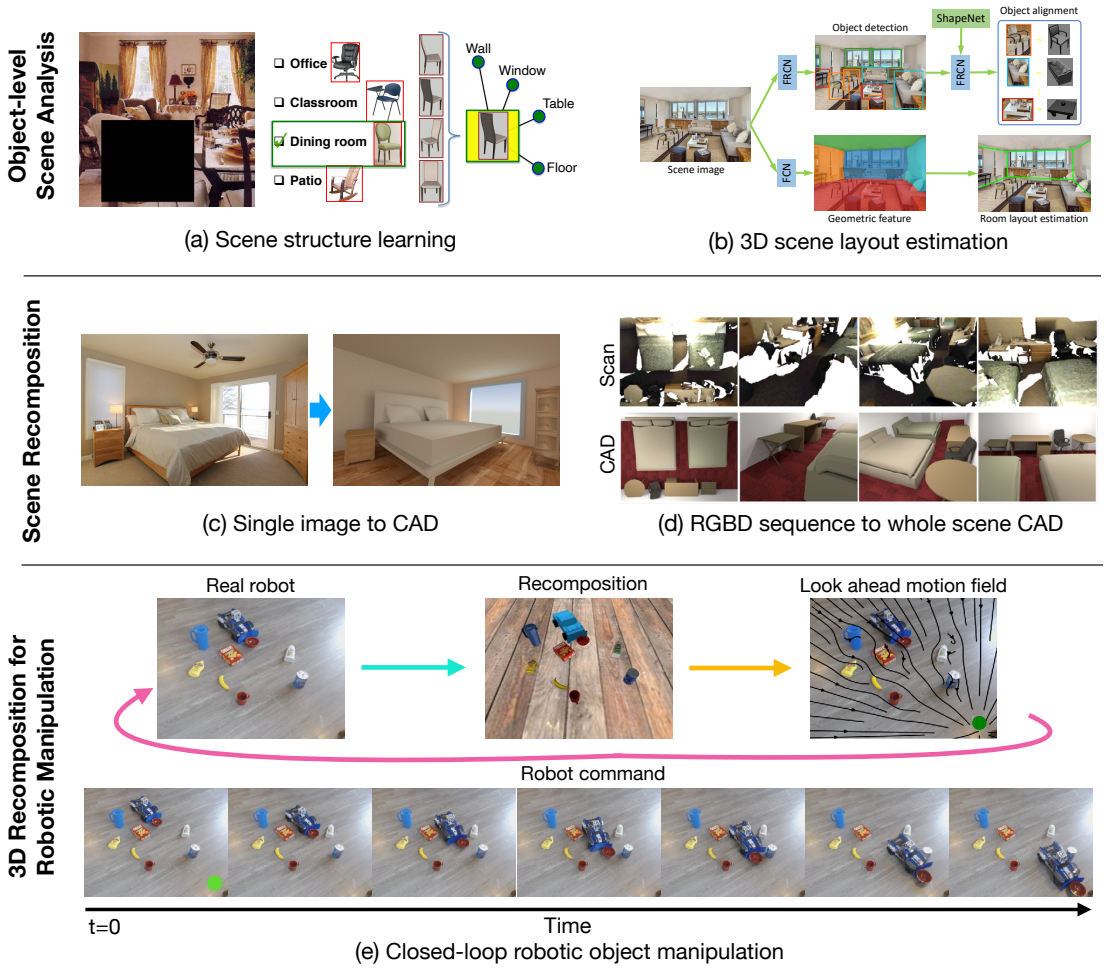


Figure 1.2: In this thesis we study object-level scene analysis and propose learning approaches for (a) scene structure learning and (b) 3D scene layout estimation. By inferring 3D objects and scene layouts, we introduce a fully automatic object-level scene recomposition technique from (c) single RGB images as well as (d) RGBD sequences. We demonstrate how 3D recomposition can be used for a real application, (e) robotic object manipulation, which involves robot object interaction in the real world.

D images and various viewpoints, we can improve the fidelity and completeness of our 3D scene recomposition and produce an object-level 3D CAD model of large environments. Along with that goal, we also propose a novel learning-based iterative closest point (LICP) technique to infer objects pose and rotation from incomplete point clouds captured with a depth camera. Finally, we consider the challenging problem of nonprehensile robotic manipulation in cluttered scenes as a use case application for scene recomposition. We demonstrate, both qualitatively and quantitatively, that based on scene recomposition, we can devise novel solutions for challenging underactuated

robotic problems with object interactions. We now describe each of these sub-problems and our contributions.

1.1 Object-level Scene Analysis

A scene is created by a set of objects arranged in a space, typically for a special purpose. Naturally, the type of objects that appear in a scene depends on functionality. Therefore, we make the assumption that a scene category imposes tight distributions over the kind of objects that might appear in the scene, the appearance of those objects and their layout. We study the correlation between different objects conditioned on different scene categories to learn scene structure. In our proposed method, we learn scene structure that encodes three main components of a scene: the scene category, the context-specific appearance of objects, and their layout. We compare and evaluate our method to a closely related prior method of Deformable Part Models [40] in detecting objects in a scene and observe that our learned scene structures result in better performance. Our model provides a level of scene understanding that is amenable to deep visual inferences. The scene structure is characterized by learned features that can later be used for scene categorization. Using these features, we also show promising results on scene categorization. Our learned scene structures are also useful in scene completion, by providing a probability distribution over the category of the missing object.

1.2 Single-View Scene CAD Recomposition

By analyzing RGB images of various scenes, we show that we can learn different scene arrangements and infer prominent objects of a scene even from incomplete images. Having access to only single RGB images, can we infer the 3D structure of a scene along with the 3D shapes and style of the objects that have furnished the scene? (see Figure 1.2 part (c)) Can we convert an RGB image (IM) to its corresponding CAD model? We call this problem Image to CAD or IM2CAD for short, and we propose 3D scene CAD model recomposition from a single image. Given a single photo of a room and a large database of furniture CAD models, our goal is to reconstruct a scene that is as

similar as possible to the scene depicted in the photograph, and composed of objects drawn from the database. We present a completely automatic system to address this IM2CAD problem that produces high quality results on challenging imagery from interior home design and remodeling websites. Our approach iteratively optimizes the placement and scale of objects in the room to best match scene renderings to the input photo, using image comparison metrics trained via deep convolutional neural nets. By operating jointly on the full scene at once, we account for inter-object occlusions. We also show the applicability of our method in standard scene understanding benchmarks where we obtain significant improvement.

1.3 Scene CAD Recomposition from RGBD Sequences

Using our proposed IM2CAD method we can create the 3D CAD model of a scene from a single RGB image. How can we handle more complicated scenes, and how can we make our CAD recomposition more accurate and complete by leveraging a sequence of images taken from different viewpoints in the scene? We work to address this question by proposing a fully automatic system for generating a 3D CAD model of the environment from a sequence of RGB-D images. Suppose we move a depth sensor around a room and scan the scene from multiple viewpoints. We can use the scan sequence as input to our proposed method and recompose the room in an object-based CAD model that captures the room shape and contents such as chairs, desks, sofas, and tables.

To generate the object-level scene recomposition, we match, place, and align each object in the scene to thousands of CAD models of objects. This is in contrast to the classic approaches where the geometry is reconstructed. More technically, we propose a novel learning method based on deep reinforcement learning to align CAD models to 3D scans. This approach, which we call Learning-based ICP (LICP), outperforms prior ICP methods in the literature, by learning the best points to match and conditioning on object viewpoint. Our proposed LICP method learns to align using only synthetic data and does not require ground truth annotation of object pose or keypoint pair matching in real scene scans. While LICP is trained on synthetic data and without 3D real scene annotations, it outperforms both learned local deep feature matching and geometric based

alignment methods in real scenes. The proposed method is evaluated on publicly available real scenes datasets of SceneNN [64] and ScanNet [30] as well as synthetic scenes of SUNCG [155]. High quality results are demonstrated on a range of real-world scenes, with robustness to clutter, viewpoint, and occlusion.

1.4 Scene CAD Recomposition for Robotic Nonprehensile Manipulation

Using our proposed scene CAD recomposition, we can generate the object-level 3D representation of the scene, which includes 3D localization of objects in the scene, estimation of their 3D poses as well as objects semantic categories. Our scene CAD recomposition enables robots to interact with objects while taking into account their physical properties, such as their complete shape and geometry. Building upon such scene representation, we propose to use the simulation for predicting the future effects of applying robot actions in a cluttered scene and analyzing the reward of robot action in simulation. We study the problem of nonprehensile robot object manipulation for pushing an object on a surface to a goal location by a real RC-car mobile robot platform. Nonprehensile object manipulation of pushing an object in a cluttered scene is challenging as it involves making and breaking contacts between robot end-effector and the objects as well as object-object contacts. Given a single RGB-D image, we automatically recompose the scene layout that includes the location of 3D objects, their 3D poses as well as objects semantic category labels. Once we recompose the scene CAD model, we run the simulation with different control parameters of a second-order dynamic system controller. Using a predictive reward estimate inside the simulation, we determine the best robot action. Accordingly, we design a closed-loop robot controller where we repeatedly recompose the scene after applying each robot action in the real world. At each step, we run our proposed Real-to-Sim future reward estimation on the updated recomposed scene in simulation and compute the best robot action. We repeat this process until the target object is pushed by the robot to the goal location. We evaluate the performance of our proposed approach on simulated and real-world RC-car robot platform and show high quality results for pushing objects in cluttered scenes.

1.5 Contributions and Roadmap

This thesis is divided into six chapters. Each chapter presents one of our main contributions and provides our proposed algorithm along with the experimental results and closely related work as follows:

Chapter 2: In this chapter, we study the scene formation as influenced by objects. We propose to learn object-based scene structures which can provide a high level scene representation. Such structures can be used in multiple downstream tasks, such as improving scene recognition and scene completion. This work is published in [67].

Chapter 3: In this chapter, we present IM2CAD which can produce a 3D CAD model of the scene from single RGB images. This work is published in [69].

Chapter 4: In this chapter, we present a learning-based full 3D scene recomposition using 3D point clouds. We propose Learning-based ICP (LICP) and generate full environment 3D scene recomposition from a sequence of sensory data in the format of point clouds. This work is published in [68].

Chapter 5: In this chapter, we present an application of scene recomposition for solving nonprehensile object manipulation, which is one of the challenging open research problems in robotics. This work is presented in [66].

Chapter 6: Finally, we discuss our concluding remarks which motivate future research in several directions.

Chapter 2

INCORPORATING SCENE CONTEXT AND OBJECT LAYOUT INTO APPEARANCE MODELING

What is behind the black box in Figure 2.1? What are the cues that enables human vision to make an intelligent guess about the object behind the box? How can human vision go beyond category prediction and reason about details of pose, style, and material? Such predictions require complex reasoning about several interlacing components that define a scene. The fact that this picture shows a dinning room scene suggests the existence of dinning chairs, dinning table, walls, and windows. By considering the layout of the room and relative locations of the table, walls, windows and other chairs we expect to see a chair behind the black box. But do we expect to see an office chair? How about a rocking chair? By knowing the layout and the scene category we can also make strong predictions about fine-grained categories in the scene. Is the chair behind the box facing the camera? Or we expect to see a lateral view of the chair? The layout of the scene, along with the appearance of other parts in the scene suggests that the chair should be at the $3/4$ view, facing away from the camera and to the table.

We argue that the *scene* type imposes a tight distribution over the categories of *objects* in the scene, their *layout*, and also their context-specific *appearance*. For example, in an office scene we expect to see a desk, an office chair, and a monitor. The desk should probably be located against the wall, the monitor should be on the desk, and the chair should face the desk. The appearance of the chair is affected by its pose (side views of chairs look different from their front views), type (office chairs typically have one central leg whereas dinning room chairs have four wooden legs), and possible predictable occlusions that one should expect in an office (in an office, chairs are occluded by desks or other office chairs in predictable patterns).

In this chapter, we propose joint learning of scene categories, the context-specific layout of the

Figure 2.1: What is behind the black box? Human observer can make predictions about the category of the object behind the box, its orientation, pose, material and style. By joint reasoning over scene categories, objects, their type, layout and context-specific appearances our method can make correct predictions about what is hidden behind the black box. Four image patches on right are the ones selected by our method based on how well they can fill in the black box.

prominent objects in the scene, and their context-specific appearance. To this end, we need to know the underlying structure of object layouts. This structure can be discovered by exploiting spatially consistent relations among objects with similar appearance.

We cast the problem of joint learning of the scene category, the object layout and their appearance models as a structure learning problem where both the topology of the structure and its parameters are to be learned. The structure corresponds to the spatial relationships between objects; For example, monitors tend to appear on the desks. The layout corresponds to the locations and scales of objects in a scene. Learning the structure of the layout requires optimizing challenging objective functions that aim at incorporating the layout topology, the layout parameters, and the appearance of objects all together. In this chapter we propose approximate solutions to this challenging problem.

Our experimental evaluations show that joint learning of scene categories, object categories, their context-specific layouts and appearance models improves not only object detection but also

scene classification. Our method outperforms state-of-the-art Deformable Part Models (DPM [39]) as well as context based object detection [25] and provides a level of scene understanding that allows deeper inferences about scenes such as those necessary to do Black Box Test (BBT). Also, our method outperforms a strong baseline that observes the content of the image behind the box in BBT.

2.1 Related Work

Our work is related to the efforts done in scene recognition. Space does not allow a comprehensive review of the literature. Here, we briefly mention few related work. Scene recognition has long been regarded as a global classification task and several methods have utilized holistic features for scene categorization [170, 87, 115, 99, 48, 182, 175]. However, as shown in [125], holistic image features fail to provide detailed scene information that is amenable for discrimination between large number of scenes with various configuration of similar objects. Large scale indoor scene classification is an example of such a task. [125] proposed training a classifier using global features combined with local features captured from manually segmented salient regions of the scenes. Following this approach, [117, 152, 140] explore the problem of scene recognition through automatically discovering discriminative scene parts using various techniques. Pandy and Lazebnik [117], utilized Deformable Part Models (DPM) object detector [39] to automatically find the salient scene regions. Sadeghi and Tappen [140] represent a scene via discriminatively discovered scene parts called Latent Pyramidal Regions (LPR). [152] and [34] learn scene parts in a jointly unsupervised/weakly-supervised manner. All these recent approaches [117, 140, 152] seek the use of *parts* as an intermediate representation of scenes but do not provide any semantics for the discovered parts. [164] presents an exemplar-based approach to image parsing. In an earlier effort, Xiao et. al. proposed the extensive Scene UNderstanding (SUN) dataset with 899 scene categories [179]. In [179] a subset of well-sampled categories (397 categories) of SUN are used to evaluate the state-of-the-art holistic features in scene recognition. To the best of our knowledge, non of the recent scene classification algorithms has considered SUN dataset for evaluation. This is mainly due to the large number of categories and images in this challenging dataset.

On the other side of the spectrum, an object-centric approach represent an image as a pool of pre-trained object detectors (called Object Bank [94, 93]). Scene recognition is performed by learning a scene category classifier using the object score-map as new features [93]. The main limitation of this approach is that the object detectors fail to provide accurate object localization because object models are learned independent from each other and the scene information. In addition, several scene components (e.g. sky, grass, wall, road) can hardly be modeled with object templates while they can be efficiently recognized if the context model is taken into account. In our method we show that learning the scene structure and the layout of prominent scene objects (semantic scene parts) can boost the performance of both scene recognition and object localization.

For improving object detection, [25] proposes an extension of non-maximum suppression that uses contextual information in the form of a single cooccurrences relation tree. [25] takes independently trained detector responses as input and smartly prunes out contextually irrelevant ones which results in improveing precision but not recall. However, we have scene specific trees that takes into account the scene specific appearances of objects (chairs in offices look different from chairs in family rooms) and their spatial/contextual relationships in a single framework. Also, we jointly train our detectors and contextual model and improve both object detection and scene recognition.

2.2 Proposed Method

Learning the underlying structure of the layout entails reasoning about the appearance of the scene, the context-specific appearance and layout of prominent objects in the scene, and the topology of objects layouts. We cast this joint learning problem as learning the underlying layout structure and the structure parameters.

To setup the notations, assume that $\mathcal{X} = \{x_1, ..., x_n\}$ is the set of training images that belong to one of the scene categories $\mathcal{C} = \{c_1, ..., c_m\}$. Each image x_i might depict a set of objects $\mathcal{O}_i = \{o_i^1, ..., o_i^{q_i}\}$ where q_i corresponds to the number of objects in the i^{th} image. Each training image has a ground truth layout $\mathcal{H}_i$ that indicates the location of bounding boxes of each object $\mathcal{H}_i = \{h_{i,1}, h_{i,2}, ..., h_{i,q_i}\}$, $h_{i,1}$ corresponds to the location and scale of the first object in the i^{th} image.

Each scene category c imposes a latent structure (topology) $\mathcal{G}_c$ over the layout. A scene structure $\mathcal{S}$ for the scene c correspond to the layout of objects, their relative locations, and their appearance models $\mathcal{S}^c = \{\mathcal{W}_c^a, \mathcal{W}_c^d, \mathcal{G}_c\}$ where $\mathcal{W}_c^a = \{W_{c,1}^a, W_{c,2}^a, ..., W_{c,p_c}^a\}$ is the set of p_c appearance models for the objects in the c^{th} scene, and $\mathcal{W}_c^d$ corresponds to the set of weight vectors that encode relative locations of objects $\mathcal{W}_c^d = \{W_{c,j,k}^d | j,k = 1 : p_c\}$. The function $\mathcal{D}$ measures how well a scene structure $\mathcal{S}^c$ is aligned with an observation x_i:

$$\begin{aligned} &\mathcal{D}(x_i, \mathcal{H}_i, \mathcal{W}_c^a, \mathcal{W}_c^d, \mathcal{G}_c) = \\ &\sum_{j=1}^{p_c}(W_{c,j}^a \phi(x_i, \mathcal{H}_i, \mathcal{G}_c) + \sum_{k=1}^{p_c} W_{c,j,k}^d \psi(\mathcal{H}_{i,j}, \mathcal{H}_{i,k}, \mathcal{G}_c)) \end{aligned} \tag{2.1}$$

where ϕ encodes unary appearance features (in our case vectorized HOG features) and ψ corresponds to binary deformation features (in our case quadratic distance transform function [39]). Discovering scene structures for a scene category c can be formulated as a structure discovery problem:

$$\begin{aligned} &\min_{\mathcal{G}_c, \mathcal{W}_c^a, \mathcal{W}_c^d, \mathcal{H}, \xi} \quad \sum_{j=1}^{p_c} \|W_{c,j}^a\|_2^2 + \\ &\sum_{j,k=1}^{p_c} \|W_{c,j,k}^d\|_2^2 + \lambda_1 \sum_{i=1}^{n} \xi_i + \lambda_2 \|\mathcal{G}_c\|_\bullet \\ &\mathcal{D}(x_i, \mathcal{H}_i, \mathcal{W}_c^a, \mathcal{W}_c^d, \mathcal{G}_c) \geq \\ &\max_{\mathcal{H}^*} \mathcal{D}(x_i, \mathcal{H}_i^*, \mathcal{W}_c^a, \mathcal{W}_c^d, \mathcal{G}_c) + \Delta(\mathcal{H}_i, \mathcal{H}_i^*) - \xi_i \quad \forall i \\ &\xi_i \geq 0 \quad \forall i \end{aligned} \tag{2.2}$$

where $\|\mathcal{G}\|_\bullet$ is a form of complexity regularizer over the topology of the structure, and Δ is a form of structured loss.

Joint optimization over the topology of the structure and the parameter of the structure, the appearance and deformation models of objects, is extremely challenging. This is an optimization over highly interleaving parameters $\mathcal{G}$ and $\mathcal{H}$. In fact, this is an NP-Hard problem. To approximate this hard optimization we decouple the optimization over $\mathcal{G}$ from the rest of the parameters. If $\mathcal{G}$ is known, we can rewrite the optimization 2.2 as a form of margin based structure learning problem.

Fixing $\mathcal{G}$ and putting $\mathcal{W}^a$ and $\mathcal{W}^d$ together into $\mathcal{W}$ results in:

$$\begin{aligned}
&\min_{\mathcal{W},\mathcal{H},\xi} \quad \|\mathcal{W}\|_2^2 + \lambda_1 \sum_{i=1}^{n} \xi_i \\
&\mathcal{D}(x_i, \mathcal{H}_i, \mathcal{W}) \geq \max_{\mathcal{H}^*} \mathcal{D}(x_i, \mathcal{H}_i^*, \mathcal{W}) + \Delta(\mathcal{H}_i, \mathcal{H}_i^*) - \xi_i \\
&\xi_i \geq 0 \quad \forall i.
\end{aligned} \tag{2.3}$$

We use hamming loss for the structured loss Δ and solve this optimization problem by cutting plane method [162, 183, 161].

Optimizing for $\mathcal{G}$ is a challenging problem and requires approximation. Learning for $\mathcal{G}$ entails reasoning about which objects will make it to the final layout and what is the topology of the graph connecting these prominent objects. To approximate this, we make use of domain knowledge. Distribution of objects in scenes is known to follow Zipf's law [179]. Meaning that there are large number of objects which occur very rarely in each scene and have small correlation with other objects in the scene. At the same time, a limited subset of objects exposes strong correlation across instances of a scene category. This suggests pruning scene-specific objects that are rare and relations (edges) that are spatially inconsistent. To this end, we form a Scene-Object graph $SOG_c = (V_c, E_c)$ whose nodes correspond to objects that appeared in scene c. The edges E_c correspond to spatial consistency of two objects with respect to each other across samples of scene c. Starting from a full graph SOG_c, discovering $\mathcal{G}$ can be formulated as selecting a set of nodes and edges that maximizes the prominence of objects and spatial consistency of their relations. Since the discovered structure will be used for further inferences, a crucial constraint would be to avoid loops in the resultant structure. More formally, the discovered $\mathcal{G}^*$ can be represented as the optimized set of nodes V^* and edges E^* such that:

$$\begin{aligned}
&\max_{\sigma_e, \sigma_v} \quad \sum_{v \in V} \sigma_v \Omega(v) + \sum_{e \in E} \sigma_e \Gamma(e) \\
&\text{Subject to} \\
&\sum_{v \in V} \sigma_v \leq p_c \\
&\sum_{e \in E(\mathcal{N})} \sigma_e \leq |\mathcal{N}| - 1, \quad \forall \mathcal{N} \subset V, \mathcal{N} \neq \emptyset \\
&\sigma_e \in \{0, 1\}, \sigma_v \in \{0, 1\}
\end{aligned} \tag{2.4}$$

where σ_e and σ_v are binary indicator variables that indicate which nodes and edges will make it to the final scene structure, $\Omega(v)$ is proportional to the prominence of each object in a scene, and $\Gamma(e)$ is proportional to the spatial consistency between two objects in a scene and p_c is the total number of objects in scene c. The second constraint avoids loop in the final structure. This optimization can be reduced to a form of weighted maximum spanning tree problem. We initialize the tree with the vertex with maximum $\Omega(v)$ and grow the tree by adding one edge at a time which brings the maximum gain to the equation 2.4. We stop this process until there is no edge that increases the total gain more than a certain threshold. In our experiments the prominence function $\Omega(v)$ corresponds to the frequency of object v in the instances of scene c. The spatial consistency $\Gamma(e)$ is set to $\frac{1}{\sigma}$ where σ is the variance of the Gaussian distribution for the relative spatial location of each pair of objects.

Our learned scene structures include a set of context-specific appearance filters for prominent objects in a scene category and their corresponding deformation models. To be more expressive, we use mixture models for the appearance and deformations for prominent objects. This allows our model to capture context-specific variations within a scene category. Our model encodes the relationships between the layout of the objects and their appearance. For example, the pose of a chair affects the appearance model of the nearby desk.

Inference involves computing equation 2.1 for all possible scene structures and picking the highest scoring one for each image. More specifically, we first find the part convolution scores for all mixtures and take the maximum among mixtures for computing part score in every location of the image. Then distance transform is used to efficiently pass messages from child nodes to their parents. The maximum collected score in the root node is considered as the best scene structure. In the message passing, we save the location of maximum score for each node and use them as the detection of all scene parts for the best inferred structure.

This provides not only the scene category labels but also the layout of prominent objects and their spatial relationships. The scene structures provide a level of understanding for scenes that is much richer than just scene category labels. Our experimental evaluations show that our model can localize objects in a scene significantly more accurate than state-of-the-art object detectors that

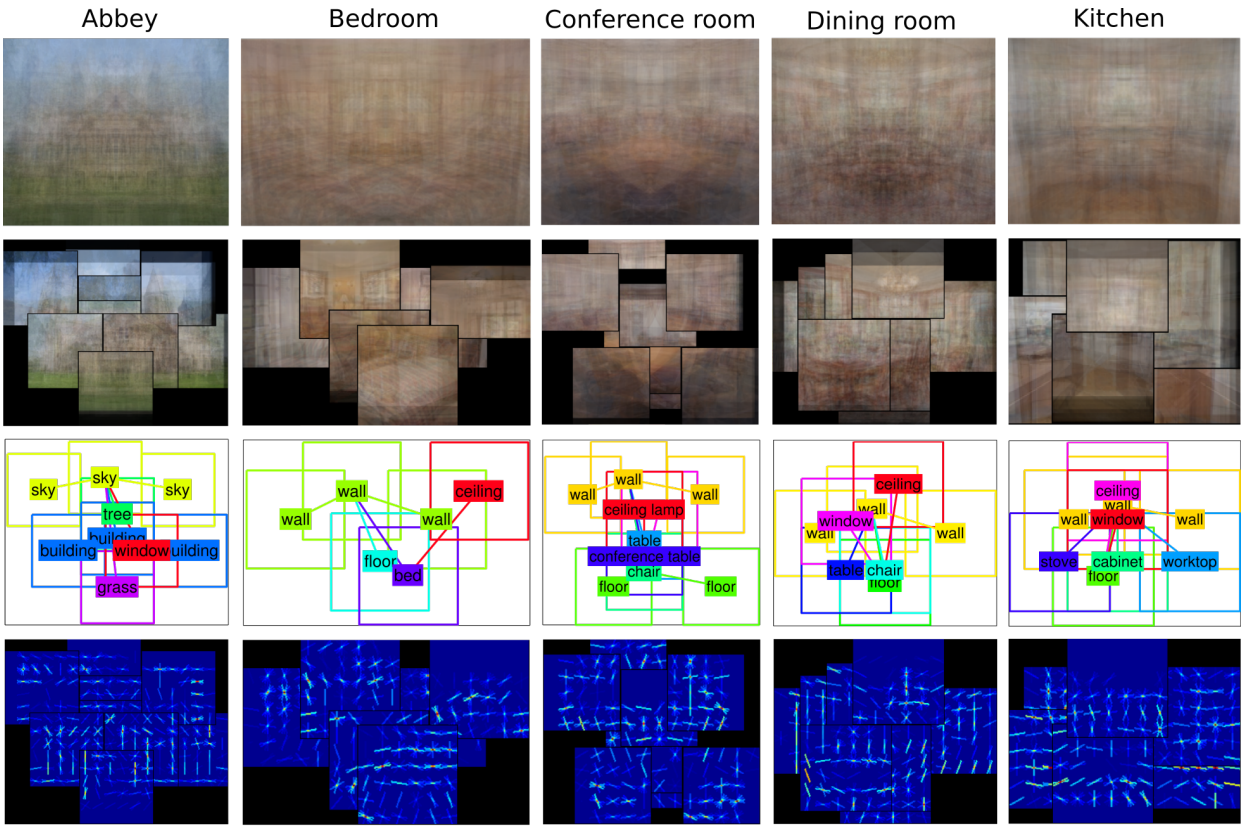


Figure 2.2: The visualization of the model for five scenes. Each scene is shown in different column. Row 1: the average of training images which shows if we use one filter for the entire image the model would be vague. Row 2: the average of image patches after finding best scene layout in each training image. The constellation of scene parts shows the discriminative shapes in each scene such as bed in bedroom and table in Dining room. Row 3: the semantic object label for each scene part and their learned layout as a tree. Row 4: visualization of the appearance models learned for each scene. Note that both the appearance models and their learned locations are context-specific.

trained independently.

2.3 Experiments

We train our scene structures using the SUN dataset that includes 397 scene categories. We use 390 categories which contained enough annotated images. We experimentally evaluate the benefits of using the discovered scene structures in object detection, the Black Box Test and also scene recognition.

Figure 2.3: Sample of best scoring scene structures that lead to accurate object detection in several challenging cases. In this image we show four different categories: Street, Office, Kitchen and Dorm room. Each row shows samples of one scene category. Discovered scene structures are superimposed into each image and objects are color-coded according to the legend on the right most column. Our model can detect the objects accurately even in very different layouts of a scene. Stuffs such as buildings and walls are also detected precisely using our flexible mixture of parts. The edges between objects correspond to the layout discovered by our method. For example, our method discovered that chairs are typically at desks and desks are typically located by the wall or windows in office scenes.

2.3.1 Object Detection

At inference, the maximum scoring scene structure contains information about objects, their appearance and also layout. Layout contains bounding box information about prominent objects in a scene.

To better encode the appearance and deformations of stuff we consider a three part model for each stuff. This allows our model to detect stuff more reliably by leveraging the structured boundaries of stuff with other prominent objects in the scene. For example, for the abbey scene,

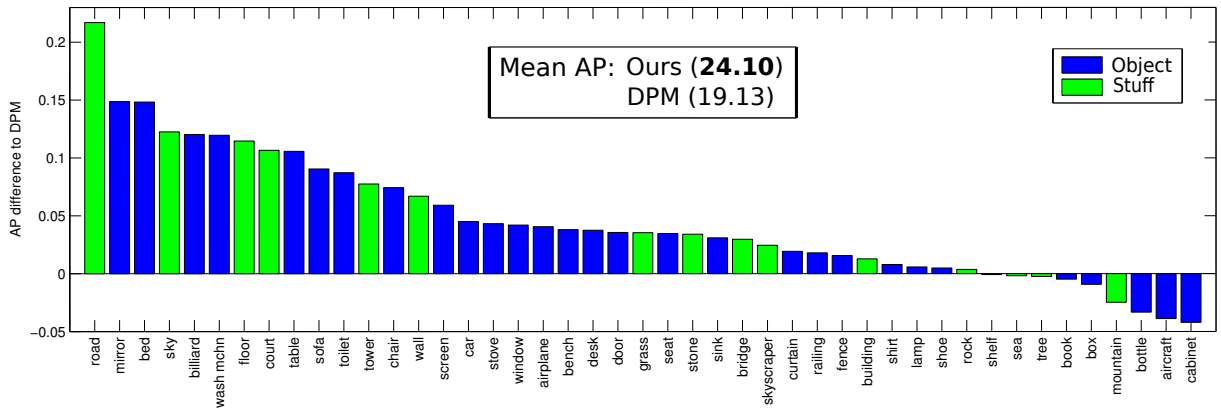


Figure 2.4: Average precision of object localization using our scene structures (ours) compared to Deformable Part Models. This plot shows the gain over DPM. Positive values corresponds to the case where our method outperforms DPM and negative values correspond to cases where DPM works better than our method. Our method outperforms DPM by significant margin. The biggest gain corresponds to objects or stuffs that are hard to detect but can be detected with the help of contextual information encoded in our scene structure. The green bars correspond to stuffs and blue bars to objects. For very small objects like bottle or objects that typically appear in less structured scenes like airplanes DPM performs better. For most of the stuff our method outperforms DPM.

our model can reliably localize sky be leveraging the fact that the boundaries of sky with the abbey structure produces a very context-specific pattern (Fig. 2.2). Fig. 2.3 also includes interesting detection results for stuff and also objects.

We compare the performance of object detection using our scene structure versus that of Deformable part models trained in Object Bank (OB) [93]. We use 7249 annotated images in SUN database from all 390 scene categories. For evaluation, in each image we only consider the labeled objects for which there is a node in our method and also an object detector trained in OB. The precision of object localization is measured by computing the intersection over union of object localization mask B_l and ground truth polygon P_{gt} (i.e. $\frac{B_l \cap P_{gt}}{B_l \cup P_{gt}}$).

For DPM models in OB, we apply models on each image and compute the score map in different levels of feature pyramid. Then the score of each level is propagated in a window with the same size as that of HOG filter [32] as object mask. The score of all levels is pooled over each pixel using max pooling. Similar to standard object detection criteria we threshold the object score map with different thresholds and compute the object localization precision for each threshold. The

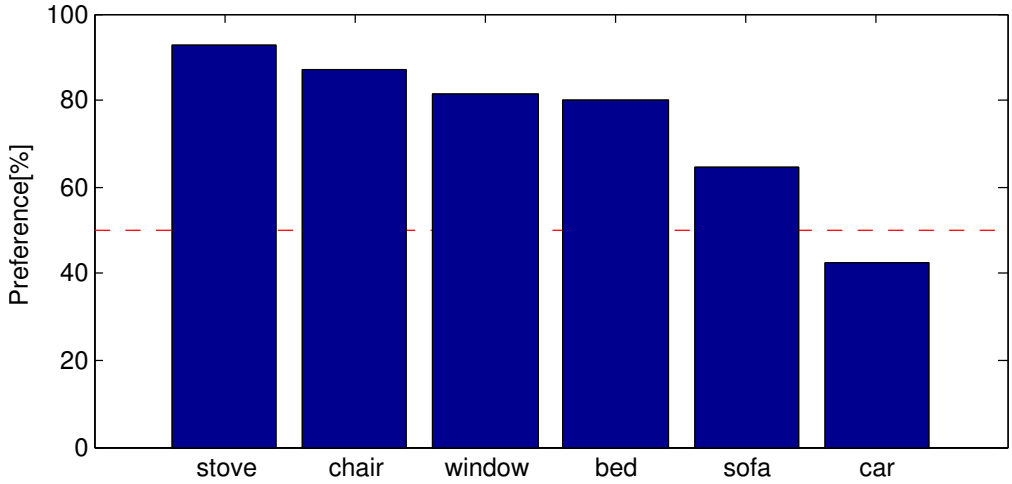


Figure 2.5: Human subject experiments for the Black Box Test: In our forced choice human subject task, annotators preferred our method on 75% of the cases compared to a baseline that can actually see what is behind the box. For objects like Stove and Bed which appear in more structured scenes our method produces better results compared to objects like cars that tend to appear in less structured scenes.

threshold with the best precision is used for comparison with our method in Fig. 2.4.

As shown in Fig. 2.4 the scene layout information encoded in our scene structures help object detection. In fact, our method outperforms DPM by large margins. Our mean Average Precision is 24.10 compared to 19.13 of DPM. Examples of detected objects along with their best scoring scene structures that produces those detections are shown in Fig. 2.3. It is interesting to see that some of the occluded objects have been correctly detected using our method. Scenes in our experiments vary from indoor to outdoor, from scenes with more dominant layout such as "Street" to scenes with very complex layout like "Dorm room". Our method can also localize stuff such as sky, grass, road, etc. See Fig. 2.3 for examples. We have also compared our method with the context modeling approach of [25] using their publicly available context tree model for object detection on SUN dataset. The average precision of [25] is 22.22% compared to 24.10% of our method.

2.3.2 The Black Box Test

Our scene structures provide a level of scene understanding that enables deep inferences such as the one used in Black Box Test. This test is inspired by Antonio Torralba's Context challenge [166]

Figure 2.6: Black Box Test: What is behind the black box? Our scene structures provide a level of scene understanding that allows deep inferences such as the one necessary to complete the back box test. Our method not only predicts what is behind the box but it also provides interesting detailed information about object poses.

and has also been studied in [102]. To be successful in BBT one needs to understand the scene category, and the context-specific layout and appearance models. For example, in Fig. 4.1, our model understands that because of the location of the wall and the table the expected chair should be in $3/4$ view. To test the performance of our model in BBT, we randomly select 120 images and black out examples of stoves, chairs, windows, beds, sofas, and cars in these images. We then use these blacked out images to find the highest scoring scene structures. To avoid artifacts due to the black box, we cancel out the effects (both appearance and deformation) of nodes that have any overlap with the black box. The best scoring scene structure contains information about the layout of the scene. This structure also encodes information about the missing part. For example, if there is a wall in a specific location and there is desk at the wall, there should be a chair with a specific pose at the desk. We use the part filters in the best scoring structure that has enough overlap with the black box to retrieve image patches. Fig. 4.1 and 2.6 show samples of image patches suggested by our method. Note that, our method not only finds the object of the right category but also its right pose.

To evaluate the quality of suggested patches by our method, we compare it against a baseline that can see what is behind the box. We also use the information about what objects we expect

Method	Accuracy	Method	Accuracy
LBP	18.12	GIST-color+SP+DPM [117]	43.1
HOG [117]	22.8	LPR [140]	44.84
ROI-GIST [125]	26.5	Ours	45.91
GIST-color [117]	29.7	Ours+LBP	47.64
DPM [117]	30.4	Ours+Texton	49.36
SSIM	33.45	Ours+LLC	49.38
Spatial Pyramid (SP)[87]	34.4	MLD Patches+GIST+SP+DPM [152]	49.4
Texton	35.98	Ours+SSIM	49.62
Object bank [93]	37.6	Ours+LLC+SSIM+Texton+LBP	52.41
LLC	37.53	BoP+IFV [76]	63.10
MLD Patches [152]	38.1	Midlevel elements+IFV [34]	66.87

Table 2.1: Scene categorization results on MIT indoor: The average per-class accuracy results on MIT indoor-67 dataset.

to see. This optimistic baseline uses the HOG2x2 descriptors of the whole image (including the object behind the box) to retrieve images of similar appearance. We then use the corresponding DPM model to obtain the best scoring patches among the retrieved images with similar appearance.

To compare the quality of the patches retrieved by our method to that of the baseline we perform a human subject forced choice task where subjects were asked to choose between the patches produced by our method and those of the baseline. For each image, we gather between 3 to 4 annotations. On average, the annotators preferred our patches to those of the baseline on 74.74% of cases. Fig. 2.5 shows the results of the human subject test on BBT for different categories of objects. For objects such as Stove and window that appear in more structured scenes like kitchen and rooms our method shows larger gain compared to objects like cars in less structured scenes such as streets. Fig. 2.6 shows qualitative results of the BBT.

2.3.3 Scene Category Recognition

We also exploit the scene structures to generate features that can later be used for scene recognition. To this end, we run all of our scene structure models on all test images and pick the k best scoring structures per scene category. We then record the structure scores of the k best structures as features and append the convolution scores of the objects in the best structures, and their normalized loca-

Method	Accuracy	Method	Accuracy
LBP	6.84	Ours	28.45
SSIM	21.06	Comb 15 features [179]	38
Texton	22.04	Ours+LBP	28.59
Object bank [93]	22.93	Ours+Texton	31.57
LLC	26.23	Ours+SSIM	31.58
RH [45]	26.9	Ours+LLC	33.45
HoG2x2 [179]	27.2	Ours+LLC+SSIM+Texton+LBP	35.95

Table 2.2: Scene categorization results on SUN: The average per-class accuracy results on SUN database.

tions. We also include relative locations of objects using the parents in the best scoring structures. Then, for each scene category, we train an SVM with HI kernel using this feature vector [37].

To evaluate the performance of the structures in scene categorization, we use SUN as well as the MIT indoor-67 dataset as test beds. In both these datasets we report the average per class accuracy obtained by our method and compare it with other state-of-the-art methods. We use half of the annotated images in each SUN category for learning scene structures. For categorization task on SUN, we randomly choose 100 (50 test, 50 train) images from the unannotated portion of each category which is not used in the training set of our structure learning. For the MIT indoor-67 we use the standard train/test split which is available in [125]. In scene recognition task on MIT indoor-67 dataset we use the scene structures trained on SUN. It is not possible to train our scene structures for MIT indoor 67 because object-level annotations are not available.

Tables 2.1 and 2.2 compare our results with the state-of-the-art methods in scene recognition. Following [179, 117, 152] we also combine our result with other state-of-the-art holistic features to encode global scene appearance information. We use Locally-constrained Linear Coding(LLC) [170], Self-Similarity (SSIM) [149], Local Binary Patterns (LBP) [114] and Texton [104] features. We observe that combining our feature vectors with other global features boosts our recognition performance in both MIT indoor-67 and SUN database. Our scene structures provide information orthogonal to the state-of-the-art holistic features.

According to the results of Table 2.1 scene structures produce promising results in scene categorization (accuracy of 45.91%). When combined with other state-of-the-art holistic features, the accuracy boosts to 52.41% .Note that our scene structures are trained on SUN dataset and tested

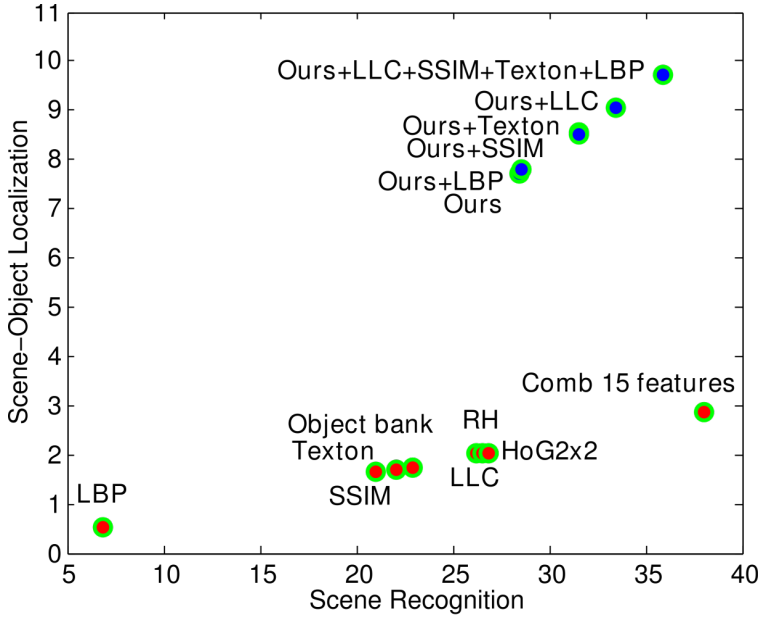


Figure 2.7: Scene recognition and object detection accuracy spectrum. Our method can recognize the scene label and detect objects simultaneously. For other methods since they do not have object detection step, we use object detectors of Object bank for their object detection step.

on MIT indoor-67 dataset whereas other methods have trained their models on the MIT indoor-67 dataset.

Table 2.2 compares our scene recognition results with state-of-the-art models on SUN database. We have provided the recognition accuracy of the state-of-the-art holistic features using our test and train splits. The best single-feature scene classification accuracy is 27.2% which is obtained by HOG2x2 [179]. As reported in [179], the recognition accuracy can be boosted up to 38% by combining 15 different feature types. According to Table 2.2, our method obtain an accuracy of 28.45% in SUN. After combining our method with four holistic features (LLC, SSIM, Texton and LBP) we have reached the accuracy of 35.95% which is on par with the combination of 15 features. Note that [76, 34] are using very high dimensional features to produce state-of-the-art results.

2.4 Discussion

We introduced a model that can link up the scene categories, the context-specific layout and appearance models for prominent objects and stuff in a scene. Our experimental evaluations show

that the learned scene structures are capable of localizing objects significantly better that state of the art detectors. We also show that our scene structure enables the level of scene understanding that is amenable to deep inferences such as those required in BBT. We compared our method with a baseline that sees what is behind the black box and show significant improvement in a forced choice human subject task. We also show promising results in scene recognition.

Our scene structures enables both scene categorization and object localization. To better understand the space of scene understanding methods in terms of both object localization and scene categorization we propose to consider a joint evaluation. Fig. 2.7 compares the performance of our method in the scene recognition and scene-object localization tasks with other state-of-the-art scene recognition methods. In this graph, the x-axis shows the scene recognition accuracy whereas the y-axis represents the scene-object localization precision. This can be measured as multiplication of scene recognition performance and scene-object localization. Most conventional scene recognition methods focus on the categorization and do not provide object-level information. To make a fair comparison, we use the available pre-trained detectors of OB [93] as the object detection step for state-of-the-art methods. We assume that for other methods we first run the scene recognition for each image and then use scene label for applying the corresponding object detectors on that image. As shown in Fig. 2.7, our method outperforms other state-of-the-art methods in terms of both scene recognition accuracy and object detection precision. In this plot an ideal method would be located on the top right corner. Our method takes advantage of the scene layout to improve the object localization and uses the object localization to improve scene recognition.

Chapter 3

IM2CAD: SINGLE-VIEW SCENE CAD RECOMPOSITION

In his 1963 Ph.D. thesis, Lawrence Roberts [129] demonstrated a system that infers a 3D scene from a single photo (Figure 3.2). Leveraging a database of known 3D objects, his system analyzed edges in the image to infer the locations and orientations of these objects in the scene. Unlike the vast majority of modern 3D reconstruction techniques, which capture only visible surfaces, Robert's method was capable of inferring back-facing and occluded surfaces, object segments, and recognized which objects are present.

While Robert's method was visionary, more than a half century of subsequent research in computer vision has still not yet led to practical extensions of his approach that work reliably on realistic images and scenes. One major limitation is the need for an accurate, a priori 3D model of each object in the scene. While a chair model, e.g., is not hard to come by, obtaining exact 3D models of every chair in the world is not presently feasible. A further challenge is the need to reliably match between features in photographs and CAD models, particularly when the model does not exactly match the object photographed.

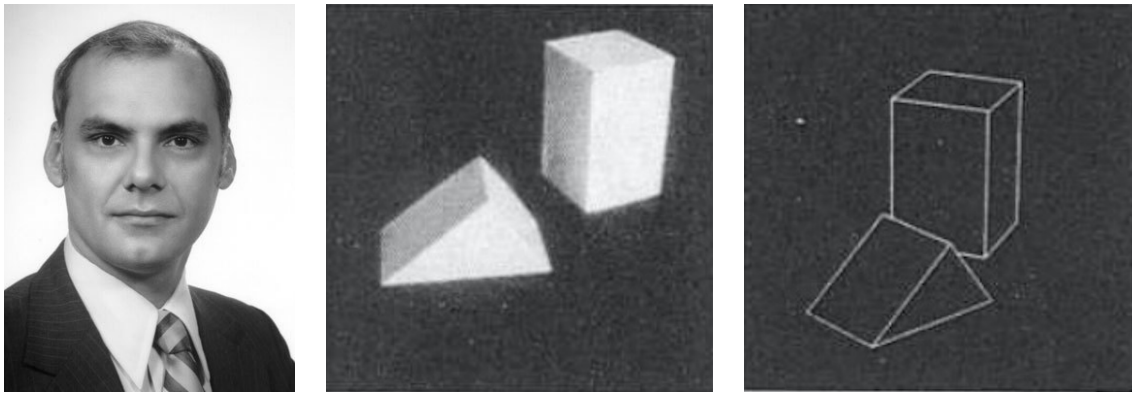
We therefore introduce a variant of Robert's original problem, that we call *IM2CAD*, in which the goal is to reconstruct a scene that is **as similar as possible** to the scene depicted in a photograph, where the reconstruction is composed of objects drawn from a database of available 3D object models. For example, the bed in Fig. 4.1 resembles but does not exactly match the one in the input photograph at left, as we did not have that specific bed in the database. While our results are not perfect, they represent a significant step forward to achieving Robert's vision on real-world imagery. Producing CAD models of real scenes has applications for VR/AR, robotics, games, and education.

Our work builds on a number of recent advances in the computer vision and graphics research

Figure 3.1: We introduce IM2CAD, a new system that takes a single photograph of a real scene (left), and automatically reconstructs a 3D CAD model (right) that is similar to the real scene.

community. First, we leverage ShapeNet [20], which contains millions of 3D models of objects, including thousands of different chairs, tables, and other household items. This dataset is a game-changer for 3D scene understanding research, and was key to enabling our work. Second, we use state-of-the-art object recognition algorithms [127] to identify common objects like chairs, tables, windows, etc.; these methods work impressively well in practice. Third, we leverage deep features trained by convolutional neural nets (CNNs) [83] to reliably match between photographs and CAD renderings [8, 79, 142, 65]. Finally, we build on recent research on room reconstruction [57, 88, 103].

Our main contribution is a fully automatic system that produces full-scene CAD models (room + furniture) from a single photo. While many of the technical ingredients of our system draw heavily from prior work (as detailed in the previous paragraph), we also contribute noteworthy technical advances on room modeling and scene optimization. Our room modeling approach produces significant improvement on standard benchmarks. Our novel full-scene optimization approach iteratively adjusts the placement and scale of objects to best align rendered photos with input images, operating jointly on the full scene at once, and accounting for inter-object occlusions. Our models include semantics (e.g. “table”, “chair”) segmented into objects, and take only a few bytes to represent, encoded as a collection of ShapeNet object IDs and transformations that define position, orientation and scale. We evaluate our performance on scene understanding using the datasets



(a) (b) (c)

Figure 3.2: Lawrence Roberts's (a) 1963 system took an input photo (b) and computed a 3D scene, rendered to a novel viewpoint (c).

of [57], LSUN [2], SUN RGB-D [153] and 3DGP [26]. We show significant improvements in the 2D and 3D room layout estimation as well as 3D object location using only single RGB images.

3.1 Related Work

The last decade has seen renewed interest in single-image 3D modeling, following the work of Hoiem et al., [61] and Saxena et al., [144]. Single-image modeling of indoor scenes has enjoyed significant recent progress, with a series of papers on room-shape estimation (floor, walls, ceiling), e.g., [57, 88, 103, 33, 128] that yield increasingly good results. Our approach for room shape estimation obtains competitive results. More recently, researchers have moved beyond walls, and toward approximating furniture in the room using *cuboids* [180, 187, 26, 54, 121, 145]. While the cuboid based approach avoids the need for object databases, the resulting models are primitive and do not accurately depict scene appearance.

Another closely related line of research is 3D object and pose recognition of chairs and other objects [8, 79, 142, 95, 65, 169, 10, 176]. These methods can produce very accurate alignment of a single object to a photograph or depth image. Our work leverages similar 3D object recognition techniques, combined with room shape estimation, to jointly solve for all of the objects in the room in a way that accounts for inter-object occlusions. Our work also builds upon recent advancements of research on object detection from single images [50, 127].

Researchers have explored a variety of techniques to automatically compute CAD scene models using non-photographic means, e.g., using example based approaches [41], utilizing text descriptions [21], and optimizing for furniture arrangements in a given space [184, 108]. These approaches rely on analyzing location and pose correlations between furniture types, based on analyzing databases of scene models. Collecting such data is a challenge, and therefore these approaches can greatly benefit from our solution which generates more comprehensive and plausible indoor models in a fully automatic fashion.

The closest works to ours are [143] and [97] which find the best matching 3D scene model to a given image. Our system is a significant advance in a number of ways. In particular, [143] requires a complete scene in the database that matches each image. Hence, their approach can be thought of as "3D scene retrieval," whereas we reconstruct each scene from scratch, using a database of furniture (not scene) models. The latter allows for a much broader range of reconstructable scenes. While [97] reconstructs the scene by placing individual pieces of furniture, they make a number of limiting assumptions (axis aligned furniture, no walls, easy-to-segment objects), operate on a much smaller database (180 models), and do not demonstrate as broad a range of results. Both of [143] and [97] use hand-crafted features, while our proposed method uses CNN feature which is learned end-to-end on just image data. Guo et al. [53] render a synthesized model of the scene using RGBD (depth) images while our method only uses RGB information. The synthesized rooms produced by [53] have low fidelity in terms of object details while we retrieve the detailed ShapeNet CAD model for each object.

In the context of 3D prediction, several previous approaches estimate the depth and surface normals of visible surfaces from a single image [35, 85, 10]. In contrast, our approach does not require dense surface normal estimation but is capable of estimating both visible and invisible surfaces through joint estimation of room and object CAD models.

3.2 Proposed Method

Our approach to reconstructing CAD models from an image (see Figure 3.3) is based on recognizing objects in the scene, inferring room geometry, and optimizing 3D object poses and sizes in the

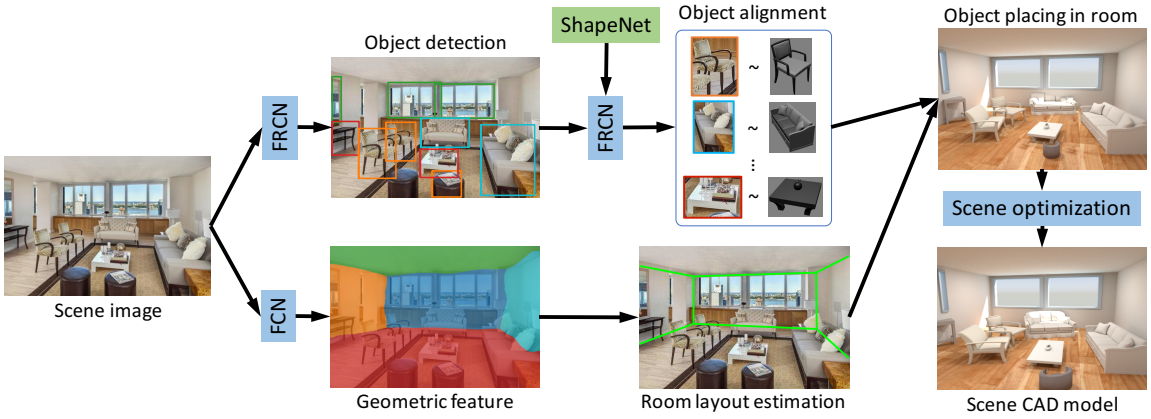


Figure 3.3: System overview: an input image (left) is processed through a series of steps to produce a scene CAD model (bottom right).

room to best match synthetic renderings to the input photo.

The proposed approach involves several steps, as follows. We first fit room geometry, by classifying pixels as being on walls, floor, or ceiling, and fitting a box shape to the result. In parallel, we detect all of the chairs, tables, sofas, bookshelves, beds, night tables, chest, and windows in the scene using state of the art object detection techniques. Wherever an object, e.g., a bed, is detected with high confidence, we estimate its 3D pose, by comparing its appearance with renderings of hundreds of beds from many different angles, using a deep convolutional distance metric, trained for this purpose. Finally, we optimize for the placement of all objects in the reconstructed room by optimizing the difference between the rendered room and the photograph. Our optimization approach operates on all objects jointly, and thus accounts for inter-object occlusions.

In the remainder of the section, we describe these technical components in detail: room geometry estimation, object detection, object alignment, and scene optimization.

3.2.1 Room Geometry Estimation

Humans are adept at interpreting the shape of a room (i.e., positions of walls, ceiling, and floor), even in the presence of significant clutter. Computer vision algorithms have also become increasingly good at this task in the last few years by following a paradigm introduced by Hedau et al. [57] and Lee et al. [88] in which a set of room shapes are hypothesized (typically 3D boxes), and eval-

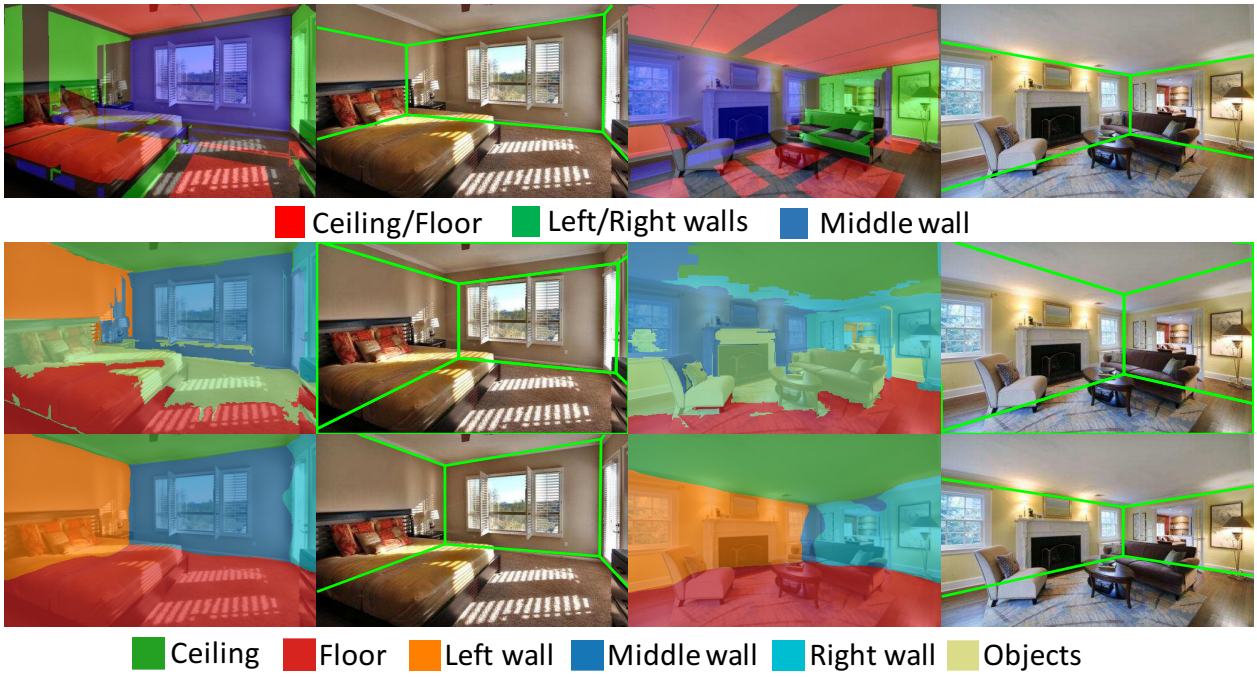


Figure 3.4: Geometric feature and room layout estimation. Results from (Row1) [88] and (Row2) [57]. Bottom row: our results.

uated using features in the image.

We improve upon previous approaches to room geometry estimation, by adopting an alternative approach for ranking the room 3D box hypothesis using deep convolutional features. Specifically, we train a network that estimates per-pixel surface labels (ceiling, floor, left, middle, and right walls). These features are analogous to the context geometric feature ("support", "vertical", and "sky") of [62].

Unlike [62] that learns the geometry features from hand-designed low level descriptors (e.g., color, texture, and other perspective cues) over superpixels, our method uses an end-to-end deep Fully Convolutional Network (FCN) [98], using VGG [151] and converting each fully connected layer into a convolutional layer with a kernel covering the entire input region. Finally, the weights are fine-tuned for the pixel-level labeling task. In this work, we produce the output dense score map of size $41 \times 41 \times 5$ given an input image of 321×321. We then use upsampling to produce a probability map with the same size of the input image. We trained the FCN network on the annotated indoor scenes in the LSUN dataset [2].

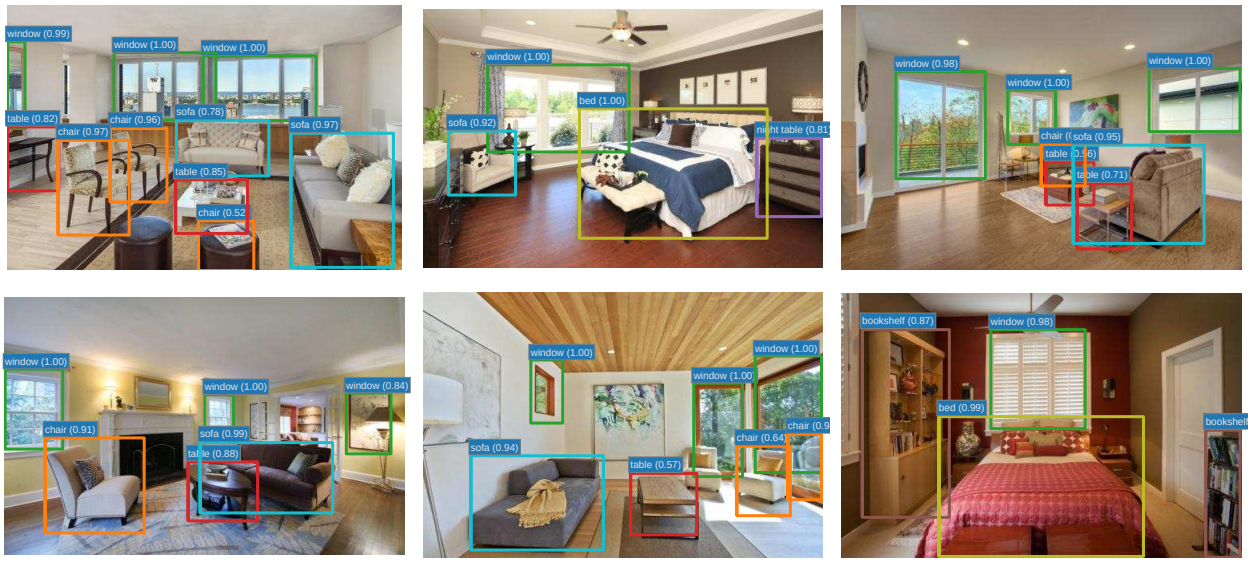


Figure 3.5: Object detection result on sample images. Each object category is shown with a different color. The numbers attached to boxes show the probabilities assigned to each detection.

A key advantage of the FCN-based architecture is that it integrates contextual information over the entire image. Whereas most methods use a "distractors" class [57, 103] to remove furniture from consideration, the FCN is able to use furniture as additional context, e.g., using the presence of a bookshelf or bed to infer the likely presence of an adjacent wall. We note that [103] also used a convolutional network, but rather than classifying surface orientations directly as we do, they estimate informative edges in the scene, and employ a second stage to iteratively re-label room surfaces and rank room box estimates.

3.2.2 Object Detection

The first step in our furniture modeling pipeline is to detect the presence of objects of interest in the image, and their 2D bounding boxes. While any number of object detectors can be trained, we focused specifically on the following: chair, table, sofa, bookshelf, bed, night table, chest, and window.

Object detection is an area that has seen explosive progress in the last several years, and existing methods work impressively well. In particular, we use the state-of-the-art Faster-RCNN [127] deep

network for detection. This network performs two steps to detect objects. First it produces object region proposals, and then it computes the likelihood of each category for the proposed objects using deep convolutional layers. The region proposal layer produces bounding boxes of different scales and aspect ratios. This network is initialized with pre-trained models from large scale object recognition tasks (ILSVRC2012) [72]. The network weights are then fine-tuned for the object proposal and object detection tasks by minimizing an objective function for a multi-task loss on bounding box regression and object misclassification. The trained network is then able to produce bounding boxes with object categories for any image. The network output also includes an object score which shows the probability of that particular object in the bounding box. Greedy non-maximum suppression (NMS) is used to obtain a single peak detection for each object, remove low scoring detections that overlap with higher scoring object bounding boxes.

Our Faster-RCNN implementation uses VGG16 [151] architecture. We further fine-tune the weights of this network for the object detection task on our eight furniture categories using three publicly available datasets, namely SUN2012 detection dataset [179], ImageNet detection challenge dataset [133], and the window category of Rent3D dataset [96]. We show detection results on a sample of our images in Figure 3.5.

3.2.3 CAD Model Alignment

The object detection results from Section 3.2.2 identify the presence of a "chair" (e.g.,) in a certain region of the image with high probability. Now we wish to determine what *kind* of chair it is, its shape, and approximate 3D pose.

Inspired by [8], we solve this retrieval problem by searching for 3D models that are most similar in appearance to the detected objects in the image. Specifically, we consider all 3D models in the ShapeNet repository [20] associated with our object categories of interest, i.e., chair, table, sofa, bookshelf, bed, night table, chest, yielding 9193 models in total. Each 3D model is rendered to 32 viewpoints, consisting of 16 uniformly sampled azimuth angles and two elevation angles (15 and 30 degrees above horizontal).

Robust comparison of photos with CAD model renderings is not straightforward; simple norms

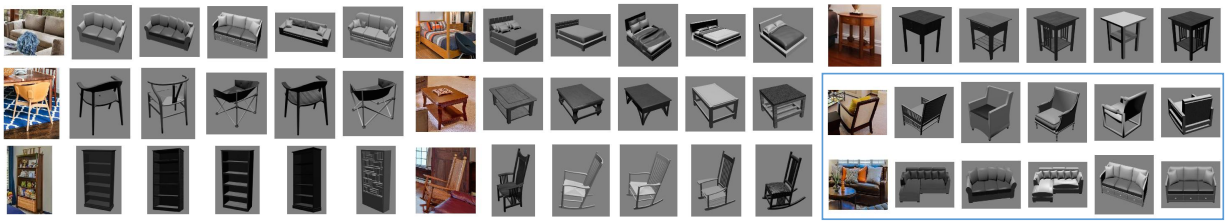


Figure 3.6: Results of the top five aligned CAD models retrieved for the given object detection bounding box. The retrieved models have similar style and pose with the given object. Last two rows on the right column show failure cases: (Row1) visual feature confusion between different poses of the chair, and (Row2) heavy occlusion of sofa by table has made it visually similar to an L-shaped sofa.

like L2 do not work well in practice, due to differences in shape, appearance, shading, and the presence of occluders. We achieve good results, once again, by using convolutional nets (see Figure 3.6); we compute deep features for each of the rendered images and the detected image bounding boxes and use cosine similarity as our distance metric. More specifically, we use the convolution filter response in the ROI-pooling layer of the fine-tuned Faster-RCNN network [127] described in Section 3.2.2. A benefit of using the ROI-pooling layer is that the length of its feature vector does not depend on the size and the aspect ratio of the bounding box, thus avoiding the need for non-uniform rescaling (a source of artifacts in general). Choosing the rendering that best matches each image object detection yields an estimate both for the best-matching CAD model and its approximate 3DOF orientation.

3.2.4 Object Placement in the Scene

Equipped with a set of CAD models and their approximate orientations, we now wish to place them in the reconstructed room. This placement need not be exact, as we will further optimize it in a subsequent step, but should be a reasonable initial estimate. To this end, we first estimate the intrinsic camera parameters (K) and camera rotation (R) with respect to the room space using three orthogonal vanishing points [57], and choose one of the visible room corners as the origin of the world coordinate system. If none of the corners are visible, we choose the origin to be the intersection of the visible wall edge with the floor.

The ShapeNet 3D models are normalized with a bounding box corresponding to a unit cube.

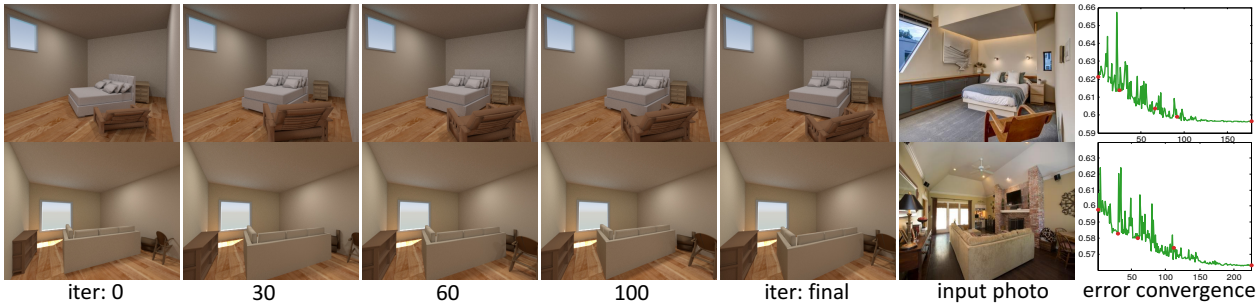


Figure 3.7: Results of the joint scene optimization step. (Column 1) The initial object placement in the scene. (Columns 2-5) Rendering of the scene in sample iterations during optimization. (Column 5) The last iteration of optimization. (Last column) The objective function error and the optimization convergence. The objective function minimizes dis-similarity between the real and the rendered image. Red dots show the sample iterations that are shown above.

Based on the alignment procedure from Section 3.2.3, we can determine the input photo pixel locations corresponding to each of the eight corners of this cube. We can find the object location and scale in the x and y (parallel to ground plane) directions by intersecting the ground plane with the ray casted from the camera center through the input image pixels corresponding to the bottom four corners of the aligned CAD model cube. To compute the object scale along the z axis, we compute the ratio between the length of the four vertical edges of the projected cube and the length of those edges from the ground plane to the intersection of those lines with the horizontal vanishing line. Note that the height of the vanishing line is equal to the camera height.

We treat windows as a special case, as they are attached to walls instead of floor. To place windows, we find the intersection of the window bounding box from object detection with each of the walls and assign the window to the wall with which it has the largest overlap. The window's detected bounding box in the image back-projects to a quadrilateral on the assigned wall. The pose and location of window is determined by the largest axis-aligned rectangle on the wall plane contained within that quadrilateral.

3.2.5 Scene Optimization via Render and Match

The placement procedure in Section 3.2.4 is sensitive to several sources of error including the quantization of object orientations, ground plane misregistration, occlusions, and other factors, which

can lead to erroneous estimates of object pose and scale. We therefore introduce an optimization in which the configurations of all objects in the scene are jointly aligned. A benefit of this procedure is that it properly accounts for inter-object occlusions, and yields more accurate estimates for object location, scale, and orientation.

After estimating the 3D room geometry and the initial placement of the objects in the scene, we refine our object placements by optimizing the visual similarity of the rendered scene with that of the input image. To this end, we solve an optimization problem where the variables are the 3D object configurations in the scene and the objective function is the minimization of the cosine distance between the convolutional features obtained from the camera view rendered scene and the input image.

More formally, suppose we detect objects $\{O_1, ..., O_k\}$ in the scene. The placement of each object O_i is represented by its (x, y, z) location, scale along the x, y and z axis as well as the rotation. The variables for all N objects are concatenated into a $7N$ parameter vector. Given a parameter vector, we can generate the rendered image of the scene, denoted I^*. The cost function used in our optimization tries to minimize the cosine distance between I^* and the original input image I:

$$\min \Phi(I^*, I) = \frac{1}{|\mathcal{C}|} \sum_{C_i \in \mathcal{C}} 1 - \frac{C_i(I^*) \cdot C_i(I)}{\parallel C_i(I^*) \parallel \parallel C_i(I) \parallel} \tag{3.1}$$

We model the feature vector of an image by using the outputs of all convolutional layers [1]. In the above equation, $\mathcal{C}$ refers to the set of conv layers in the network and C_i is the feature vector obtained from the ith layer. The total cost function is the average similarity of all layers. The convolution filters in higher layers of the network provide abstract shape features while the details of the images such as edges and corners appear in the features obtained from the lower layers of the network. The features in higher levels have larger receptive fields, and can therefore cope with larger displacements, and help the optimization to converge in the first iterations when the initial estimates are far off. Similarly, the lower convolutional layers play a greater role in later

[1] We use conv1-1, conv1-2, conv2-1, conv2-2, conv3-1, conv3-2, conv3-3, conv4-1, conv4-2, conv4-3, conv5-1, conv5-2 and conv5-3 layers in the VGG network

iterations, to help the objects converge with more precision. In this way, the network provides a natural coarse-to-fine structure to the optimization.

Since our objective function is not differentiable we use COBYLA [122], a derivative free numerical optimization method, deployed in a Python optimization package. We found this procedure to work very well in practice. Figure 3.7 shows the convergence of the method for example scenes.

3.2.6 Coloring CAD models

We use a medoid color of each object in the input image for scene optimization (Section 3.2.5) and visualization. The process is as follows. First, we project the best aligned CAD model of an object onto its bounding box in the image. We then find the median value of each color channel separately, and take the closest color which appears within the mask. We also compute the medoid color for each wall of the room using a similar approach. We compute the mask of each wall through the room geometry, and exclude the bounding boxes from detected objects. This approach works well in practice for scene optimization and creates visually pleasant renderings without falling into the uncanny valley [148] (see results in Figure 3.9).

3.3 Experiments

We evaluate our IM2CAD system both qualitatively and quantitatively on scene understanding benchmarks.

3.4 Qualitative Evaluation

We evaluated the proposed IM2CAD system with 100 real world indoor images collected from "Zillow Digs" [191]. These images are living room and bedroom shots as our training object categories are chair, table, sofa, bookshelf, bed, night table, chest, and window, i.e., typical bedroom and living room furniture. We cover a variety of room styles from traditional to modern with various furniture arrangement, complexity, and clutter that are representative of real world scenes. We also show example results on the SUN RGB-D dataset.

Method	Pixel Error(%)
Lee et al. [88]	24.70
Hedau et al. [57]	21.20
Del Pero et al. [121]	16.30
Gupta et al. [54]	16.20
Zhao et al. [188]	14.50
Schwing et al. [146]	13.59
Ramalingam et al. [126]	13.34
Mallya et al. [103]	12.83
Dasgupta et al. [33]	9.73
Ren et al. [128]	8.67
IM2CAD	10.15

Table 3.1: Room layout pixel misclassification error on Hedau [57].

Our IM2CAD approach consistently produces reasonable results on most of the test images. Figures 3.9 and 3.10 is representative of the top 30% of our results, where most pieces of furniture are detected, represented using well-matched CAD models, and properly posed. Typical failure results are shown in Figure 3.8. Our failures are rarely catastrophic, and generally fall into the category of some furniture items being omitted or misplaced.

Object pose estimation can sometimes get stuck in local optimal. Notice that the foreground chair in Figure 3.8(a) is in an incorrect pose while the chair legs are aligned almost perfectly to the image. The last two rows of Figure 3.6 demonstrate cases where the visual similarity fails to retrieve appropriate CAD models. Heavily occluded objects impose additional challenges. Notice the missing chair and coffee table in Figure 3.8(a) and (b). If the room shape is not perfectly cubic (Figure 3.8(c)), room layout estimation can fail to recover the true room shape. The windows can be confused with paintings as they have very similar visual features (see Figures 3.9 and 3.10). Both windows and paintings typically appear as glassy and shiny rectangular shapes on a wall.

We use Caffe [72] to implement and train our deep networks. We learn the weights of our FCN network for the room geometry estimation using stochastic gradient descent with initial learning rate of 0.001 and weight decay of 5e-4. We train our network in 45 epochs where the learning rate decreases every 15 epochs. For the object detection we use same threshold for all object categories and only keep the detection boxes with scores higher that 0.5.

Method	Pixel Error(%)
Hedau et al. [57]	24.23
Mallya et al. [103]	16.71
Dasgupta et al. [33]	10.63
Ren et al. [128]	9.31
IM2CAD	10.04

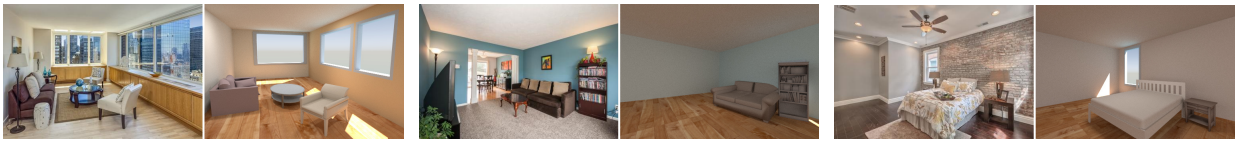
Table 3.2: Room layout pixel misclassification error on LSUN [2].

Object detection and geometric feature extraction are processed on a Titan X GPU, while room layout sampling and object pose estimation are computed on CPU. For a typical input image of size 300×500, the computational time is approximately 0.15 seconds for object detection, 0.3 seconds for geometric feature extraction, 8 seconds for room layout sampling and ranking, and 10 seconds for object placement. Scene optimization is an iterative process where each iteration takes about 1 second. We set the maximum number of iterations to be 250. The overall CAD model creation process finishes within 5 minutes.

To produce final room renderings with global illumination, we use Blender Cycle Render Engine [13], with fixed lighting consisting of distant sunlight from the top right point and five area lights on the ceiling. The final rendering process takes about 15 minutes with global illumination.

3.5 2D Room Layout Estimation

To evaluate the accuracy of room layout estimation, we compute the pixelwise difference between the predicted layout and the ground truth layout labels, averaged across all images as the evaluation metric. We evaluated on the test split of [57] dataset (we do not use their training split). Our FCN features (without 3D box estimation) achieve a 12.4% pixel misclassification error compared to 28.9% of [62] on the leading benchmark dataset [57] (see Figure 3.4). When combined with a box-fitting step of [57, 88], we achieve competitive result of 10.15% error compared with [33] and [128] as shown in Table 3.1. More specifically, we improve the reported result of [103] by 2.7%, [33] by 3.1%, and [128] by 4.2%. As an ablation study to evaluate the effect of different room hypothesis estimation approaches, we tested our approach while being combined with either of [57] or [88] and we obtain an error of 11.02% and 11.13%, respectively.



(a) (b) (c)

Figure 3.8: Failure cases: inaccurate chair pose (a); mis-detection of a chair (a) and table (b); non-cubic room shape (c).

We also evaluated performance on the task of room layout pixel misclassification using the LSUN dataset [2]. As summarized in Table 3.2, IM2CAD outperforms previous approaches [57, 103] significantly as well as [33] and obtains competitive results with recent approach of [128].

3.6 3D Room Estimation and Scene Understanding

Our IM2CAD system is also applicable for 2D and 3D scene understanding as well as room layout estimation. For evaluating our performance in scene understanding tasks, we use the SUN RGB-D dataset [153]. This dataset contains images captured from different view points, some of the images have low field of view and a considerable number of them are captured from highly cluttered scenes. Note that, although the SUN RGB-D dataset contains the depth data for all the images, we do not use the depth information at either train or test time, but estimate the 3D room geometry as well as object layout using only single 2D images. We use the test split for the bedroom and living room scene categories with a total of 484 images.

3.6.1 3D Room Layout Estimation

3D room layout estimation enables precise reasoning about free space versus spaces occupied by objects. In the absence of depth data, this task is challenging as it requires reasoning about room geometry from 2D images. Our 3D room layout estimation is evaluated by computing the intersection over union (IoU) between the predicted and the ground truth free spaces. Following [153], we assume empty rooms without objects and define a voxel grid of $0.1 \times 0.1 \times 0.1$ meter. The effective voxels are the ones that are located within 0.5 and 5.5 meters from the camera and are inside the field of view. We check whether each voxel is inside the 3D room polygon and compute the intersection and union computed by counting the 3D voxels. Table 3.4 summarizes our obtained

Method	SUN RGB-D		3DGP	
	voxel IoU	mAP	voxel IoU	mAP
3DGP [26]	38.7	42.1	38.4	59.7
IM2CAD (w/o optim.)	46.1	74.7	53.5	86.6
IM2CAD (w/ optim.)	49.0	75.6	53.8	86.6

Table 3.3: 3D scene free space prediction (voxel IoU) and object localization (mAP) results on SUN RGB-D [153] and 3DGP [26] datasets (higher is better).

results. Our method outperforms [57] by 13.2%.

3.6.2 Scene Understanding

The task of scene understanding integrates recognition and localization of all the objects as well as estimating the room structure. Compared to the task of 3D room estimation, this is a more challenging task as it requires detecting non-free spaces occupied by the objects. We compute the distance between the projection of the box centroid on the ground plane for all pairs of predicted and ground truth objects with the same label. We sort the distances in ascending order for each available pair and choose the pair with the shortest distance while the two boxes are marked as unavailable. We compute the precision and recall by varying the distance threshold and use the mean average precision as object localization metric.

Free space prediction is evaluated in a similar manner to the 3D room layout. The visible 3D voxels for the free space inside the room polygon but outside any object bounding box is computed and then the IoU between the free space prediction and the ground truth is computed. Table 3.3 shows the results of free space prediction and object localization on SUN RGB-D dataset. We compare the performance of our approach for scene understanding with [26]. IM2CAD obtains superior results compared with [26] in both metrics i.e., 33.5% boost in the mean AP and 11.7% in scene free space prediction. We compare our results before and after applying scene optimization (Section 3.2.5). Our scene optimization approach results in improved accuracy for the task of scene understanding.

We also report IM2CAD performance on the dataset presented in [26] which we call 3DGP. We use 372 images from the test split of living room, bedroom and dining room categories. How-

Method	SUN RGB-D	3DGP
Hedau et al. [57]	49.4	47.3
IM2CAD	62.6	63.2

Table 3.4: 3D room estimation results using voxel IoU on SUN RGB-D [153] and 3DGP [26] datasets (higher is better).

ever, we do not train our model on the 3DGP training set. To estimate the ground truth camera parameters, we compute the pseudo ground truth vanishing points by using the annotated ground truth edges corresponding to the three vanishing points following the experimental setting of [26] for 3D scene evaluation. We evaluate on the three tasks of 3D room layout, whole scene free space prediction, and object localization. These results are summarized in Tables 3.4 and 3.3. For the task of 3D room estimation, IM2CAD significantly outperforms [26] by 15.9%. In the free space prediction task, IM2CAD obtains significantly better results than 3DGP in both voxel IoU and mean AP criteria.

3.7 Discussion

This chapter presents a fully automatic system that reconstructs a 3D CAD model of an indoor scene from a single photograph, by utilizing a large database of 3D furniture models. It estimates room geometry, and detects and aligns objects in the image with accurate 3D poses. We introduce novel approaches for room modeling and scene optimization, that are keys to the success of our system. We evaluate on a wide range of living room and bedroom photographs with a variety of home styles. The results demonstrate the effectiveness of our approach in creating 3D CAD models that faithfully resemble the real scenes. With the abundance of indoor photos available online, our system is applicable to produce a large database of indoor scene models. Our approach obtains significant improvement on the 2D room layout estimation and 3D scene understanding benchmarks.

Our system does have limitations suggesting a number of areas for future work. We assume the room geometry in the image can be modeled with a cube. Working with complicated room geometry is an area of future improvement. Understandably, heavily occluded objects impose

challenges. We assume objects are always on the ground plane (e.g., chairs and beds) or attached to walls (windows), posing a lamp on a table would require extension of our work. Incorporating more object types would lead to more general scenes and room types (e.g. kitchens and bathrooms).

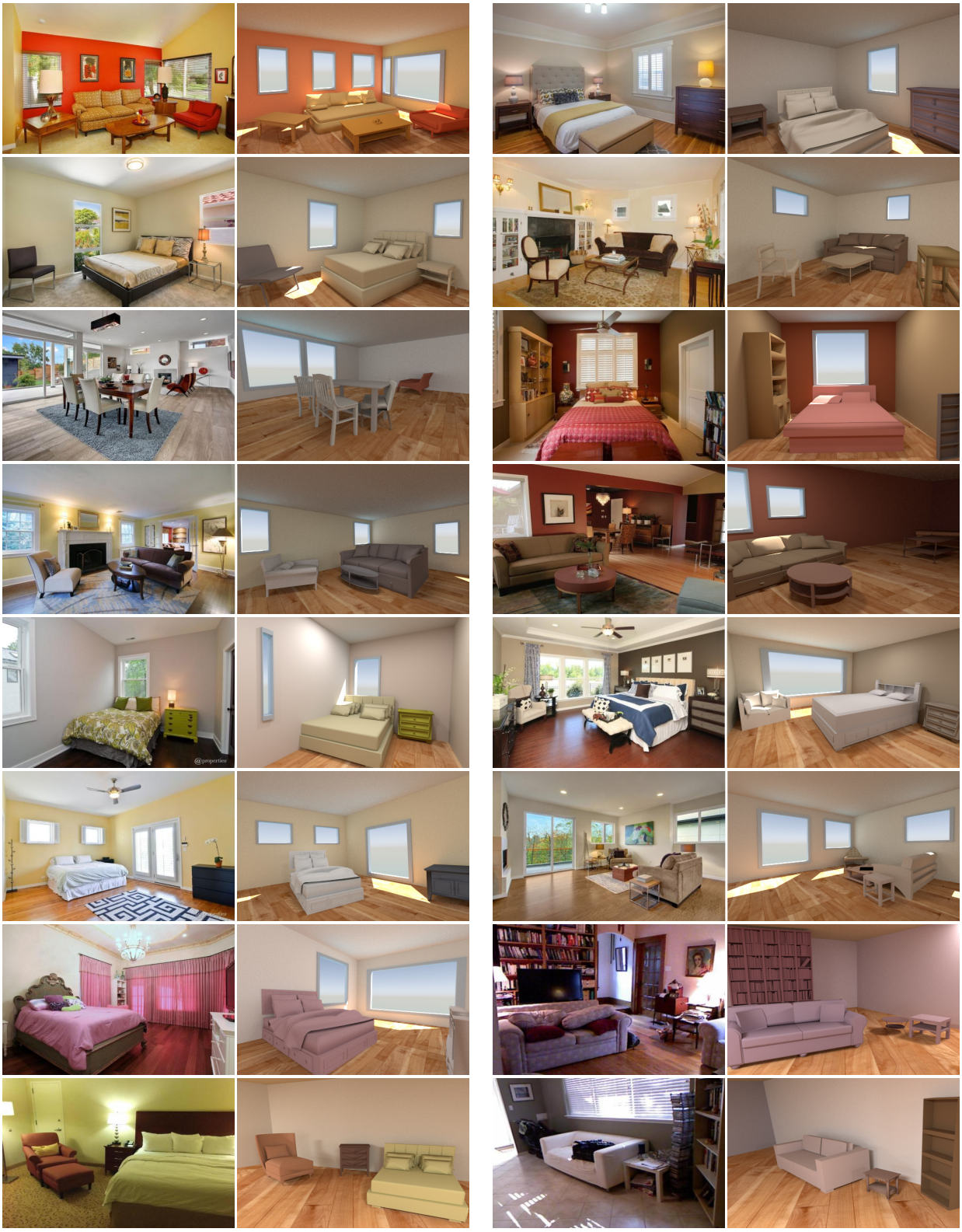


Figure 3.9: The reconstruction results. In each example the left image is the real input image and the right image is the rendered 3D CAD model produced by IM2CAD. Last three example results on the SUN RGB-D dataset.

Figure 3.10: The reconstruction results. In each example the left image is the real input image and the right image is the rendered 3D CAD model produced by IM2CAD.

Chapter 4

SCENE RECOMPOSITION BY LEARNING-BASED ICP

3D scene reconstruction is a fundamental challenge of computer vision. Most reconstruction techniques focus on estimating surface geometry, in the form of meshes, point-clouds, voxels, or other low-level representations. Suppose that you had access to a database of 3D models of every object in the world; then you could generate a scene model by identifying which objects are in which locations and placing them there. We call this variant of the reconstruction problem *scene recomposition*. While previously such an approach was not feasible at scale, the advent of large CAD repositories like *ShapeNet* [20] and *SUNCG* [155] begins to make scene recomposition tractable for real-world scenes.

Scene recomposition has a number of advantages over scene reconstruction. First, whereas reconstruction methods often generate holes and capture only visible surfaces, recomposition yields more complete models, including back-facing and hidden geometry (see Figure. 4.1). Second, CAD models are clean, segmented, and hand-optimized, and thus better suited for applications like games, VR, robotics, and etc. And third, recomposed models can be easily edited by moving objects around, replacing objects, and often come with semantic labels and annotated parts.

Recomposition is not a new idea, dating back to the first "blocks world" methods from the 1960s [129], which also employed a model-based approach. More recent examples include SLAM++ [142] and IM2CAD [69]. We introduce the first fully automatic 3D scene recomposition method that takes an RGBD sequence as input and produces a model of the scene composed of best-matched CAD models from *thousands* of 3D CAD models. In addition, we propose a novel learning-based ICP technique for aligning CAD models to scanned geometry.

Aligning 3D object models to depth scans is a classical problem in computer vision and geometry processing, and a staple of many practical applications spanning mapping, robotics, and

Figure 4.1: Given an RGBD sequence from a moving camera, we produce a 3D CAD ***recomposition*** of the scene. While a fused reconstruction (top) contains holes and noisy geometry, our recomposition (bottom) models the scene as a set of high quality 3D shapes from CAD databases.

visualization. The Iterative Closest Point algorithm (ICP) [12] works by alternating between finding the closest points between the model and the depth image (or other sensor data), solving for the best transformation that aligns the two point sets, and iterating until convergence. ICP and its variants can robustly converge when the model is initialized close to the solution, but suffer without good initialization or in the presence of significant occlusions and scene clutter. Matching discriminative local 3D features [74, 43, 165, 136, 137] is an alternative which relaxes the initialization requirements to be more robust, but is less effective for matching synthetic CAD models to real scenes, where 1) the models are simple and feature-poor, and 2) the shapes of the model and real object only approximately agree.

To address these problems, we cast the alignment problem of 3D object CAD models to RGBD scans in a reinforcement learning framework which we call Learning-based ICP (LICP). LICP is trained entirely on synthetic scenes without requiring ground truth annotation of object pose alignment or keypoint pairs in real scenes. Despite this fact, our quantitative evaluations show that LICP outperforms prior methods in real scene experiments. We demonstrate the application of our approach for fully automatic scene recomposition of complex real room environments populated with different types of furniture exhibiting a high degree of occlusion. Our recomposed scenes are

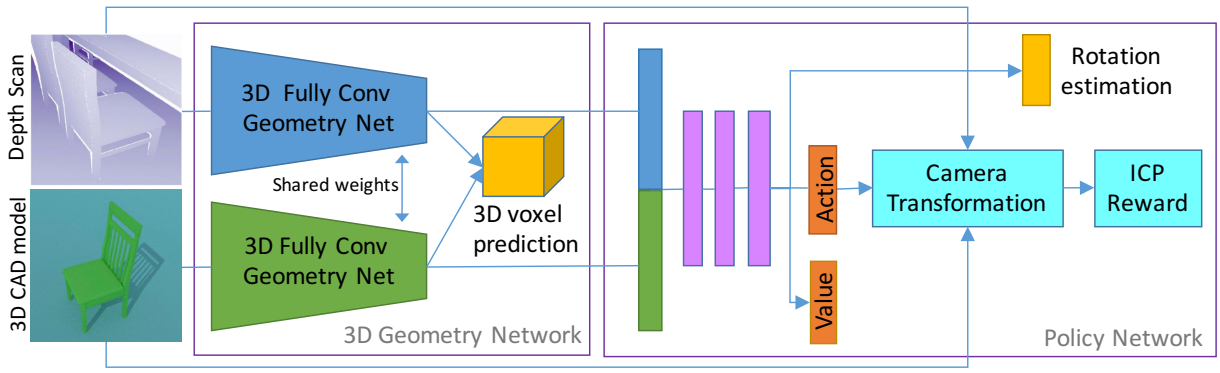


Figure 4.2: **LICP Network Architecture**: The input to our network consists of a scanned object paired with a reference CAD model (left) which are processed by the **geometry network** (middle). The geometry network is trained via a supervised loss to predict 3D voxel labels (yellow). The input representations are then concatenated to form the input to the **policy network** (right) which is trained via policy gradient to predict action distribution and value (orange) in order to maximize an ICP reward function. An auxiliary reward function (yellow) that estimates the rotation degree of the 3D CAD model with respect to the scanned shape is also incorporated.

comprised of best-matched CAD models from thousands of 3D CAD models in ShapeNet.

4.1 Related Work

Inferring 3D object pose and scene recomposition relates to prior works in computer vision and graphics, as follows.

ICP: ICP was introduced by [23] and [12] solves for the transformation between two point sets. Much research has been devoted to improving this method over the years, including [132, 23, 29, 131]. Where prior methods focus on feature representation and optimization, we introduce a data-driven and learning-based approach.

3D shape alignment, 3D features and keypoint matching: An alternative to dense alignment via ICP is to detect robust features (aka *keypoints*) to facilitate shape alignment. [74] proposed *spin images* and used RANSAC for shape alignment. Other examples of geometric descriptors are Geometry Histograms [43], Signatures and Histograms [165], Feature Histograms [136] and many more available in Point Cloud Library [137]. However, keypoint methods can be sensitive to noise and do not always perform well particularly for matching CAD models which are often piece-wise planar and feature-poor. Local features are not robust to symmetries (e.g., all chair legs may have

the same features). Model-fitting approaches, also known as registration approaches, try to align an input with a training model but without using descriptors [12, 73, 177]. These approaches do not incorporate learning so that they do not benefit from large amount of data to gain robustness in keypoint detection and matching. Techniques like [55, 154, 47, 86, 112, 106, 81] estimate complete scene geometry by fitting instance-level 3D mesh models to the observed depth map. Compared to these methods, our model learns global models over CAD shapes to align poses.

A recent approach for CAD to scan alignment [9], requires *manual* annotation and curation of a large dataset of 3D keypoint correspondences between object CAD model and real scans. [9] uses the collected annotation data for learning correspondences between CAD models and scans. However, our proposed method only uses available synthetic data during training without needing annotated keypoint correspondences in both CAD and real scan domains. While not needing annotated data, our proposed method performs well in the real scene scenarios at the test time. Also to find correspondences at the test time, [9] uses the ground truth object set or a limited set of CAD models, whereas our method can find corresponding CAD models from an *unconstrained* set of objects.

Object level RGBD scene reconstruction: Like our approach, SLAM++ [142] performs room scale semantic object reconstruction using KinectFusion [113] followed by 3D shape recognition. Also, SLAM++ only uses a handful of 3D object models (vs. the thousands in ShapeNet), and does not incorporate a learning-based approach.

3D CAD scene model generation: Several prior works proposed methods of generating CAD-based room models using a variety of techniques. Example of these approaches are CAD from text descriptions [21], example based methods [41] or optimizing furniture arrangements in a space [184, 108]. Scene models can also be generated by matching 3D objects to a given image [143, 97], rendering a low fidelity synthesize model using RGBD images [53] or recomposing each scene by analyzing layout and furniture and jointly optimizing their placements [69].

Voxel prediction and shape completion: Single object shape completion and voxel category prediction has been studied by several authors [130, 163, 178]. In this chapter, we utilize voxel category prediction as an auxiliary loss function to learn 3D representation, but the output of our

model is 3D CAD model with correct pose instead of a voxel grid. As such, we do shape completion, but compared to prior voxel-wise shape completion methods, our method produces CAD meshes with shape semantics.

Shape pose estimation: Single object 3D pose recognition from a photograph or depth image is also related to our work [8, 79, 142, 95, 65, 169, 10, 176]. However our approach differs since we learn the best points to match by conditioning on object viewpoint.

Deep feature learning and deep reinforcement learning: A number of researchers have used deep neural networks to learn 3D feature representations [155, 185, 31]. Recently, deep Reinforcement Learning (RL) approaches have gained considerable attention due to their success in learning efficient policies to play games [110, 150] and obtaining promising performance in robotics [52, 6]. Part of the success of deep RL is its applicability in solving black-box non-differentiable optimization problems [160]. Our approach for selecting the correct camera transformation action based on score approximation is closely related to a class of RL techniques called policy gradients [11, 173]. In our method, we have a non-differentiable reward function based on ICP scores of two point clouds and we want to learn the policy that results in receiving maximum reward by using stochastic gradient decent and following a policy gradient update rule.

4.2 Proposed Method

We begin by describing our learning-based ICP (LICP) approach. Then we explain how to use LICP for recomposing a scene from an input point cloud. For scene recomposition, 3D object detection and 3D semantic segmentation are incorporated for extracting the object instances in the scene. Then, LICP is applied to match and align 3D object CAD models to segmented regions of scene geometry.

LICP seeks to estimate the transformation parameters of a scanned rigid object in natural real scenes. This is a challenging task due to inter-object occlusion, self-occlusion and clutter. We train a deep neural network that takes in a scanned shape (query) paired with a reference CAD model as input and learns to infer the transformation that should be applied to the reference CAD model to best align its point cloud with the query scan (Figure. 4.2). To learn such a model, we

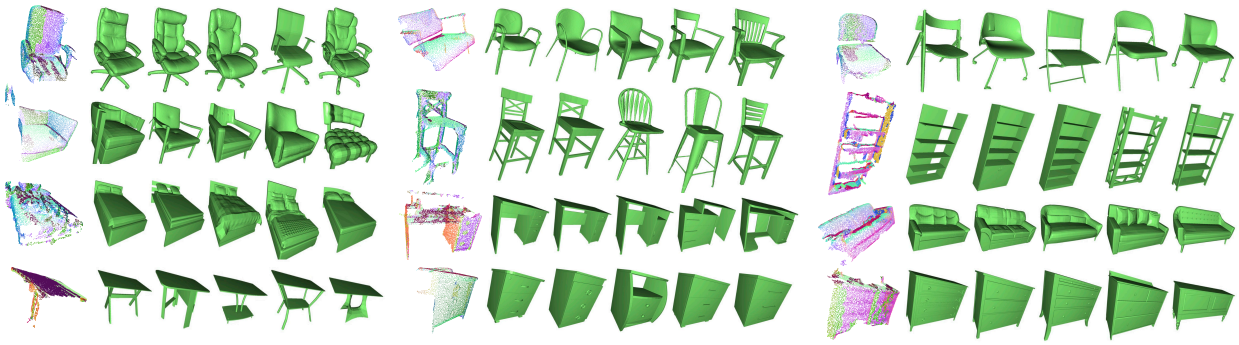


Figure 4.3: Top retrieved CAD models for each object instance segmentation as query. Point cloud query is color-coded with surface normal.

take advantage of the fact that we can apply any transformation on the reference CAD object and simulate a depth map (point cloud) of the transformed object using ray tracing. To this end, we generate a training set of 3D scans, each paired with a 3D object with known 6DoF parameters. We pose the learning problem in a Reinforcement Learning (RL) framework where the task is to predict the best action that should be applied to the reference shape such that we can generate the queried input scan. Each action encodes a possible 3D transformation that will be applied to the reference 3D shape. By applying each action, we can produce a reward that reflects how much the transformed 3D shape matches the queried shape.

4.2.1 Shape Alignment by Deep RL

We pose the problem of 3D pose estimation with respect to a reference shape in an RL framework. Suppose we have a reference shape X^r which is presented in a reference pose P^r. Using this reference shape, we want to learn to predict the 3D pose of any queried 3D object scan X^q that is being cropped out of a complete scene scan. The 3D scan can contain a high amount of occlusion, complicating the alignment process. For representing 3D models, we use a voxel-based 3D feature representation function $\Phi(X)$ for both reference and query shapes. The goal of the RL agent is to select transformation actions to the queried object which maximize the expected sum of future rewards. Our reward function, shows the matching score of the queried shape with the reference shape if point-to-point local closest point alignment is performed (details in Section 4.2.2).

We consider a Markov Decision Process (MDP) defined by states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ where each 3D rotational camera transformation is an action a that the RL agent can potentially apply to a 3D shape. We define each pair of query object scan and reference object scan captured with camera transformation ϱ as a state $s : (\Phi_\tau(X^q), \Phi_\varrho(X^r))$. By taking each camera transformation action a, each state transits to a new state by capturing the 3D scan of the reference object X^r. We uniformly discretise the action space of each dimension of rotation degrees into a list of 32 bins where each bin corresponds to a rotation transformation with a fixed angle. Reducing the action space complexity by discretization accelerates learning and makes it more sample efficient.

4.2.2 ICP-based Rewards

Each training instance is composed of a 3D point cloud of a scanned query object $\Phi_\tau(X^q)$ as captured with an unknown camera pose τ paired with a reference 3D object X^r. After choosing an action a, we apply the corresponding camera transformation a and render the transformed point cloud $\Phi_a(X^r)$ of the reference shape X^r. Our reward function takes in the point cloud of the query object $\Phi_\tau(X^q)$ and the point cloud of the reference object $\Phi_a(X^r)$ captured under camera transformation imposed by a and produces a score value which reflects how well the two of the point clouds can be matched.

$$r(s, a) = f(\Phi_\tau(X^q), \Phi_a(X^r)) \tag{4.1}$$

In practice, we leveraged the ICP matching score as the feedback to compute the reward function f.

4.2.3 Learning by REINFORCE

Our reward function is non-differentiable. To solve this black-box optimization problem we opt to use the REINFORCE learning rule [173] where our goal is to find a policy $\pi_\theta(a|s)$ with parameters θ which maximizes the expected sum of rewards: $J(\theta) = \mathbb{E}_{\rho_\theta \tau}[R_t]$, where $R_t = \sum_t \gamma^{t-1} r(s_t, a_t)$. This expectation is with respect to the distribution of rollout trajectories generated by the policy

π_θ. The gradient of this objective with respect to the parameters θ can be computed by $\nabla_\theta J = \mathbb{E}_\theta[\sum_t \nabla_\theta \log \pi(s_t|a_t)(R_t - b_t)]$ where b_t is a baseline that does not depend on a_t of the future states and actions. Following a well-known approach, we choose the baseline to be $\mathbb{E}[R_t|s_t]$ and in practice we approximated it with the average value of rewards, updated over time.

To accelerate training, we augmented the loss function obtained from the REINFORCE learning rule with an auxiliary reward function that is particularly tailored for our task of shape pose estimation. This loss function reflects the error in estimating the rotation angles between the reference CAD model and the shape query scan and corresponds to sum of squared distance between the ground truth rotation and the regressed rotation. We use stochastic action sampling based on the probability produced by the current policy. Also, we use dropout [157, 44] to incorporate stochastic action selection and standard epsilon-greedy strategy in RL [160] for providing exploration in learning.

4.2.4 LICP Network Architecture

Learning complex shape representation from sparse rewards is very challenging and requires a large number of trials. Instead, we learn the shape representation using dense voxel category labels in a supervised approach, as follows. Freezing the learned shape representation network, we compute features of the 3D observation signal and use a separate network to learn the policy for finding the object poses.

3D Geometry Network: For 3D geometry feature representation, we use a 3D fully convolutional network that takes in a 3D volume as input and learns to produce per-voxel category labels in a supervised fashion, using the softmax loss function over object categories. Each tower of our geometry network uses 3D fully convolutional architecture of [155] which incorporates several 3D convolution layers.

Input volume generation: Our observation signal is in the form of 2D depth maps, which we convert to a volumetric grid of Truncated Distance Function (TDF) values. TDF representation can encode both single depth and multiple depth images. Specifically, each voxel takes a value which indicates the distance between the center of that voxel to the nearest 3D surface. Following [185],

these values are truncated, normalized and then flipped to be between 1 and 0, indicating on surface and far from surface, respectively.

Policy Network: Our policy is learned via a fully connected network consisting of three layers, each with 256 units followed by dropout and ReLU, using the policy learning and loss and reward function in Sections 4.2.2 and 4.2.3.

Training Details: We implement our model in TensorFlow [4] and use stochastic gradient descent with a learning rate of 0.001 and decay factor of 0.95. We train both 3D geometry and policy network over more than 1 million training samples in simulation.

4.2.5 Generate Training Data using Simulation

We generate synthetic training data using SUNCG scenes [155]. In each room, we move the camera at a person's height while looking at different objects in the scene. We generate a wide range of camera angles: yaw varies between $[-180, 180]$, pitch depends on the height of objects and varies between $[-90, 90]$ and roll randomly takes a value in $[-10, 10]$ degrees. To produce a variety of viewpoints, we jitter the camera with a small amount of noise. For each view, we capture the depth image and crop the box around the object which also contains some parts of the other objects. We then pass the partial point cloud to the network as input. We rasterize the mesh of the 3D CAD model into a point cloud and use the produced point cloud as the reference input of the network. The truncated distance function of the point cloud is used as input to the network.

4.2.6 Scene Recomposition

Our scene recomposition pipeline takes in a point cloud which is produced from RGBD video of a real scene. We apply 3D object detection and semantic segmentation for extracting 3D object instances. Then, we use the output of our trained 3D geometry network (see Figure 4.2) for finding the nearest 3D CAD model in the set of CAD models and use it as reference 3D shape. Finally, we deploy LICP for aligning the 3D CAD model to object instance segmentation, as described in Section 4.2.1.

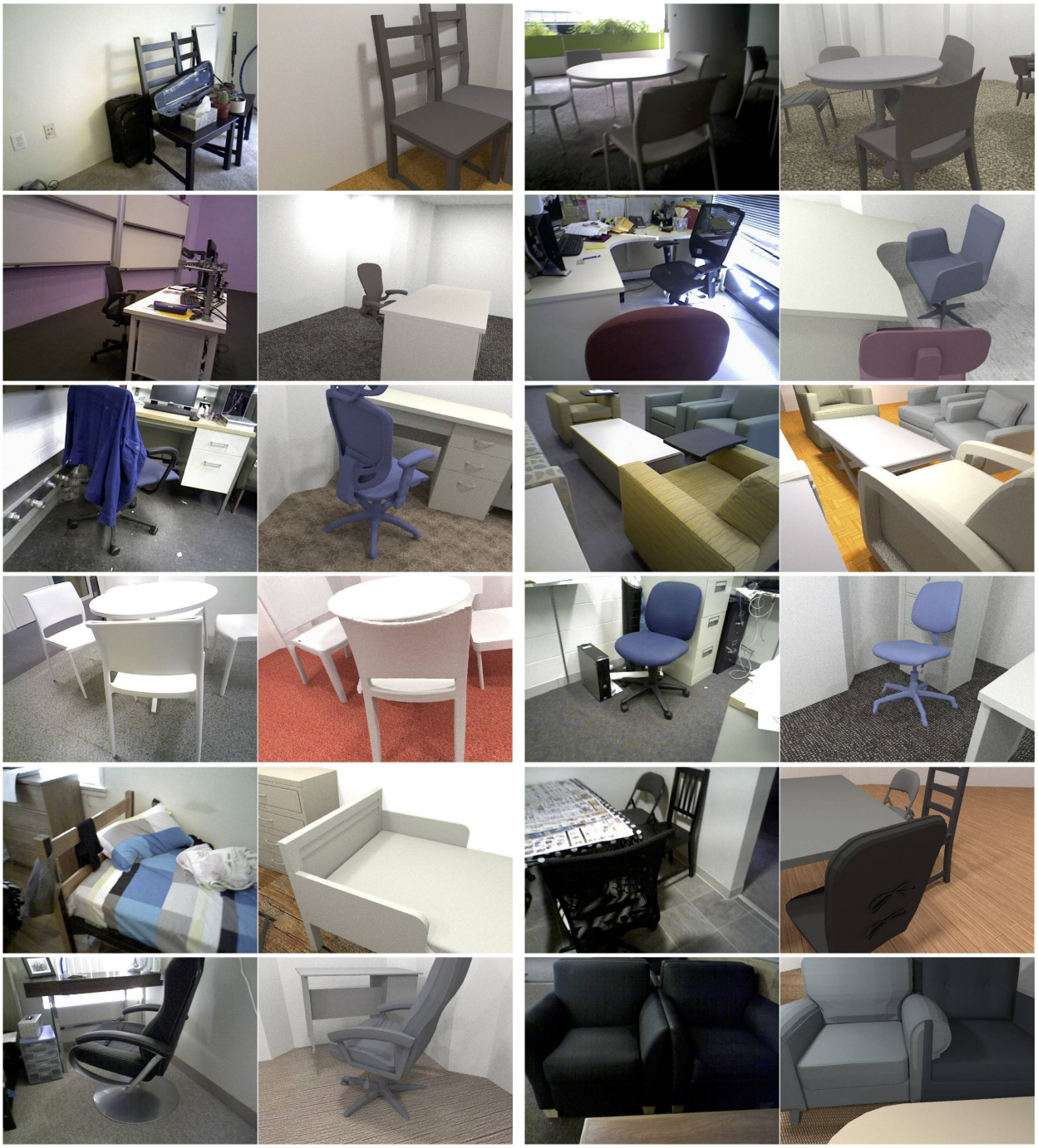


Figure 4.4: Qualitative examples of the recomposed CAD model of the scene. Each example shows a view of the camera in the scanned scene on *left* and recomposed CAD from the same view on *right*. Our method can successfully recompose cluttered scenes with lots of distractor objects (first and second rows) and huge amount of occlusions in scenes populated with many furniture objects and in confined spaces (third and fourth rows). Less accurate CAD recomposition can occur due to ambiguous extent of scanned meshes with nearby objects (bottom row, right), or lack of discriminative shape features in different views (cabinet in row four, right)

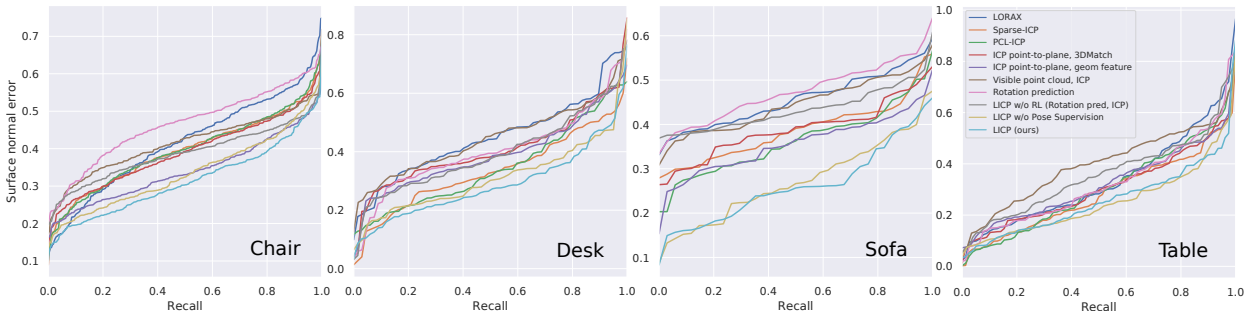


Figure 4.5: Comparison of proposed LICP method with local feature matching and alignment methods on real data (*lower values are better*). The legend is only shown on the right plot for better readability and the color of methods are the same for all plots.

3D Object Detection: We use the two-step object detection regime [127, 22, 84]. We train a category agnostic region proposal network which gives the objectness score for different 3D bounding boxes over the point cloud. We simultaneously train another network for classification of 3D bounding boxes for each of the object categories. Both networks share the feature extraction layers which are based on VGG architecture [151]. We use cross entropy loss for both region proposal and classification networks. We also learn the deviation of the 3D boxes using regression loss in x and y dimensions and the z_l and z_h for the lower and higher extent of object along Z axis in regard to the ground plane. We rectified the point cloud in world coordinates by rotating the gravity direction and then making it axis aligned with the dominant X-Y orientation on the ground plane. To compute feature maps from the point clouds we use the top-down view of the point cloud representations and extract feature from planes in different heights following [22]. For training, we use rendered depth images from SUNCG [155] as explained in 4.2.5. We use the entire scene composed of multiple objects in the field of view for each camera pose. We set 0.5 threshold for intersection over union (IoU) of 3D detection boxes and use non-maximum suppression for removing low scoring 3D boxes which have high overlap with higher scoring detections. We find the translation and scale of the objects via 3D object detector and apply the inferred translation and scale to the CAD models.

3D Semantic Segmentation: Clean object instance segmentation is important for the alignment stage of our method. For instance, when a chair is next to a table the 3D bounding box of the

chair may include some part of table and vice versa. In order to remove such distractors from the detection bounding box of each object detection we incorporate the semantic segmentation inferred on the point clouds. We take all points inside the 3D detection box and remove the points with semantic label of other object categories with overlapping detected bounding boxes. We also remove the points with "floor" and "wall" labels. We follow [123, 92, 124] for training semantic segmentation over the point cloud and learn a model for all object categories as well as floor and wall classes.

Room Layout Estimation and Scene Visualization: We use the inferred wall points from the 3D point cloud segmentation to estimate the room layout. For each point on the ground plane (X, Y), we count the number of wall 3D points, aggregating over the Z axis. The locations on the ground plane with high frequency of wall voxels define the boundary of the room. We use the extent of the floor voxels wherever scan does not have wall in the boundary. Once all wall voxels on the ground plane are computed we run the concave hull algorithm to find the room boundary. We infer the location of the floor plane to be at the Z which has the highest frequency of floor voxels inferred via semantic segmentation of 3D points. The color of each object is estimated by medoid color of the point clouds belonging to the object instance segmentation. The floor texture is selected based on the feature similarity to a set of texture image.

4.3 Experiments

In our experiments we want to investigate: 1) How accurate is our learning-based ICP compared to non-learning previous approaches, 2) how is our method compared with keypoint matching approaches based on deep features, and 3) how can our model be applied in scene CAD model recomposition of unstructured and cluttered real world environments. To answer these questions, we evaluate the performance of our method both quantitatively and qualitatively. For real-world evaluation, we use the publicly available SceneNN [64] and ScanNet [30] datasets. SceneNN and ScanNet test sets contain scans of 95 and 312 scenes from different real world indoor spaces, respectively. These scene point clouds are scanned from various offices, bedrooms, living room, kitchen, etc., and as such they exhibit a diverse collection of unstructured real world scenes popu-

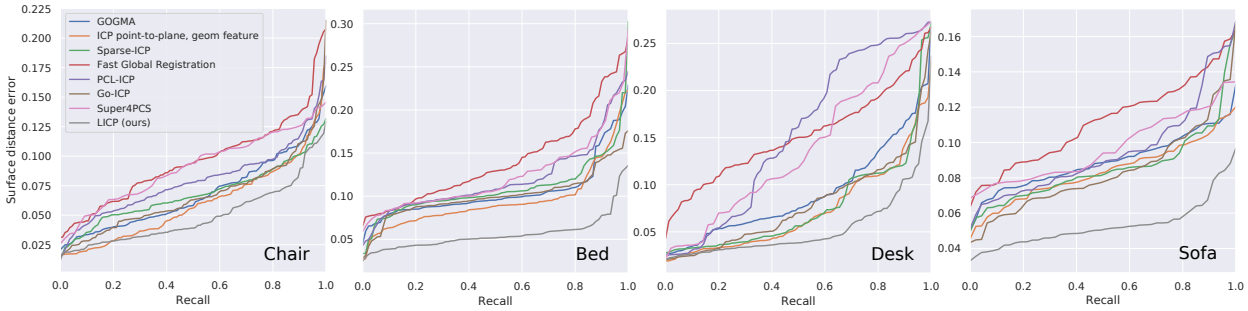


Figure 4.6: Comparison of proposed LICP method with robust and global alignment algorithms on synthetic data (lower values are better).

lated with various furniture types, styles and clutters of many distractor objects. These scenes are scanned with commodity depth camera and we use the fusion output.

4.3.1 Quantitative Evaluation

We evaluate the accuracy of our proposed method for 6DoF pose estimation of furniture objects in both real and synthetic scenarios. We compare our results with prior works of [23, 135, 185, 36, 15, 189, 181, 19, 107]. For the evaluation criteria, we compute the alignment error between the scanned mesh and the shape CAD model with the predicted pose. To compute the alignment score, the closest point in CAD model is found for each point in the input scan and the cosine distance between surface normals are computed. In the synthetic data experiment, we use the distance between points in reference CAD model and scan given that we have access to the ground truth mesh of the object in simulation.

Quantitative evaluation on real data: To evaluate the effectiveness of LICP for 6DoF object pose estimation, we incorporate the ground truth point cloud segments and object labels. We use the feature representation of our trained 3D geometry network for finding the nearest 3D CAD model from a database of 1550 CAD models from [155, 159] and use it as the reference CAD model. The quality of the object style match for retrieved CAD models are shown in several examples in Figure 4.3.

We compare LICP with local feature matching and variants of ICP from the literature. For local feature matching, we compare against hand-designed geometric feature of FPFH [135], learned

local deep feature by 3DMatch [185] and LORAX [36]. After matching the local features, we use RANSAC for coarse registration followed by point-to-plane ICP [23] for fine alignment of CAD model and input scan. For comparing against LORAX, we use released code of [36] for super-point extraction and use local deep features learned in an unsupervised fashion from point clouds of synthetic object CAD models via GAN. We also compare with *Sparse ICP* [15] (a variant of ICP that is robust to input noise), and *PCL* implementation of ICP. Figure 4.5 summarizes our quantitative comparison results. In the plots of Figure 4.5 *"ICP point-to-plane, geom feature"* refers to FPFH setting. As demonstrated in Figure 4.5, our method outperforms all aforementioned prior methods.

We also compare LICP with other baselines and variants of proposed LICP with different combinations of loss and reward function. **Rotation prediction** only uses object rotation estimation output of the learned network in Figure 4.2 and does not use our RL component. **Rotation pred., ICP point-to-plane** uses the rotation estimation output of the LICP network and applies ICP point-to-plane for finer object alignment. **Visible point cloud, ICP** only uses the visible points of the point cloud from predicted object pose for ICP alignment. **LICP w/o Pose Supervision** uses a policy network that is only trained with RL component and without strong object pose supervision of auxiliary loss. All of these variants have lower performance than our full LICP model that combines ICP-based reward and auxiliary loss for learning the policy network. Also the performance of LICP only with RL is close to LICP which suggests that LICP performance is mostly gained by RL learning rather than strong object pose supervision.

We do not have access to the ground truth CAD model of the shapes in the input scan and we use the surface normal error between recomposed CAD and input scan. We plot the surface normal error vs. recall for each category, which is the percentage of samples with surface normal error lower than each error value. Note that the smallest average ICP distance between the pair of scan and CAD model never goes to zero since the point cloud input pairs to the ICP method are sampled differently and are never identical.

Quantitative evaluation on synthetic data: We test on SUNCG [155] test set where objects are placed in 3D scenes with realistic furniture arrangements. This experiment is performed on several

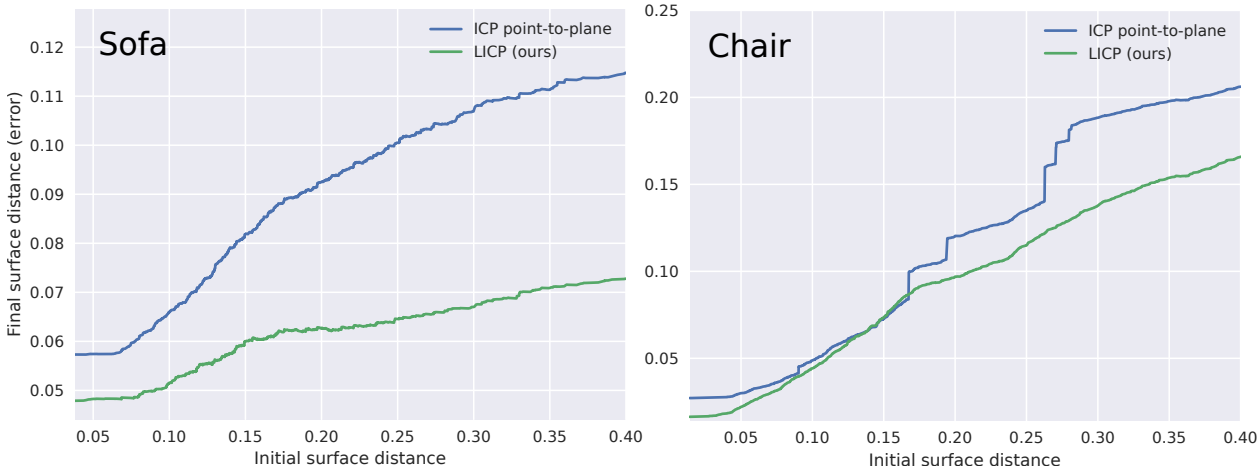


Figure 4.7: Evaluating the robustness of our proposed LICP method for aligning 3D CAD models with drastic orientation differences to the input scan using synthetic data.

input CAD models and input scans. The alignment error is the mean surface point distance in meter between the object surface in scan and the reference CAD model. In this experiment we test on synthetic scans where we have the ground truth surface of the scanned object. Therefore, we can compute the distance between the surface of the reference CAD and surface of the CAD in the scan. We compare LICP with robust and global alignment algorithms: Fast Global Registration [189], globally-optimal algorithm Go-ICP [181], GOGMA [19], Super4PCS [107] and Sparse ICP [15]. We also compare LICP against point-to-plane ICP [23] with FPFH geometric feature and PCL implementation of ICP. The results are summarized in Figure 4.6. Our LICP alignment outperforms other global and robust alignment methods by a large margin.

We also evaluate the robustness of LICP against large orientation differences between the object scan input and the reference CAD model and compare against the Chen and Medioni ICP [23] in Figure 4.7. The reference CAD models are initialized with different orientations for each experiment. In Figure 4.7, the x-axis shows the initialization error while the y-axis shows the final alignment error after ICP is converged. While both methods reduce the alignment error, LICP obtains lower final error compared to [23].

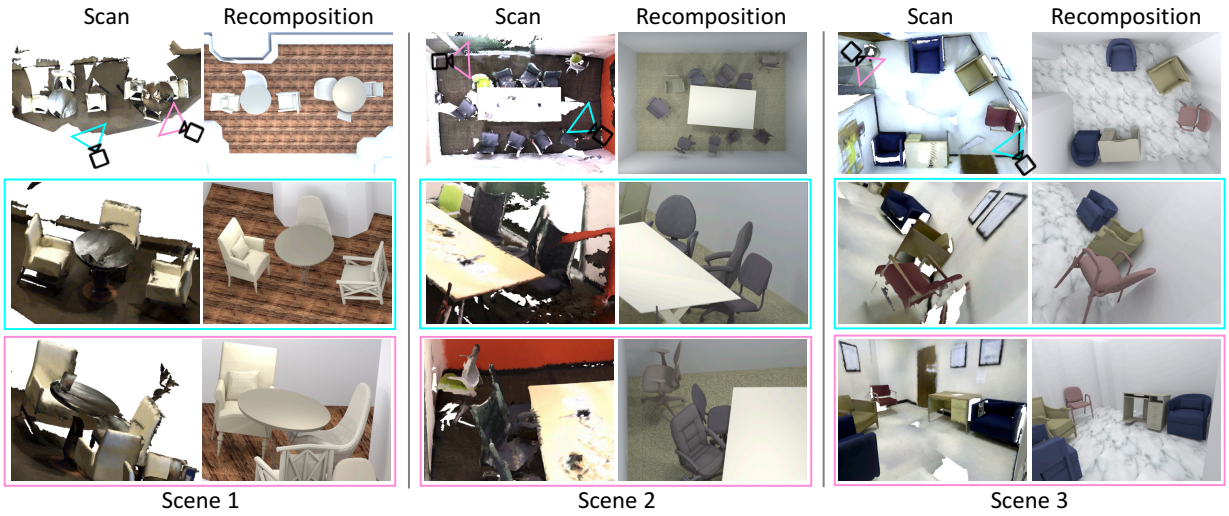


Figure 4.8: Scene *recomposition* using our proposed fully automatic method. Scene recomposition is shown for three different scenes. In each scene, the top row shows the top-down view of the scene; the middle and bottom rows demonstrate two close-up views of each scene. Camera location and pose is color coded on top-down view).

4.3.2 Qualitative Evaluation

Real scene shape alignment: Figure 4.4 demonstrates several examples of scene CAD models recomposed (on right) from the depth scan of real scenes (on left) by applying our algorithm where best-matched CAD models and 6Dof object poses are estimated. The first row in Figure 4.4 shows several recomposed CAD scene models in the presence of a high amount of scene clutter. For example, the surface of the two chairs on the top left is filled with random objects, and the back cushion of the blue office-chair (first row, middle figure) is occluded with a shirt. While such arbitrary objects results in significant amount of noise in the depth scans, our method can estimate the 6DoF pose and object style reasonably well. Examples of the second row in Figure 4.4 show scenarios with significant occlusions as the result of a densely populated scene. As shown in the figure, our method handles such occlusions well and produces CAD scene models with accurate object pose and styles. Several failure cases are shown in the bottom row of Figure 4.4 where the estimated object poses are less accurate. For example, in the middle example of the forth row, the pose of the cabinet behind the blue chair is not estimated correctly due to the lack of strong discriminative shape features between the right face and the front face of the cabinet. Also the

retrieved armchair style is not accurate in the left example of the forth row, as the extent of the armchair cannot accurately be obtained from the scanned point cloud because of high level of occlusion with the nearby objects.

Real scene recomposition: We deploy our fully automatic scene recomposition method on real scenes, with results shown in Figure 4.8. For each scene, we render two different close-up camera viewpoints and the top-down view of the scene recomposed by our method and also show corresponding views from the scan. As shown in Figure 4.8, these scenes are densely populated with different furniture and the scene scans contain many holes. Despite many occlusions and holes, our method produces satisfying scene recompositions. Using TITAN Xp GPU, the computational time for a typical scene with an average complexity is approximately 6.5 seconds for 3D object detection and 9.5 seconds for 3D semantic segmentation. LICP 3D CAD alignment takes 1.22 seconds per object instance which includes 0.65 seconds for 3D Geometry Net, 0.008 seconds for Policy Net and 0.56 seconds for ICP Reward.

Surface point visualization during inference: LICP learns to assign different weights to surface points of the reference CAD model when queried with arbitrary posed object scans. The assigned weights for surface points in the reference CAD model are computed based on the visible surface points. The visible surface points are captured via ray tracing from the actions inferred, i.e., the camera transformation multiplied with the value estimated by the value function in our policy network. These weights reflect the contribution of each surface point in inferring the correct transformation action.

Figure 4.9 shows the surface point weights obtained for different objects when queried with scans from various viewpoints. The assigned weights are conditioned on the viewpoint of the queried shape. When LICP is queried with a left-sided armchair, the visible surface points on the left side of the reference armchair gain higher weights and vice versa. Similarly, office chairs with different poses and occlusion patterns are provided. LICP assigns higher weights to the surface points that are not occluded and ignores the contribution of the occluded surface points. The bottom row of Figure 4.9 shows similar patterns in the produced weights for surface points of desk and L-shaped sofa instances.

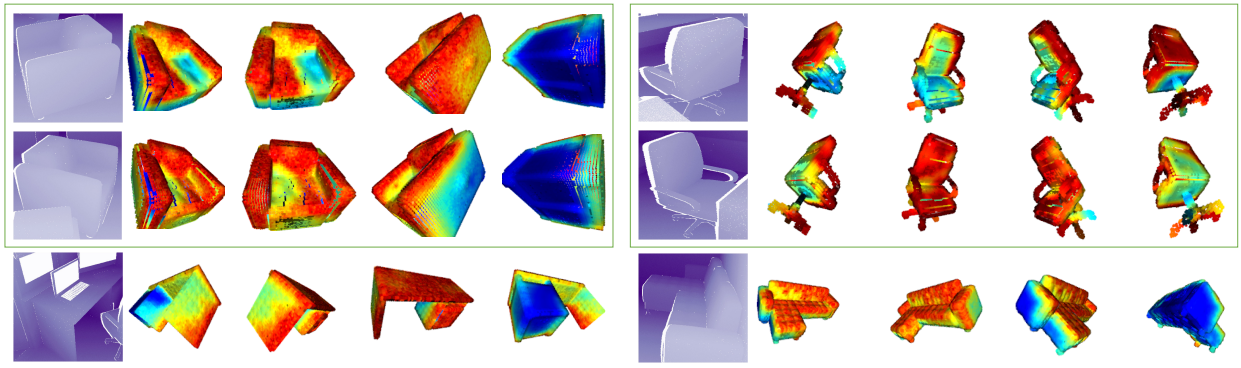


Figure 4.9: Visualization of the learned weights (*right*) for different samples and various query scan viewpoints (*left*). The learned weights are shown from four different views of the reference CAD model. Weight values are color-coded from low (*blue*) to high (*red*). The first two rows show that the surface points of the same reference CAD model are assigned with different weights depending on the queried scan viewpoint.

4.4 Discussion

In this chapter, we compute 3D scene recompositions from a sequence of RGBD scans captured by a moving camera from a real scene. We present a learning based approach for shape alignment called Learning-based ICP (LICP). LICP combines deep 3D feature learning with reinforcement learning and is able to infer the 6DoF object transformation with respect to a reference shape. By leveraging large scale shape 3D databases and learning the transformation policy for various object poses, LICP becomes robust to scene clutter and partial occlusions. Our experimental results on diverse real world scans demonstrate high performance of our method compared to various baselines.

Chapter 5

NONPREHENSILE RIEMANNIAN MOTION PREDICTIVE CONTROL

Nonprehensile object manipulation is an important and yet challenging robotic task that had been studied since more than three decades ago [105] and despite its old history, the demanding and unsolved aspects of it are still subject to research [91, 111]. Nonprehensile manipulation involves long horizon underactuated object interactions and physical contact with different objects that can inherently introduce a high degree of uncertainty. Suppose we want to push an object in a surface cluttered with objects of various shapes and properties. In such a scenario, predicting the outcome of a push action is not trivial as pushing an object not only affects its own status but also can change the status of other objects.

Traditionally, motion planning and search algorithms were adopted for nonprehensile manipulation. Planning algorithms challenge with high computational cost and rely on full knowledge of a constrained workspace to produce a sequence of kinematic actions [105]. To address the computational complexity brought by uncertainty, the quasi-static planners aim to predict the motion of pushed objects by assuming a physical model of the world involving shape, friction and objects center of mass [100].

More recently, deep learning has been adopted for learning simple robotic manipulation skills using visual sensory input in a controlled setup [89, 5, 90]. Central to their idea is to train a deep neural network model in a data-driven fashion to predict the cost of applying different robotic actions for accomplishing tasks. While it is essential to reliably predict the cost of taking possible actions in a real robotic task, it is challenging to obtain the data and the supervision needed for training such cost function in unconstrained setups.

Instead of estimating the cost of different actions, can we instead simulate the real environment in an on-line closed-loop such that we can compute the outcome of taking possible actions inside

the simulation? In this chapter, we introduce a Real-to-Sim reward analysis technique as a reliable method to imagine and evaluate the outcome of taking possible actions for a real robotic platform. Our approach creates a 3D object-level recomposed model of the real scene where we can simulate the effect of different trajectories. We produce a closed-loop controller to reactively push objects in a continuous action space. We compute the acceleration command for the robot by benefiting from Riemannian Motion Policy (RMP) and second order dynamic model at every location on the surface [24]. Our goal is to produce an efficient second order dynamic system controller that maintains a continuous trajectory while preserving correct kinematics and smooth dynamics for nonprehensile object manipulation with involves high amount of uncertainty. To accomplish this goal, we focus on leveraging simulation to predict parameters of our second order controller. Instead of estimating the accurate physical parameters of the scene such as center of mass and surface friction we apply a reactive controller that deploys closed-loop feedback. Consequently, we are capable of correcting the motion if, due to uncertainties involved, the object deviates from the foreseen trajectory.

5.1 Related Work

Conventionally, planning algorithms has been deployed for solving nonprehensile manipulation by incorporating search techniques to create a trajectory of kinematic movements for a robot arm to push or rearrange objects [27, 158, 75]. To push various objects on simple trajectories, [58] learns to predicts suitable contact points by optimizing a scoring function trained on histogram features. To reach objects in clutter, [80] deployed Model Predictive Control (MPC) [172] to model the dynamics of a robot arm in contact with the environment. Compared to these past methods, our approach is more general and more efficient and the task we consider is more challenging and involves multiple object interactions.

More recently, deep learning has been incorporated for learning simple manipulation skills with a robot arm. [5] predict the outcome of a pushing action by training a network on pairs of RGB images of before and after push action. This work is extended in [111] to learn a forward models for manipulating a rope with a robot arm in a supervised fashion. Using a series of observations

via camera images, [91] tracks a history of push interactions and learns a model for pushing in scenarios with only a single object on the table and quantizing the action space. In contrast, our method is designed to be applicable in complex scenarios involving several objects and our controller produces continues actions.

Our work is related to recent efforts for deploying 3D and physics simulation environments for efficiently learning robotic policy. Several past works deployed domain adaptation to transfer policies learned in simulation to the real world [134, 16]. Domain randomization was proposed in [139] to train highly generalizable robotic policies inside a randomized simulation environment and was shown to be successfully applicable in various robotic applications [70, 167, 7]. In [16] pixel-level image generation was incorporated to create auxiliary source of data. [71] learned to translate real and simulation data into a canonical representation to be used as robot observation. Several past works incorporated real images as a complementary source of data for learning control policies inside simulation that can be transferred to the real world [16, 141, 138]. In contrast to all these works, we generate the simulation environment by recomposing the 3D scene and incorporating simulation physics environment as a computational model for predicting the outcome of robot action trajectories in a closed loop. To our knowledge, closed loop 3D scene recomposition and simulated action look ahead has not been addressed before for efficient policy evaluation inside simulation that is applicable in the real world.

5.2 Proposed Method

We assume a nonprehensile robotic manipulation setup with N objects $o_{i,i\in\{1,...,N\}}$. Each object has 5 degree-of-freedom (5-DoF) defined by its rotation $R \in \mathbb{SO}(3)$ and translation $T \in \mathbb{R}^2$ and can take any arbitrary pose p in $\mathbb{SE}(2)$ on the plane. All the objects are movable on the plane and can be displaced from a location and pose to another physically feasible pose on the plane so at any time t, the state of the world s_t encapsulates the pose of the objects in the scene $s_t = (p_1, ..., p_N)$. Given a target object o_c and a goal location $l_g = (x_g, y_g)$ as inputs, we want to control the target object o_c by pushing it to l_g. To accomplish that, we need to find the best sequence τ of robot actions $\tau = \{u_0, u_1, ..., u_t\}$ that pushes o_c to l_g while satisfying the constraints such as avoiding collision

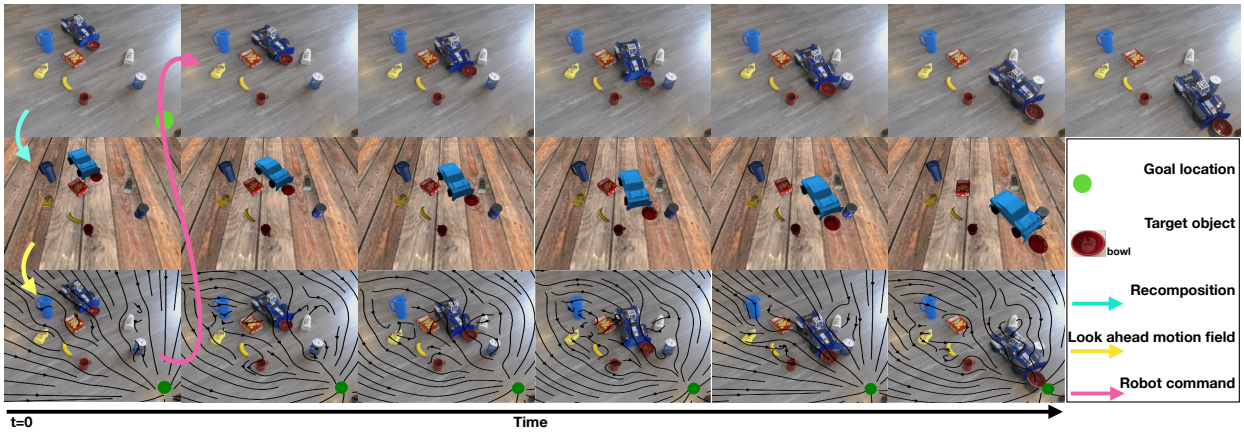


Figure 5.1: Overview of our approach. **Top row** shows the rgb snapshots of the point cloud stream input to our model which we use to recompose the scene into a simulated 3D scene (**middle row**) and generate look ahead motion field (**bottom row**).

with other objects as well as maximizing a reward function. Each robot action u is defined by the velocity and acceleration of a push by a robot end-effector, i.e. $u_t = (v_x, v_y, a_x, a_y)$. At every iteration, our goal is to estimate a short trajectory of future actions τ that maximizes the expected sum of future rewards. By applying each action u_t at state s_t, a reward $r_t(s_t, u_t)$ will be obtained which reflects how much the object has progressed in becoming closer to the goal location. If the action u_t results in a collision between the controllable object o_c and any of the objects in the scene $o_{k,k\in\{1,\dots,N\},k\neq c}$ then a penalty would be considered. More formally, the reward at each step r_t is defined as:

$$r_t(s_t, u_t) = \Delta(d^t_{o_c,l_g}, d^{t-1}_{o_c,l_g}) - \sum_{i=1,i\neq p}^{N} d(p^t_i, p^{t-1}_i) \tag{5.1}$$

The first term in Equation 5.1 reflects how much the distance of o_c to the goal location d_{o_c,l_g} has changed between time $t-1$ and t and d denotes the Euclidean distance. The second term of Equation 5.1 quantifies the collision event that might occur by applying action u_t. During one step of push action, if the controllable object o_c collides with any other object o_i, it will result in an unwanted change of 5-DoF pose in o_i. To quantify the collision event, we compare the pose of each object p_i between the time t and $t-1$ by computing the Euclidean distance d between p^t_i and p^{t-1}_i. Accordingly, the reward of a trajectory τ with fixed horizon H is computed as: $R_\tau = \sum_{j=t}^{t+H} \gamma^{j-t} r_t$

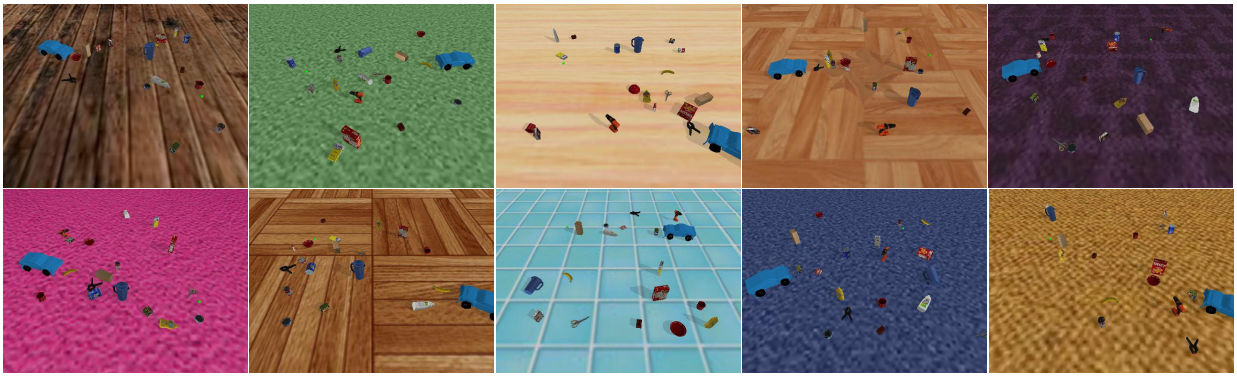


Figure 5.2: Sample pushing tasks for RC-car.

with the decay factor of γ.

5.2.1 Closed-Loop Control

We use RMPflow [24] to compute the sequence of robot actions based on the current robot position and velocity and the configuration of scene objects. The RMPflow combines local policies in their coordinate system and computes a global policy for the robot motion. We use attractor policy for pushing to the goal and collision avoidance policy for avoiding obstacles. By combining these local motion policies the potential field of robot action is computed in continuous space which determines the robot global motion policy at every location. Suppose the average dimension o_c along its width and length is $\bar{m}_{o_c}$ and its location on the plane is $l_{o_c} = (x_{o_c}, y_{o_c})$. The local policy in the coordinate of the target object rotated by the orientation of the RMP global policy for an immediate pushing action is defined by $(v_x, v_y) = (\alpha_1 x^2 - \alpha_2 y^2 - \alpha_3 (\frac{\bar{m}_{o_c}}{2})^2, \alpha_4 xy)$ and α_is are the hyperparameters of our local potential field.

5.2.2 Riemannian Motion Predictive Control

Using the computed RMPflow [24] we can select an action u_t at each time step t to find the optimum trajectory to push o_c to l_g. However, in dynamic systems with a high degree of uncertainty selecting the actions purely based on RMPflow may not be optimal specially in underactuated tasks [101] such as ours where the target object makes and breaks contact the with robot end-

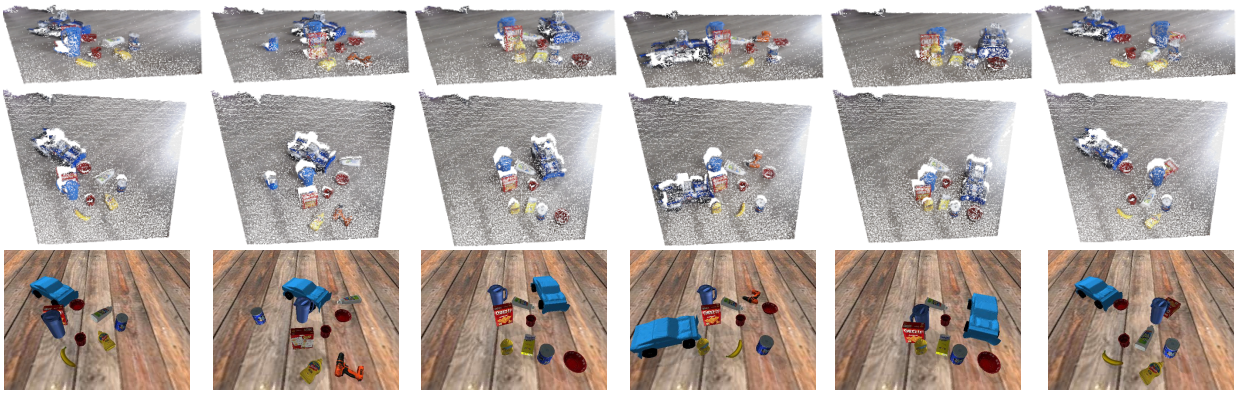


Figure 5.3: Examples of fully automatic object-level scene recomposition from the point cloud. The **first** and **second** rows show the side view and the top view of the captured point cloud. The **third** row shows our object-level scene recomposition.

effector and other objects . Therefore, we propose Riemannian Motion Predictive Control (RMPC) where instead of fixing a single parameter for computing the RMPflow, we consider a range of weights for each node. By sampling from the different node weights, we produce different pushing trajectories at every time step t and select the next best action based on the trajectory with the highest estimated reward using Equation 5.1.

Designing an accurate reward function and learning to predict the outcome of different action sequences is challenging due to inefficiencies in collecting real robot data and inherent uncertainties in under-actuated and non-prehensile object manipulation. To alleviate those challenges, our proposed method estimates the reward of different sampled action trajectories by recomposing the 3D scene and imagining the outcome of different action trajectories inside a 3D physics simulation. We leverage the power of 3D physics simulation to look ahead the outcome of pushing an object along different trajectories by proposing to recompose the 3D scene at each step during the trajectory. In our setup, we have a depth camera that captures the scene and we incorporate sensory point cloud input for object-level scene recomposition. Using the captured point cloud, we detect 3D objects in the scene. Once we recomposed the scene using object-level 3D shapes, we migrate the scene to a physics simulator to evaluate the outcome of different pushing trajectories. To train a 3D object detector that is robust to clutter and high amount of occlusion, we generate a high volume of diverse synthetic data in a simulation environment inspired from prior work of [68] and domain

randomization [139]. We create a huge number of diverse scenes by randomly dropping objects in a physic simulation. Such data generation approach will capture the realistic physical arrangement of objects on a surface and results in learning a more efficient 3D object detector. Figure 5.1 illustrates the overview of our approach where scenes captured by point clouds are recomposed in a 3D simulation environment where we can compute the look ahead motion field.

5.2.3 Implementation Details

We implemented our method using PyTorch [120] and used Bullet physics engine for simulating the scenes [28]. We generate over 1 million simulated scenes diversified with random object arrangements and floor texturing to train our 3D object detector and Figure 5.5-(c) provides several examples of the generated scenes. The speed of our recomposition step is 15 frames per second. We set the hyperparameter of our local field to α_is$= 1$ in all our experiments.

5.3 Experiments

We conduct experiments both in simulation and in a real world to evaluate the performance of our proposed approach for nonprehensile manipulation. To evaluate the performance of our proposed approach for nonprehensile manipulation we conduct experiments both in simulation and in a real robot platform.

5.3.1 Experimental Setup

In our evaluations, We use objects of the YCB dataset [18] in simulation and in the real world experiments. For real robot experiments, we build a real RC-car hardware platform on a 1/10 chassis featuring a 4×4 suspension, and non-flat tires inspiring from the prototypes introduced in [3, 156]. To control the RC-car we use move commands that define the velocity and steering direction of the RC-car through a ROS interface. In the real experiments, we localize the RC-car robot using the April tag [116] while we detect and localize other scene objects by 3D object detection as explained in Section 5.2.

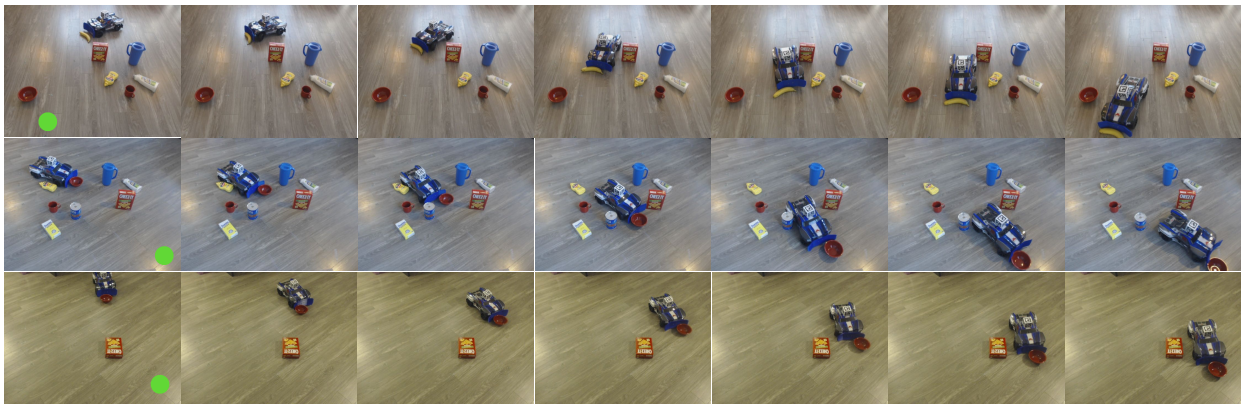


Figure 5.4: Real world RC-car robot action trajectory generated by proposed RMPC. Each row shows a trial with a different initial location of the target object, other objects as obstacles. The goal location is indicated with a green dot.

5.3.2 Results on Simulation

To evaluate our proposed motion policy we generate a held-out test set of scenes with a different combination of objects and various arrangements in simulation. To produce diverse scenarios, we randomly place objects in random locations of the workspace surface area and randomly select the target object and goal location in simulation. Figure 5.2 shows several examples of push tasks at test time. We use 500 held-out test scenes on our simulated RC-car robot platform and compare performance based on the collision rate. We compute the recall rate at the $k\%$-collision ratio as evaluation criteria. For each trial, the collision ratio is computed by dividing the number of collision events by the total length of the trajectory. A higher recall rate at a lower collision ratio indicates better performance.

We compare the performance of our RMPC method against several baselines:

Riemannian Motion Predictive Control (RMPC): Our proposed RMPC recomposes scene from input point cloud and computes different trajectories using RMPflow with different weights. The next best action is selected based on future reward obtained from Equation5.1

Riemannian Motion Policy (RMP): RMP generates the action trajectory based on the computed RMPflow with a fixed parameter similar to [24].

Model Predictive Control (MPC): MPC samples different trajectories at each step and selects

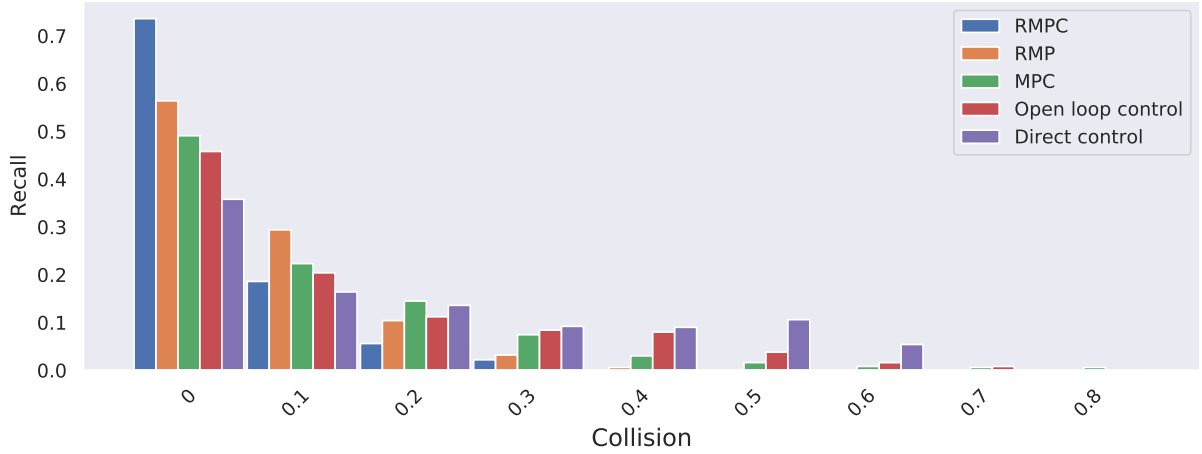


Figure 5.5: Results of collision rate on held out test set using proposed Riemannian Motion Predictive Control (RMPC) on RC-car robot platform for pushing target object to goal location compare to prior works. (higher recall in lower collision rate is better).

the next action from the trajectory with the smallest cost in a closed-loop. The cost function is computed based on the distance of the target object and the location of the obstacles so the trajectory that moves closer to the obstacles is of higher cost.

Open-loop control: This baseline is similar to MPC except that it computes the whole trajectory from the initial location of the target object to the goal location and applies action sequence in an open-loop.

Direct control: This baseline moves the target object toward goal location without considering collision avoidance and recomputes the action in closed-loop.

Comparison results are summarized in Figures 5.5 and 5.6 and show that our proposed RMPC method outperforms all of the baselines and experiences less number of collisions during the trails. Hence RMPC obtains higher recall at lower collision ratios.

5.3.3 Results on Real-World

In all real experiments we capture the scene point cloud using a depth camera. Given the point cloud of a scene, our 3D object detection produces the 3D bounding boxes around each object with their 3D poses as well as category labels from which we can recompose the scene in the physics

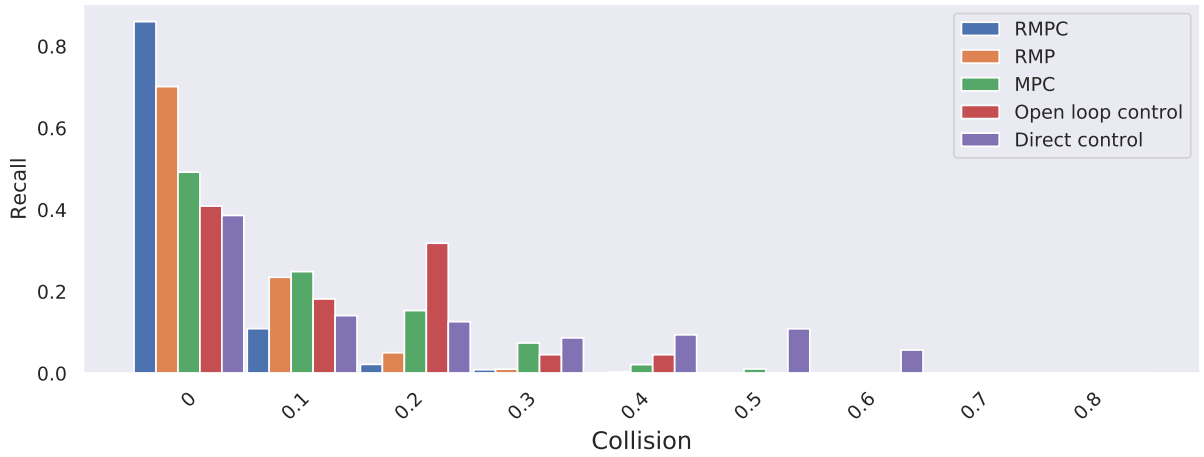


Figure 5.6: Collision rate of proposed RMPC controller on RC-car robot platform for successful trajectories where target object is pushed close to the goal location on held out test set compared to prior works. (higher recall in lower collision rate is better).

simulation. Figure 5.3 depicts several examples of the recomposed real scene in the 3D simulator. As can be seen, our scene recomposition produces simulated scenes that are identical to the real world in terms of object scene arrangement, various object categories, and their poses. Also, our scene recomposition is robust to a high amount of clutter and occlusion in the scene.

We quantitatively compare our proposed RMPC with RMP which we observed to be the strongest baseline in section 5.3.2. We evaluate the performance by computing the percentage of non-collision push actions in different trials. We evaluate the performance in 32 different real scene arrangements and obtained the average collision rate of 17.9% for RMPC and 35.4% average collision for RMP. Figure 5.4 shows the example trajectories of real RC-car pushing tasks in three different scenarios. As Figure 5.4 shows, our proposed RMPC method produces a closed-loop sequence to successful actions and pushes the controllable object to the goal location(shown by green dot) without colliding to with obstacles in the scene where pushing to a goal location could be accomplished without a collision event.

5.4 Discussion

Solving for the parameters of a second order dynamic system controller is challenging when the task involves objects of various properties and complex manipulation under high uncertainty specially in nonprehensile settings. In this chapter, I propose a Real-to-Sim approach for analysing the reward of action trajecotry in simulation and estiamting the parameters of a second order dynamic system controller for nonprehensile manipulation task. With our controller, we produce a closed-loop reactive motion policy. I consider the complex task of pushing objects on a surface in a complex environment with high uncertainty and demonstrate the efficiency of our controller by comparing it to the prior works on a set of nonprehensile manipulation tasks in simulation and real-world RC-car robot platform.

Chapter 6
CONCLUSION

In this thesis, we proposed learning-based object-level scene recomposition from different types of input sensory data. Our proposed scene recomposition approaches re-create a scene through learning to detect and align prominent objects in the scene along with their category, shape, 3D pose, and room layout.

6.1 Contributions

Recomposition of the scene CAD model is the re-creation of the scene by localizing the objects and inferring their pose and object type and shape to match the scene captured via a input sensory data. This thesis made contributions by proposing learning-based approaches for scene recomposition an application to robotic object manipulation as follows:

- **Object-level scene analysis.** We proposed learning object-based scene structure from RGB image, including scene category, the appearance of objects in scene context, and the spatial relation of prominent objects in the scene. We showed that the relationships between prominent objects in the scene is an important clue for scene understanding, which can be used for improving object detection, fine-grained scene categorization, and inferring missing objects in an image [67].

- **Single-view scene CAD recomposition.** We proposed IM2CAD, a fully automatic scene CAD recomposition approach, from a single RGB image of an indoor scene, that detects prominent objects of the scene, finds CAD models that are as similar as possible to those objects,and places them automatically in the 3D scene. Then we trained a fully convolutional neural network for estimating the room geometry by detecting walls, floor, and ceiling. We

leveraged a large database of thousands of object CAD models from ShapeNet [20] to match the style and pose of each object detected in the image. To do that, we compared the convolution filter response of our trained object detector with the object shape database. To account for inter-object occlusions and to produce a more accurate estimation of object location and pose in the scene, we then rendered and matched for jointly aligning all objects in the scene. We showed that our method produces high quality scene recomposition from single RGB images. Finally, we compared our method against prior works, evaluating on publicly available 3D scene understanding benchmarks and observed that our method obtains superior quantitative performance [69].

- **Scene CAD recomposition from RGBD sequences.** We proposed a fully automatic scene recomposition from RGBD sequences by matching the scanned object to a large scale database of object CAD models and aligning them to the input RGBD scan. Our scene recomposition approach involves 3D object detection and semantic segmentation for extracting 3D object instances from the input scan. The best-matched 3D CAD model to the extracted object instance is found from thousands of object CAD models using our trained 3D geometry network. The scene recomposition yields more complete models, including back-facing and hidden geometry of objects, as opposed to reconstruction methods that capture only visible surfaces of objects and generate holes in the mesh. We also proposed a novel learning-based ICP (LICP) technique in a reinforcement learning framework to align CAD models to RGBD scanned geometry. LICP learns to predict the 3D object pose of the queried object to the input scan. While LICP is trained on synthetic data and without needing 3D annotation of keypoint correspondences in real scenes, it outperformed prior works on learned local feature matching and geometric based alignment methods [68].

- **Scene CAD Recomposition for robotic nonprehensile manipulation.** We proposed scene CAD recomposition for robotic nonprehensile object manipulation. We studied the task of pushing a target object to a goal location on a surface cluttered with obstacle objects of various shapes and properties on a real RC-car robot platform. Given the input RGBD

point cloud of the scene, including objects on the robot's workspace and the robot itself, we generate the object-level scene recomposition of the scene. We then used the recomposed scene to compute the expected sum of future rewards for a real-world robot action sequence in the recomposed CAD model in a physics simulation. We produce a closed-loop controller where at each timestamp robot perceives the work space, recomposes the 3D scene, and uses that to look ahead the reward of different actions in the recomposed 3D scene. Then, the action with maximum reward is selected and applied by the real robot in the scene. The performance of our proposed method is evaluated on simulated and real-world RC-car robot platform. We showed that our method obtains better performance compared to several baselines [66].

6.2 Future Work

We introduced the scene recomposition problem and proposed multiple solutions to tackle it. We believe that our work opens up new research directions as follows. Scene recomposition can be improved by incorporating more details of the scene, such as adopting illumination inference and light source detection. Deformable objects, such as a person in the scene, can also be considered. Furthermore, object type and shape can be more adapted and customized based on scene context. Scene recomposition can also be useful in various applications such as interactive games, virtual content creation, VR/AR, interactive style, matching and interior design. Now, we elaborate on the aforementioned research directions for future work.

6.2.1 Object Context in Recomposition

The object choice in our scene recomposition method is based on the perceived shape from the sensory data such as RGB or depth. While we seek to to recompose a scene to be as close as possible to the real scene, our object choice can be improved in scenarios where there is a high level of noise in the sensory data. For example, partial views, object occlusions, or noisy sensing can result in incomplete sensory data [190]. One improvement is to use scene context to enhance

the shape and style choice. Suppose we detect a chair in the scene, which is partially observable because of other furniture pieces in front. Retrieving the style and shape of such an object is difficult based on the perceived sensory data. However, if we consider the scene context and the nearby objects, we can infer the scene type (e.g., office) and bias our choice inclined towards the chair styles that occur in such scenes more often (e.g., office chair). Another example use case where incorporating context can improve the recomposition is in the environments where there are multiple instances of an object such as a classroom with many similar chairs, a lobby with multiple similar sofas, or a bedroom with similar side tables. In such cases, if we can infer the style and shape of one instance of the repeated object with high confidence, then we can propagate the inferred style to other instances.

6.2.2 Person Recomposition

Deformable objects and specifically human pose estimation from RGB and RGBD images is an active and important research area in computer vision and graphics [77, 78]. This includes inferring a person's pose from sensory input by learning the shape of different body parts as well as the person's clothing type and colors. Having a database of different part shapes and clothing types, we can *learn to re-create* the human in the scene. Such re-creation can involve inference over all possible combinations of body attributes such as pose and clothing, which makes it a very challenging and interesting future research direction.

6.2.3 Light and Illumination Estimation

Scene illumination is a crucial aspect of realistic synthetic scene rendering. Future recomposition work can consider learning to estimate the light sources for indoor scenes such as approaches proposed in [186, 46]. The various sources of light and their intensities in an indoor scene makes this problem challenging. For example, different light sources in an indoor scene can include outside natural light reflected through a window, and indoor light sources such as the ceiling or wall lamps as well as floor or table lamps; the effect of all such lights sources in the final appearance of

the scene can be different based on object materials, object occlusions and day versus night.

6.2.4 Textured Scene Recomposition

Inferring material and texture on different object surfaces can play a critical role in producing more realistic scene recompositions. However, it is an interesting research direction to investigate ways for re-creating a scene with its current texture or by generating a scene with the same object layout but new texturing that looks realistic on the surface of objects similar to the approach of [171, 119].

6.2.5 Interactive Gaming and VR/AR Use Cases

Most gaming and entertainment applications are designed with a set of prebuilt scenes and environments. Such prebuilt environments can result in limited user engagement and less convenient interactions with the scene. Having an accurate object-level scene recomposition could in many cases provide a better user experience. This can be addressed by providing situational awareness of the working area for the user. As an example, while working with a VR/AR application and interacting with virtual and real-world objects, the user should be able to identify occupied areas so that he/she can avoid bumping into the room objects. For this to be realized, we need to have an accurate scene recomposition to generate the virtual world presented for the user such that the extent of the objects is as close as possible to the CAD recomposition. As another example, a VR/AR application might require the user to sit down and stand up during VR/AR experience. User should be able to work with such an application without any interruption in between. For example, if we could model objects in a VR user's real environment, we could add them to their VR world, enabling the user to sit down on a real chair or touch a real object. Scene CAD recomposition can enable gaming and VR/AR applications with situational awareness by importing the CAD model of the dominant objects that are aligned with the real scene surrounding the user and rendering in a virtual environment.

6.2.6 Interactive Interior Design and Modeling

Object-level scene recomposition enables object oriented 3D scene editing. By copying and pasting new objects in the scene or replacing current scene objects with other objects of *different* styles, we can provide an interactive tool for interior design and room modeling. Our 3D scene recomposition technique enables an interactive virtual 3D environment where both style and size can conveniently be taken into account to provide a close-to-real experience for the user. Such an application can be deployed in a Virtual Reality setup where a user can edit and design their room while being *virtually present* in that environment.

BIBLIOGRAPHY

[1] Eye of the robot. https://people.csail.mit.edu/bkph/phw_copy_demo.shtml, 1970.

[2] Lsun room layout estimation dataset. http://lsun.cs.princeton.edu/, 2015.

[3] The mit racecar. https://mit-racecar.github.io, 2016.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[5] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NeurIPS*, 2016.

[6] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

[7] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.

[8] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of cad models. In *CVPR*, 2014.

[9] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Niessner. Scan2cad: Learning cad model alignment in rgb-d scans. In *CVPR*, 2019.

[10] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, 2016.

[11] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 2001.

[12] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*. International Society for Optics and Photonics, 1992.

[13] Blender. Blender cycles render engine. https://www.blender.org/manual/en/render/cycles/index.html.

[14] M. Blum, A. Griffith, and B. Neumann. A stability test for configurations of blocks. 1970.

[15] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. Eurographics Association, 2013.

[16] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018.

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

[18] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv:1502.03143*, 2015.

[19] D. Campbell and L. Petersson. Gogma: Globally-optimal gaussian mixture alignment. In *CVPR*, 2016.

[20] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], 2015.

[21] A. X. Chang, M. Savva, and C. D. Manning. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*, 2014.

[22] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[23] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[24] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff. Rmpflow: A computational graph for automatic motion policy generation. In *International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[25] M. J. Choi, A. Torralba, and A. S. Willsky. A tree-based context model for object recognition. *IEEE Trans. PAMI*, 2012.

[26] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Indoor scene understanding with geometric and semantic contexts. *IJCV*, 2015.

[27] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman. Push planning for object placement on cluttered table surfaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[28] E. Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*. 2015.

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

[30] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[31] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018.

[32] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[33] S. Dasgupta, K. Fang, K. Chen, and S. Savarese. Delay: Robust spatial layout estimation for cluttered indoor scenes. In *CVPR*, 2016.

[34] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NeurIPS*. 2013.

[35] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[36] G. Elbaz, T. Avraham, and A. Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *CVPR*, 2017.

[37] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 2008.

[38] O. Faugeras and O. A. FAUGERAS. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[39] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 2010.

[40] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[41] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based synthesis of 3d object arrangements. *TOG*, 2012.

[42] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[43] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, 2004.

[44] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[45] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

[46] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagne, and J.-F. Lalonde. Deep parametric indoor lighting estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[47] A. Geiger and C. Wang. Joint 3d object and layout inference from a single rgb-d image. In *German Conference on Pattern Recognition*. Springer, 2015.

[48] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008.

[49] J. J. Gibson. The perception of the visual world. 1950.

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[51] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 2013.

[52] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*, 2017.

[53] R. Guo, C. Zou, and D. Hoiem. Predicting complete 3d models of indoor scenes. *arXiv preprint arXiv:1504.02437*, 2015.

[54] A. Gupta, M. Hebert, T. Kanade, and D. M. Blei. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NeurIPS*, 2010.

[55] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *CVPR*, 2015.

[56] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[57] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[58] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick. Learning contact locations for pushing and orienting unknown objects. In *IEEE-RAS international conference on humanoid robots (humanoids)*, 2013.

[59] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 2012.

[60] D. Hoiem. *Seeing the world behind the image: Spatial layout for 3D scene understanding*. PhD thesis, Carnegie Mellon University, 2007.

[61] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005.

[62] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007.

[63] B. Horn, B. Klaus, and P. Horn. *Robot vision*. MIT press, 1986.

[64] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, 2016.

[65] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. In *SIGGRAPH*, 2015.

[66] H. Izadinia, B. Boots, and S. M. Seitz. Nonprehensile riemannian motion predictive control. 2020.

[67] H. Izadinia, F. Sadeghi, and A. Farhadi. Incorporating scene context and object layout into appearance modeling. In *CVPR*, 2014.

[68] H. Izadinia and S. M. Seitz. Scene recomposition by learning-based ICP. In *CVPR*, 2020.

[69] H. Izadinia, Q. Shan, and S. M. Seitz. IM2CAD. In *CVPR*, 2017.

[70] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv:1707.02267*, 2017.

[71] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, 2019.

[72] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014.

[73] H. Jiang and J. Xiao. A linear approach to matching cuboids in rgbd images. In *CVPR*, 2013.

[74] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *TPAMI*, 1999.

[75] A. M. Johnson, J. E. King, and S. Srinivasa. Convergent planning. *IEEE Robotics and Automation Letters*, 2016.

[76] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[77] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)*, 2018.

[78] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[79] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh. 3D object manipulation in a single photograph using stock 3d models. In *SIGGRAPH*, 2014.

[80] M. D. Killpack, A. Kapusta, and C. C. Kemp. Model predictive control for fast reaching in clutter. *Autonomous Robots*, 2016.

[81] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3d indoor environments with variability and repetition. *TOG*, 2012.

[82] J. J. Koenderink and A. J. Van Doorn. Affine structure from motion. *JOSA A*, 1991.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*. 2012.

[84] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. *IROS*, 2018.

[85] L. Ladický, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.

[86] K. Lai and D. Fox. Object recognition in 3d point clouds using web data and domain adaptation. *The International Journal of Robotics Research*, 2010.

[87] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[88] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009.

[89] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 2016.

[90] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 2018.

[91] J. K. Li, W. S. Lee, and D. Hsu. Push-net: Deep planar pushing for objects with unknown physical properties. In *Robotics: Science and Systems*, 2018.

[92] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn: Convolution on $\mathcal{X}$-transformed points. In *NeurIPS*, 2018.

[93] E. P. X. Li-Jia Li, Hao Su and L. Fei-Fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NeurIPS*, 2010.

[94] Y. L. Li-Jia Li, Hao Su and L. Fei-Fei. Objects as attributes for scene classification. In *ECCV*, 2010.

[95] J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *ECCV*, 2014.

[96] C. Liu, A. G. Schwing, K. Kundu, R. Urtasun, and S. Fidler. Rent3d: Floor-plan priors for monocular layout estimation. In *CVPR*, 2015.

[97] Z. Liu, Y. Zhang, W. Wu, K. Liu, and Z. Sun. Model-driven indoor scenes modeling from a single image. In *Proceedings of the 41st Graphics Interface Conference*, 2015.

[98] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[99] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[100] K. M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IROS*, 1992.

[101] K. M. Lynch and M. T. Mason. Dynamic underactuated nonprehensile manipulation. In *IROS*, 1996.

[102] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NeurIPS*, 2009.

[103] A. Mallya and S. Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, 2015.

[104] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[105] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 1981.

[106] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*. Wiley Online Library, 2014.

[107] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs: fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014.

[108] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. Interactive furniture layout using interior design guidelines. In *SIGGRAPH*, 2011.

[109] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[110] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[111] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA*, 2017.

[112] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *TOG*, 2012.

[113] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping

and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136. IEEE, 2011.

[114] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. PAMI*, 2002.

[115] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[116] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *ICRA*, 2011.

[117] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.

[118] S. A. Papert. The summer vision project. 1966.

[119] K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. Photoshape: photorealistic materials for large-scale shape collections. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 2018.

[120] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[121] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *CVPR*, 2012.

[122] M. J. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*. 1994.

[123] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[124] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[125] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[126] S. Ramalingam, J. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *CVPR*, 2013.

[127] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[128] Y. Ren, C. Chen, S. Li, and C.-C. J. Kuo. A coarse-to-fine indoor layout estimation (cfile) method. *arXiv preprint arXiv:1607.00598*, 2016.

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

[130] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *CVPR*, 2015.

[131] S. Rusinkiewicz. A symmetric objective function for ICP. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[132] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[133] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[134] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *CoRL*, 2017.

[135] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA*. IEEE, 2009.

[136] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IROS*. IEEE, 2008.

[137] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *ICRA*, 2011.

[138] F. Sadeghi. DIViS: Domain invariant visual servoing for collision-free goal reaching. In *RSS*, 2019.

[139] F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *RSS*, 2017.

[140] F. Sadeghi and M. F. Tappen. Latent pyramidal regions for recognizing scenes. In *ECCV*, 2012.

[141] F. Sadeghi, A. Toshev, E. Jang, and S. Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *CVPR*, 2018.

[142] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013.

[143] S. Satkin, M. Rashid, J. Lin, and M. Hebert. 3dnn: 3d nearest neighbor. *IJCV*, 2015.

[144] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NeurIPS*, 2006.

[145] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *ICCV*, 2013.

[146] A. G. Schwing and R. Urtasun. Efficient exact inference for 3d indoor scene understanding. In *ECCV*. 2012.

[147] S. M. Seitz. *Image-based transformation of viewpoint and scene appearance*. PhD thesis, University of Wisconsin-Madison Department of Computer Sciences, 1997.

[148] J. Seyama and R. S. Nagayama. The uncanny valley: Effect of realism on the impression of artificial human faces. *Presence*, 16(4):337–351, 2007.

[149] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007.

[150] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

[151] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[152] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. 2012.

[153] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015.

[154] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014.

[155] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[156] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. S. M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi. Mushr: A low-cost, open-source robotic racecar for education and research. *arXiv preprint arXiv:1908.08031*, 2019.

[157] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[158] M. Stilman and J. J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2005.

[159] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018.

[160] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[161] B. Taskar. *Learning structured prediction models: a large margin approach*. PhD thesis, Stanford University, 2004.

[162] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. *NeurIPS*, 2004.

[163] D. Thanh Nguyen, B.-S. Hua, K. Tran, Q.-H. Pham, and S.-K. Yeung. A field model for repairing 3d shapes. In *CVPR*, 2016.

[164] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*, 2013.

[165] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010.

[166] A. Torralba. The context challenge `http://web.mit.edu/torralba/www/carsAndFacesInContext.html`.

[167] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPRW*, 2018.

[168] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*. Springer, 1999.

[169] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015.

[170] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[171] T. Y. Wang, H. Su, Q. Huang, J. Huang, L. Guibas, and N. J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–13, 2016.

[172] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, 2009.

[173] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 1992.

[174] P. Winston. *Learning structural descriptions from examples*. PhD thesis, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1970.

[175] J. Wu and J. Rehg. Centrist: A visual descriptor for scene categorization. *IEEE Trans. PAMI*, 2011.

[176] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3d interpreter network. In *ECCV*, 2016.

[177] K. Wu and M. D. Levine. Recovering parametric geons from multiview range data. In *CVPR*, 1994.

[178] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[179] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[180] J. Xiao, B. Russell, and A. Torralba. Localizing 3D cuboids in single-view images. In *NeurIPS*, 2012.

[181] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *TPAMI*, 2016.

[182] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[183] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. PAMI*, 2013.

[184] L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher. Make it home: automatic optimization of furniture arrangement. In *SIGGRAPH*, 2011.

[185] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.

[186] E. Zhang, M. F. Cohen, and B. Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 2016.

[187] Y. Zhang, S. Song, P. Tan, and J. Xiao. PanoContext: A whole-room 3d context model for panoramic scene understanding. In *ECCV*, 2014.

[188] Y. Zhao and S.-C. Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013.

[189] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *ECCV*, 2016.

[190] Y. Zhou, S. Liu, and Y. Ma. Learning to detect 3d reflection symmetry for single-view reconstruction. *arXiv preprint:2006.10042*, 2020.

[191] Zillow. Zillow Digs. `http://www.zillow.com/digs/`.