Data-Driven Geometric Scene Understanding

Scott Satkin

CMU-RI-TR-13-19

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Robotics
July 2013

Doctoral Committee:

Professor Martial Hebert, Chair
Professor Alexei Efros
Professor Abhinav Gupta
Professor Derek Hoiem, University of Illinois at Urbana-Champaign

ABSTRACT

Data-Driven Geometric Scene Understanding
Scott Satkin

In this thesis, we describe a data-driven approach to leverage repositories of 3D models for scene understanding. Our ability to relate what we see in an image to a large collection of 3D models allows us to transfer information from these models, creating a rich understanding of the scene. We develop a framework for auto-calibrating a camera, rendering 3D models from the viewpoint an image was taken, and computing a similarity measure between each 3D model and an input image. We demonstrate this data-driven approach in the context of geometry estimation and show the ability to find the identities, poses and styles of objects in a scene.

We begin by presenting a proof-of-concept algorithm for matching 3D models with input images. Next, we present a series of extensions to this baseline approach. Our goals here are three-fold. First, we aim to produce more accurate reconstructions of a scene by determining both the exact style and size of objects as well as precisely localizing their positions. In addition, we aim to increase the robustness of our scene-matching approach by incorporating new features and expanding our search space to include many viewpoint hypotheses. Lastly, we address the computational challenges of our approach by presenting algorithms for more efficiently exploring the space of 3D scene hypotheses, without sacrificing the quality of results.

We conclude by presenting various applications of our geometric scene understanding approach. We start by demonstrating the effectiveness of our algorithm for traditional applications such as object detection and segmentation. In addition, we present two novel applications incorporating our geometry estimates: affordance estimation and geometry-aware object insertion for photorealistic rendering.

ACKNOWLEDGEMENTS

"If I have seen further it is by standing on the shoulders of giants." –Isaac Newton (1676)

This thesis would not have been possible without the contributions of many, some via their direct scientific collaboration, others indirectly by their positive influence on me over the years. First and foremost, I would like to thank my advisor Martial Hebert for his unparalleled dedication to his students, for granting me the freedom to work on interesting problems, sheltering me from our funding sources, and keeping me focused year after year. And to Robert Pless, who introduced me to the field of Computer Vision, and inspired me to pursue a Ph.D.

To my thesis committee members: Derek Hoiem, whose pioneering work on spatial reasoning paved the way for the research in this thesis. Abhinav Gupta, who opened the doors to geometry estimation for me with our affordances work. And to Alyosha Efros, for our many late-night research arguments – your passion has made a profound impact on me.

To Kevin Karsch, for performing all the photorealistic renderings seen in Chapter VI. To Jason Lin, for his contributions to our initial scene matching prototype. To Maheen Rashid, for her contributions to refinement via object swapping. And to Varsha Hedau, David Lee, Daniel Muñoz and David Fouhey for sharing their code and features.

To Ryan Johns, who's ninja-like engineering helped me recover from a catastrophic data loss. And to all my friends in San Francisco, for helping me keep my eye on the prize, and for providing a couch to crash on. And to my housemates, for putting up with me for all these years.

To the entirety of The Robotics Institute, especially the vision group, for fostering an environment which encourages creativity and demands excellence. To my labmates, especially Edward Hsiao, Daniel Muñoz, Ekaterina Taralova, Yuandong Tian, Stéphane Ross, Carl Doersch, David Fouhey, Santosh Divvala and Tomasz Malisiewicz for the years of interesting discussions, much needed group dinners, and endless encouragement/commiseration.

And to Suzanne Lyons Muth, for ensuring the wellbeing of every student in the Robotics Instute.

To everyone I have had the privilege of working with at Bell Labs and Google, especially Wim Sweldens, Rajeev Rastogi, Chuck Rosenberg and Jean-Yves Bouguet. And to Chris Quackenbush, for bravely moving with me to India to seek employment.

And of course, my family. Whether nature or nurture, you are responsible. Thank you for your unconditional support, trust and encouragement.

TABLE OF CONTENTS

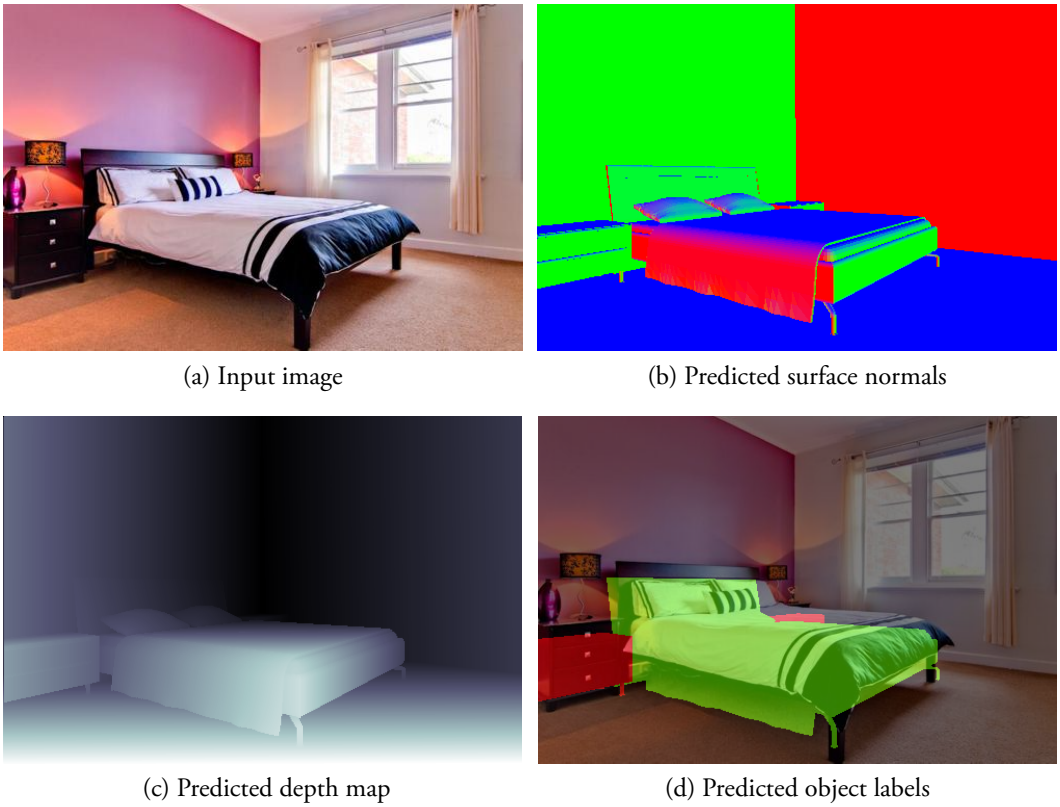
CHAPTER I

Introduction

This thesis explores the intersection of geometric reasoning and machine learning for scene understanding. Our objective is to produce a rich representation of the world from a single image by relating what we see in the image with vast repositories of 3D models. By matching and aligning an image with 3D data, we can produce detail reconstructions of an environment and transfer rich information from the models to answer a wide variety of question about the world. Our work builds upon recent advances in data-driven scene matching and single-view geometry estimation, which we now summarize.

1.1 Data-Driven Approaches in Computer Vision

Over the past decade, researchers have demonstrated the effectiveness of data-driven approaches for complex computer vision tasks. Large datasets such as [Torralba et al., 2008]'s 80 Million Tiny Images and [Deng et al., 2009]'s ImageNet have proven to be invaluable sources of information for tasks like scene recognition and object classification. Simple nearest-neighbor approaches for matching an input image (or patches of an image) with a large corpus of annotated images enables the "transfer" of information from one image to another. These non-parametric approaches have been shown to achieve amazing performance for a wide variety of complex computer vision and graphics tasks ranging from semantic labelling (e.g., [Tighe and Lazebnik, 2010], [Sing and Košecká, 2013]) and scene categorization [Oliva and Torralba, 2001], to motion synthesis [Liu et al., 2008] and even image localization [Hays and Efros, 2008].



(a) Input image (b) Predicted surface normals

(c) Predicted depth map (d) Predicted object labels

Figure 1.1: From a single image, we estimate detailed scene geometry and object labels.

Recently, large online repositories of 3D data such as 3D Warehouse [Trimble Inc., 2012] (formerly Google 3D Warehouse) have emerged. These resources, as well as the advent of low-cost depth cameras such as the Kinect [Microsoft Corporation, 2010], have sparked interest in geometric data-driven algorithms. At the same time, researchers have (re-)started investigating the feasibility of recovering geometric information, for example, the layout of a scene [Hoiem et al., 2007a, Saxena et al., 2009, Bao et al., 2010]. The success of data-driven techniques for tasks based on appearance features, for example, interpreting an input image by retrieving similar scenes [Torralba et al., 2008, Liu et al., 2008, Hays and Efros, 2007], suggests that similar techniques based on *geometric* data could be equally effective for 3D scene interpretation tasks. In fact, the motivation for data-driven techniques is the same for 3D models as for images: Real-world environments are not random; the sizes, shapes, orientations, locations and co-location of objects are constrained in complicated ways that can be represented given enough data. In principle, estimating 3D scene structure from data would help constrain bottom-up vision processes. For example, in Figure 1.1, one nightstand is fully visible; however, the second nightstand is almost fully occluded.

Although a bottom-up detector would likely fail to identify the second nightstand since only a few pixels are visible, our method of finding the best matching 3D model is able to detect these types of occluded objects. This is not a trivial extension of the image-based techniques. Generalizing data-driven ideas raises new fundamental technical questions never addressed before in this context: What features should be used to compare input images and 3D models? Given these features, what mechanism should be used to rank the most similar 3D models to the input scene? Even assuming that this ranking is correct, how can we transfer information from the 3D models to the input image?

To address these questions, we develop a set of features that can be used to compare an input image with a 3D model and design a mechanism for finding the best matching 3D scene using support vector ranking. We show the feasibility of these techniques for transferring the geometry of objects in indoor scenes from 3D models to an input image.

The graphics community has begun harvesting data from online repositories such as Google 3D Warehouse in an effort to better understand and model how objects are typically arranged in homes [Fisher and Hanrahan, 2010, Fisher et al., 2011]. Additionally, the vision community has begun utilizing 3D Warehouse data to learn about the sizes, shapes and affordances of objects [Grabner et al., 2011, Zhao and Zhu, 2013]. There has also been work using this data for 3D to 3D matching with laser scans to aid in classification [Lai and Fox, 2009]. However, our work is one of the first to combine this geometric prior with image features in a framework capable of producing detailed 3D models from an image. Of course, this is not entirely new, the idea of relating 3D models to 2D projections was a foundation of earlier vision approaches [Lowe, 1987, Brooks, 1981, Grimson et al., 1990]. The major difference here is our use of vast repositories of 3D data, which require novel vision and learning approaches.

This work is an important first step towards 3D data-driven techniques, which will contribute to addressing two major problems in image understanding. First, as most geometric scene understanding systems rely implicitly on sifting through a collection of hypotheses (iterative refinement [Hoiem et al., 2008], sampling [Pero et al., 2011], explicit search [Gupta et al., 2010], structured prediction [Lee et al., 2010, Hedau et al., 2009]), matching and ranking mechanisms such as the ones we propose provide a data-driven way to *generate multiple hypotheses* which can be used as seeds for further processing. Second, 3D data offers potentially richer information for transfer. In this thesis, we show that using 3D information for scene understanding enables us predict not only object type and location, but also viewpoint and even occlusions from other objects.

1.2 Single-View Geometry Estimation

For decades, vision researchers have strived to create high-quality 3D models of indoor scenes. Traditional approaches rely on having images taken from multiple viewpoints in order to recover the depth of each pixel using triangulation [Longuet-Higgins, 1981]. However, in recent years, the vision community has begun to focus on recovering the geometry of a scene from a single image [Hoiem et al., 2007a, Saxena et al., 2009, Lee et al., 2009].

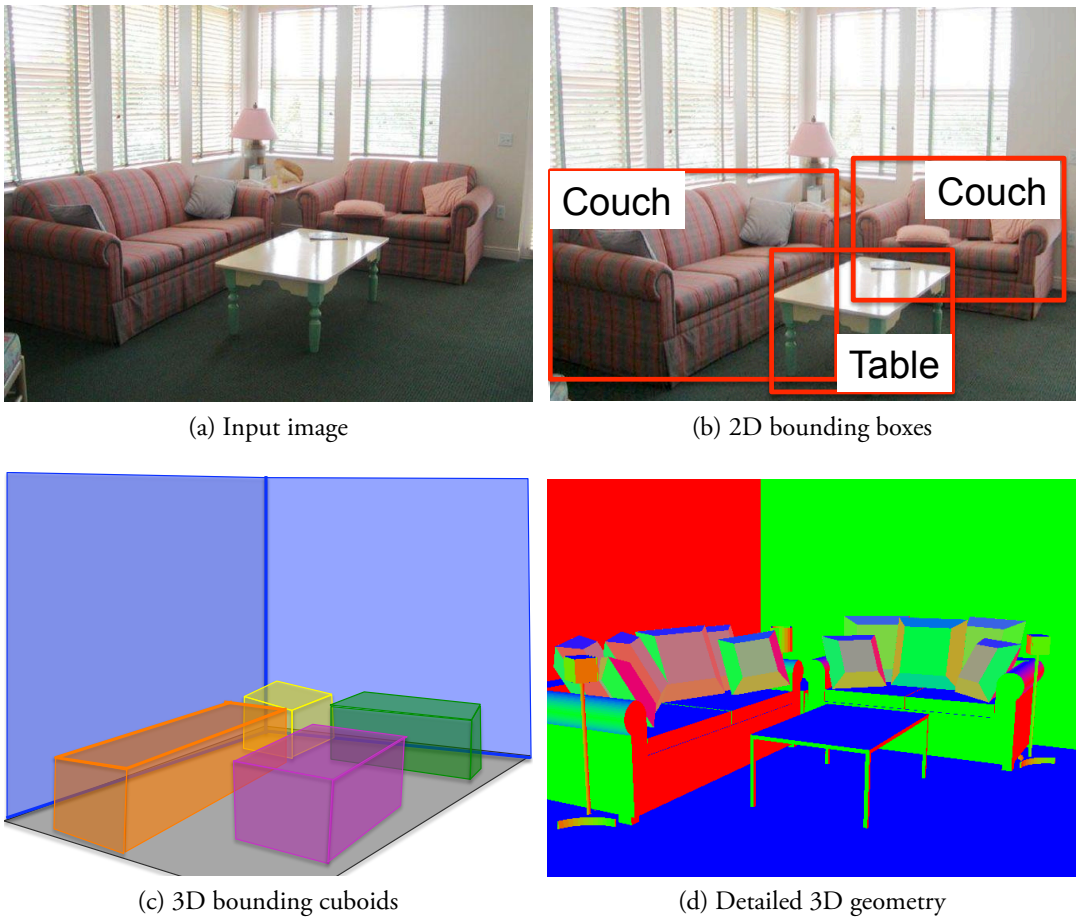
This is an inherently ill-posed problem – there exists an infinite number of 3D models which project to the same image. Despite the inherent mathematical ambiguity, humans excel at this task. When shown an image of an environment, we are not overwhelmed with an infinite number possible 3D models. On the contrary, we can quickly associate objects in images with objects we have seen before, to reason about the structure of the scene.

We are capable of this type of reasoning because the environments we live in are not completely random. The sizes, shapes, orientations, locations and co-location of objects are all dictated by the activities which the environment was designed to afford. For example, there are manufacturing standards for sizes of beds, couches, tables, etc. Moreover, when we place these objects in our homes, we tend to place them in specific locations relative to each other (e.g., nightstands adjacent to beds, coffee tables ~2ft in front of couches). When confronted with the ill-posed problem of recovering the geometry of scene from a single image, we must exploit this statistical prior and only consider 3D models which contain reasonable objects, in reasonable arrangements.

The traditional paradigm of training on one set of monocular images, and evaluating our accuracy on another, does not allow us to capture and model the spatial distribution of objects in 3D. Therefore, in this work, we leverage a massive online repository of 3D models to capture the distributions of object sizes, positions, orientations and locations.

Recently, tremendous progress has been made towards the task of estimating the geometry of a scene from a single image. The groundbreaking work of [Hoiem et al., 2007a] and [Saxena et al., 2009], showed that machine learning can be be used to tackle this tremendously challenging task. The authors of these papers demonstrate that classifiers can be trained to predict the orientation and identity of image patches from outdoor scenes, which can be used to infer the 3D structure of the environment.

For indoor imagery, [Yu et al., 2008] and [Lee et al., 2009] showed that imposing a Manhattan-world constraint enables the robust detection of vanishing points allowing cameras to be autocalibrated from a single image. The authors use their model to detect planes



(a) Input image (b) 2D bounding boxes

(c) 3D bounding cuboids (d) Detailed 3D geometry

Figure 1.2: Comparison of scene representations. In order of increasing geometric detail: traditional 2D bounding boxes (b), 3D bounding cuboids (c), our detailed 3D scene geometry (d).

and infer depth ordering to estimate the locations of the wall, floors and ceilings.

A fundamental problem when estimating the locations of walls in an indoor environment is clutter. Quite frequently, furniture or other objects will occlude the boundary between walls and where the walls meet the floor. [Hedau et al., 2009] train a classifier to predict which pixels are the result of clutter, and which pixels correspond to the walls and floor of a room. [Wang et al., 2010] also predict which pixels correspond to occluding objects; however, their technique uses latent variables and avoids the need for labeled training data. Both of these approaches show that by identifying the locations of clutter in a scene, the room layout can be more accurately estimated.

More recently, researchers have begun analyzing the detected clutter in a scene and try-

ing to model it with 3D bounding cuboids. This representation, depicted in Figure 1.2c offers more information than traditional 2D bounding boxes (Figure 1.2b), which only localize objects in the image plane. Cuboids can be used to reason about a scene in ways which cannot be inferred from 2D bounding boxes, such as estimating the freespace of an environment or analyzing where the supporting surfaces of objects are. [Lee et al., 2010] combine the geometric context used in [Hedau et al., 2009] with an orientation map to fit a parametric model to objects in a room with the goal of improving the estimates of wall locations. [Pero et al., 2011] use 3D bounding cuboids fit to objects as part of a Markov Chain Monte Carlo framework to optimize over both the locations of the walls as well as camera pitch, roll and focal length. Both of these works show that modeling the clutter in a room improves the accuracy of room layout estimation; however, the authors do not evaluate how well their cuboids match the geometry of the objects in the scene.

In [Hedau et al., 2010], the authors build upon their previous work by incorporating a cuboid detector capable of accurately detecting beds. Their algorithm searches for gradients in images which have been rectified to estimated Manhattan-world axes, and tries to align cuboids of fixed sizes (corresponding to common beds). Unlike previous work which aims only to recover the locations of walls and floors in images, this work strives to detect and align objects in scenes and evaluates their results using typical object detection metrics.

Rather than representing objects and freespace with coarsely voxelized occupancy grids or bounding cuboids, we aim to produce high-quality detailed polygonal meshes of objects, as shown in Figure 1.2d. We build upon and use the room layout estimates of [Hedau et al., 2009], and mine through a database of 3D models to discover the identity, locations and orientations of objects from a single image. New work such as [Hedau et al., 2012, Pero et al., 2012, Zhao and Zhu, 2013] also aims to recover free space by localizing cuboids representing object categories and sizes using parametric models as their prior. In contrast, we recover more detailed object geometries and we use non-parametric priors that can capture complex interactions between objects.

For each object in a scene, we aim to not only recover its exact location, orientation and dimensions in 3D (which can be modeled with cuboids), but also a detailed polygonal model of the object. In addition, we aim to recover the intrinsic (focal length and principal point) and extrinsic (position and rotation relative to corner of the room) parameters of the camera which captured the image. In this thesis, we show that recovering the detailed geometry of a scene and the corresponding camera parameters offers a complete representation of the world, which can be used to answer a wide variety of questions. For example, using the

estimated camera parameteres, we can project the polygons of each object onto the image plane to produce a segmentation mask (Figure 1.1c). We can compute the distance from the camera to each object to produce depth maps and reason about occlusions and depth ordering (Figure 1.1d).

Although there exist many sensors which are designed to capture 2.5D representations of the world, such as laser scanners and RGBD cameras, these modalities capture only what is visible from a single viewpoint. On the contrary, because we have a *full 3D* representation of the scene, we can reason about portions of the environment which are not visible to the camera. For example, in the bedroom scene in Figure 1.1, only a small portion of the nightstand in the corner of the room is visible, and the strip of floor between the bed and the wall is fully occluded. A 2.5D sensor will have no knowledge of these portions of the environment; however, our full 3D representation of the scene includes the geometry of these regions. This information cannot be measured directly, and must be inferred using prior knowledge of the world.

This brief summary of previous work shows how vibrant this research area is and how much progress has been made in a short time. The data-driven techniques that we propose here should not be viewed as a substitute to any of the above approaches. Perhaps the most exciting aspect of our approach is that it can be used to augment *any* of these scene interpretation approaches: upstream, by providing a data-driven way to generate hypotheses; and downstream by providing richer mechanisms for information transfer. We show this by building upon the work of [Hedau et al., 2009] and by demonstrating how prior 3D models can be integrated with this existing approach for room layout estimation to help discover the identity, locations and orientations of objects from a single image.

CHAPTER II

Scene Understanding via 3D Model Matching

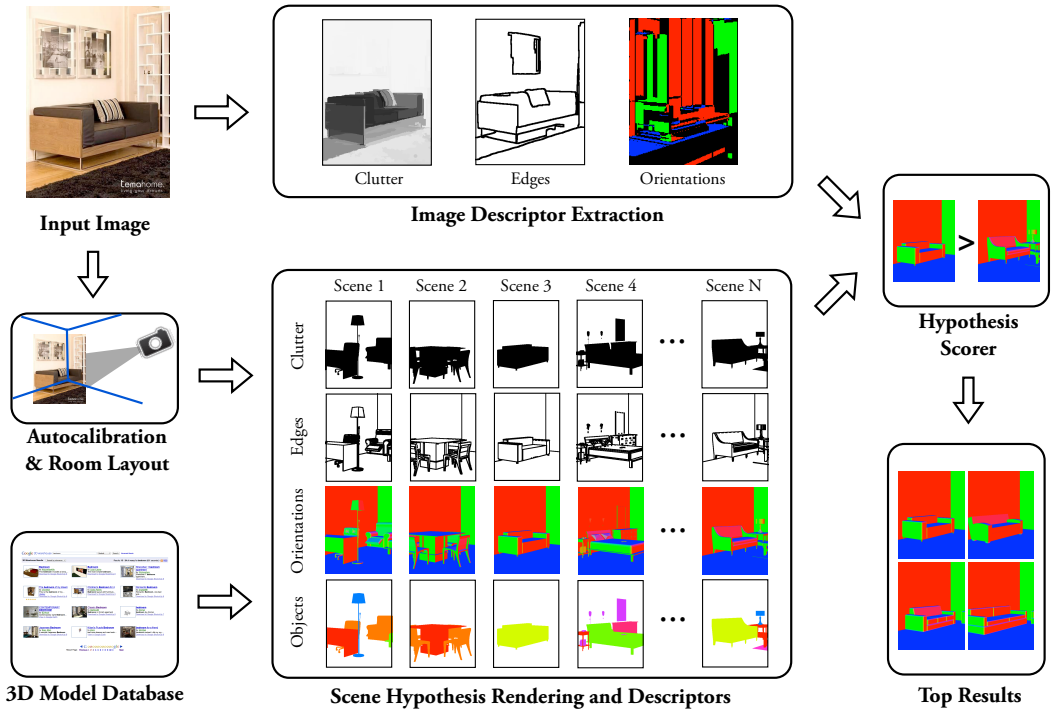


Figure 2.1: Overview of our approach for matching a 3D model with a monocular image.

We now describe our framework for comparing 3D models to monocular images. Our ability to relate what we see in an image to a large collection of 3D models allows us to transfer the information from these models, creating a rich understanding of the scene. Naturally, we cannot compare 3D models directly to a 2D image. Thus, we first estimate the intrinsic and extrinsic parameters of the camera and use this information to render each

of the 3D models from the same view as the image was taken from. We then compute similarity features between the models and the input image. Lastly, each of the 3D models is ranked based on how similar its rendering is to the input image using a learned feature weighting. See Figure 2.1 for an overview of this process. This proof-of-concept scene matching approach was presented at the 2012 British Machine Vision Conference [Satkin et al., 2012b].

2.1 Autocalibration and Room Layout Estimation

Our algorithm for recovering the geometry of a scene is an *analysis via synthesis* approach. We render 3D models from the viewpoint an image was captured and compare these renderings to the input image. This requires we first recover the parameters of the camera used to capture the image via auto-calibration and room layout estimation.

We begin auto-calibrating the camera by estimating vanishing points using the approach of [Lee et al., 2009]. The vanishing points are also used to estimate the orientation of the camera with respect to the three Manhattan-world axes in our scene (see Appendix C for details of this process and an error analysis). Next, we run the pre-trained room layout estimation algorithm of [Hedau et al., 2009] to determine the locations of the walls and the floor in the image, and use priors on camera height and room size to solve for the position of the camera. We render hypothesized 3D models of scenes using OpenGL. We align the walls of the models with the estimated wall locations relative to the camera, and incorporate our calibrated camera parameters (focal length and principal point) with a viewing frustum to create renderings which align with our input image.

There are three components to our projection matrix, summarized in Equation 2.1, which maps points in 3D world coordinates to homogeneous 3D normalized device coordinates for rendering. First we apply an extrinsic calibration matrix to transform points from world coordinates to camera coordinates. Next, we apply an intrinsic calibration matrix which unwarps the perspective effects due to the camera's optics. Lastly, we apply an orthographic projection matrix which maps points in the viewing frustum to points in normalized coordinates which can be clipped and rendered. We now derive this projection matrix.

$$\begin{pmatrix} x_{\text{clip}} \\ y_{\text{clip}} \\ z_{\text{clip}} \\ w_{\text{clip}} \end{pmatrix} = \mathbf{P}_{\text{orthographic}} \cdot \mathbf{P}_{\text{intrinsic}} \cdot \mathbf{P}_{\text{extrinsic}} \cdot \begin{pmatrix} X_{\text{world}} \\ Y_{\text{world}} \\ Z_{\text{world}} \\ 1 \end{pmatrix} \tag{2.1}$$

The orthographic projection matrix maps a viewing frustum defined by the left, right, top, bottom, near, and far clipping planes to the unit cube with coordinates in the range $[-1, 1]$. This matrix is composed by first translating and then scaling each coordinate to arrive at $\mathbf{P}_{\text{ortho}}$ in Equation 2.3:

$$\mathbf{P}_{\text{ortho}} = \begin{pmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & 0 \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & 0 \\ 0 & 0 & \frac{-2}{\text{far}-\text{near}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\frac{\text{right}+\text{left}}{2} \\ 0 & 1 & 0 & -\frac{\text{top}+\text{bottom}}{2} \\ 0 & 0 & 1 & -\frac{\text{far}+\text{near}}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

$$= \begin{pmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & \frac{\text{right}+\text{left}}{\text{left}-\text{right}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & \frac{\text{top}+\text{bottom}}{\text{bottom}-\text{top}} \\ 0 & 0 & \frac{-2}{\text{far}-\text{near}} & \frac{\text{far}+\text{near}}{\text{near}-\text{far}} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.3}$$

To unwarp the perspective effects due to camera optics, we begin with the intrinsic calibration matrix $\mathbf{K}$, which is computed from three vanishing points using an orthogonality constraint (see Appendix C). To convert from the left-handed coordinate system used by most vision researchers to the right-handed coordinate system used by the graphics community, we negate the z dimension of our intrinsic calibration matrix. In addition, to maintain the depth information required for rendering, we add a row and column to the projection matrix which preserves the Z-coordinate of points (after normalizing by w_{clip}). Note that our calibration matrix assumes zero pixel skew ($s = 0$) and unit aspect ratio ($f_x = f_y$).

$$\mathbf{K} = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad \Longrightarrow \quad \mathbf{P}_{\text{intrinsic}} = \begin{pmatrix} f & 0 & -u_0 & 0 \\ 0 & f & -v_0 & 0 \\ 0 & 0 & \text{near}+\text{far} & \text{near}\cdot\text{far} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{2.4}$$

Lastly, to convert points from world coordinates to camera coordinates, we apply an extrinsic calibration matrix which is composed of a rotation and translation. The columns of the rotation matrix $\mathbf{R}$ are computed from the estimated vanishing points, and represent unit vectors along each of the Manhattan-world axes, in the camera's coordinate frame. The translation vector $\mathbf{t}$ indicates the position of the camera relative to the origin of world's coordinate system, which we define to be a visible corner of the room. This camera position is computed using [Hedau et al., 2009]'s room layout estimation algorithm, which predicts

the locations of the walls and floor in an image. To resolve the scale ambiguity, we first fix the height of the camera to 5ft, and solve for the distance from the camera to the visible corners of the room. If the computed room size is too large or small (e.g., height $\geq$ 12ft or height $\leq$ 8ft), we raise or lower the height of the camera in 0.5ft increments and re-solve for the room size until the scale is within range. The rotation and translation are integrated into the extrinsic camera matrix $\mathbf{P}_{\text{extrinsic}}$:

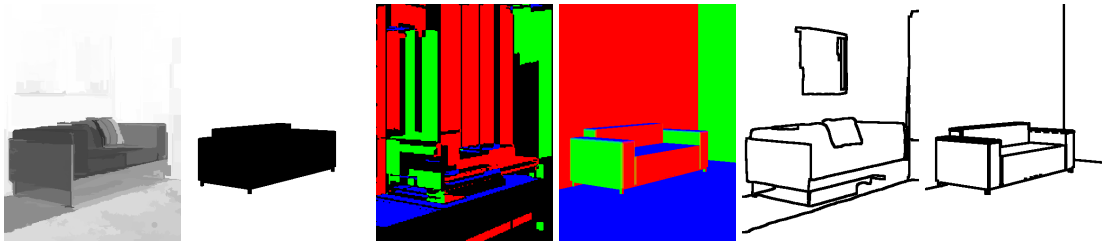
$$\mathbf{P}_{\text{extrinsic}} = \begin{pmatrix} \mathbf{R} & \mathbf{R}\cdot\mathbf{t} \\ 0^\top & 1 \end{pmatrix}. \tag{2.5}$$

Each of these transformations is chained together to produce our final projection and rendering matrix:

$$\mathbf{P} = \begin{pmatrix} \frac{2f}{\text{width}} & 0 & \frac{-2u_0}{\text{width}} + 1 & 0 \\ 0 & \frac{2f}{\text{height}} & \frac{2v_0}{\text{height}} - 1 & 0 \\ 0 & 0 & \frac{\text{far}+\text{near}}{\text{near}-\text{far}} & \frac{2\,\text{far}\cdot\text{near}}{\text{near}-\text{far}} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} & | & & | \\ — & \mathbf{R} & — & \mathbf{R}\cdot\mathbf{t} \\ & | & & | \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.6}$$

This setup allows us project objects from our 3D model library into the image plane in a manner which is consistent with the estimated camera parameters. We use this renderer as a fundamental tool in computing similarity features from each 3D model. The following section details this process.

2.2 Similarity Features



(a) p(object) (b) object mask (c) estimated normals (d) rendered normals (e) gPb (f) rendered edges

Figure 2.2: Descriptors extracted from an input image (a,c,e), and their corresponding rendered descriptors from the top-ranked 3D model (b,d,f).

An important question we address in this thesis is, "What features are useful for matching 3D scenes with monocular images?" This issue is fundamentally complicated by the fact that we need to compare two objects of a completely different nature: an array of intensity/color pixels on the one hand, and a set of surfaces with little or no appearance information on the other hand.

To overcome this challenge, we introduce the concept of *similarity features*. Unlike traditional features which are extracted from a single image, similarity features involve comparing an image with a 3D model to describe how similar the model is to the input image. Our goal here is to rank each 3D model j, with respect to image i using similarity features denoted by x_j^i. x_j^i is a vector in which each entry corresponds to a different measure of similarity between the image i and the 3D model j. We use our renderer to produce synthetic image descriptors for each 3D model, and compare these to traditional image-based descriptors to compute each similarity feature. Figure 2.2 includes example image descriptors and their rendered counterparts used to compute each similarity feature. Note that these are not photorealistic renderings, we are simply rendering descriptors of each 3D model. This section introduces our preliminary set of similarity features for relating 2D images with 3D models.

Object masks: We use the pre-trained Indoor Geometric Context model of [Hedau et al., 2009, Hoiem et al., 2007a] to estimate the likelihood that each of the pixels in an image contains an object (see Figure 2.2a). For each 3D model, we render a simple object mask (i.e., each polygon in the model is rendered black on a white background) as shown in Figure 2.2b. After scaling each of these masks to be in the range $[-1, 1]$, the dot product between the predicted object locations and the rendered object masks indicate how well the

model matches our image. This similarity measure is the first feature we use to compare 3D models to an input image.

Surface normals: We use the plane-sweeping algorithm of [Lee et al., 2009] to predict the surface normal of each pixel in an input image. For each 3D model, we render a surface normal image, by simply setting the red, green and blue color components of each polygon to the x, y and z components of the polygon's surface normal. See Figures 2.2c and 2.2d for examples of predicted surface normals, and rendered surface normals. The normalized dot product of these two descriptors quantifies their similarity. We use this value as a feature when scoring each 3D model.

Masked surface normals: We also combine our object masks and surface normal descriptors to create a highly-informative hybrid feature. For this feature, we multiply the object mask agreement score with the surface normal agreement score for each pixel (both scaled to be in the range $[0, 1]$). This combined score aims to count how many pixels in the image satisfy two constraints: Firstly, objects in the renderings should appear only where they are predicted to be. Secondly, the surface normals of the 3D models at these locations should also agree with the predicted surface normals.

Edges: We extract edges from an input image using the globalPb algorithm [Arbelaez et al., 2011] (thresholded at $p(\text{boundary}) > 0.5$). These edges are compared to Canny edges which are extracted from rendered surface-normal images of each scene hypothesis (Figures 2.2e and 2.2f). Pairs of edge images (extracted from the input image and each rendering) are compared using a modified symmetric Chamfer distance ($a \in A$ indicates a is an edge pixel in image A):

$$\Delta_{edge}(A, B) = \frac{1}{|a|} \sum_{a \in A} \min \left(\min_{b \in B} \|a - b\|, \tau \right) + \frac{1}{|b|} \sum_{b \in B} \min \left(\min_{a \in A} \|b - a\|, \tau \right). \quad (2.7)$$

To avoid the effect of outlier edges which do not match well, we truncate individual edge distance penalties ($\tau \in \{10, 25, 50, \infty\}$). Intuitively, distances computed with smaller values of τ encourage fine-grain matching of edges, while distances computed with larger thresholds aim to penalize large errors. Each of the distances computed with a different value of τ is treated as a separate feature, for a total of four features.

Additional features: Our 3D model matching framework is quite flexible and can easily be extended to incorporate additional similarity features. In Chapter 3.1, we will present an additional set of similarity features designed for more robust and precise scene matching, and analyze the performance of each of our proposed features.

2.3 Hypothesis Ranking

For a given input image, we render all of the 3D models in our scene library and compute similarity features from the renderings, as described above. We concatenate the object mask, surface normal, masked surface normal, and edge features into a 7-dimensional feature vector. A linear weighting of these similarity features is computed to determine a matching score indicating how similar each 3D model is to the given image.

We learn this weight vector by using a max-margin learning framework. Using annotated training data, we can rank how well each 3D model in our library matches each training image. We use a modified version of the masked surface normal score presented in Section 2.2 to compute a similarity score for each pair of images and 3D models. For this scoring, we do not use the predicted surface normals and object masks, we use renderings of hand-annotated scene geometries which are treated as ground-truth.

Our goal is to find a weight vector w which can correctly rank pairs of 3D scenes (i.e.: $w^\top x^i_j > w^\top x^i_k$ if scene j matches image i better than scene k). We use the difference in masked surface normal scores as the hinge loss margin δ^i_{jk}. This optimization takes the form of support vector ranking [Herbrich et al., 1999]:

$$\min_{w,\xi} \frac{\lambda}{2}\|w\|^2 + \sum \xi^i_{jk} \text{ s.t.: } w^\top x^i_j \geq w^\top x^i_k + \delta^i_{jk} - \xi^i_{jk},\ \xi^i_{jk} \geq 0. \tag{2.8}$$

We optimize Equation 2.8 using stochastic gradient descent. In each iteration, we select a training image i and a pair of 3D models (j, k). If the current weight vector causes the pair of 3D models to be incorrectly ranked, or if their difference in scores is less the margin δ^i_{jk}, we compute a subgradient and update the weight vector. This process is repeated until convergence.

Figure 2.3: An example query, and resulting "bedroom" models from 3D Warehouse.

2.4 Library of 3D Models

We acquired our 3D indoor scenes through the public search engine of 3D Warehouse [Trimble Inc., 2012]. We perform queries for common indoor scenes such as "bedroom" and "living room," and download top ranking models. Due to the nature of data harvested from the web, a large number of scenes are irrelevant for our task. For example, a query for "bedroom" may return a 3D model for a "4 Bedroom House," which contains only an architectural model of the building exterior. Using simple heuristics, we discard models which are too large or small. Although over 8000 3D models matched our queries, the majority of them were rejected based on these criterion, leaving approximately 2000 models in our database. To increase the size of our model database, we include 8 rotated and reflected versions of each scene. Each of these 3D models is then processed to segment individual components, save their polygonal faces, and identify object categories.

Each component has an associated label, such as "Couch," "Brown Leather Sofa," or "Love Seat." We automatically cluster the objects into groups by comparing their geometries with a simple 3D voxelized overlap score. This approach will discover that "Brown Leather Sofa" and "Love Seat" are synonyms of "Couch." To ensure accurate labeling of objects, we created a user-interface for quickly verifying or adjusting the label of each component.

Although our library contains only 2000 3D models, we experimentally found that this was not a major factor limiting our system's performance. See Appendix D for a detailed analysis of this issue. Moreover, we are agnostic to the source of our 3D data, and can incorporate additional models from other repositories or datasets to enable matching a wider variety of scenes. In Chapter 3.8, we demonstrate this ability to leverage additional sources of 3D data through experiments which utilize a dataset of 2D images and their corresponding 3D models for scene matching.

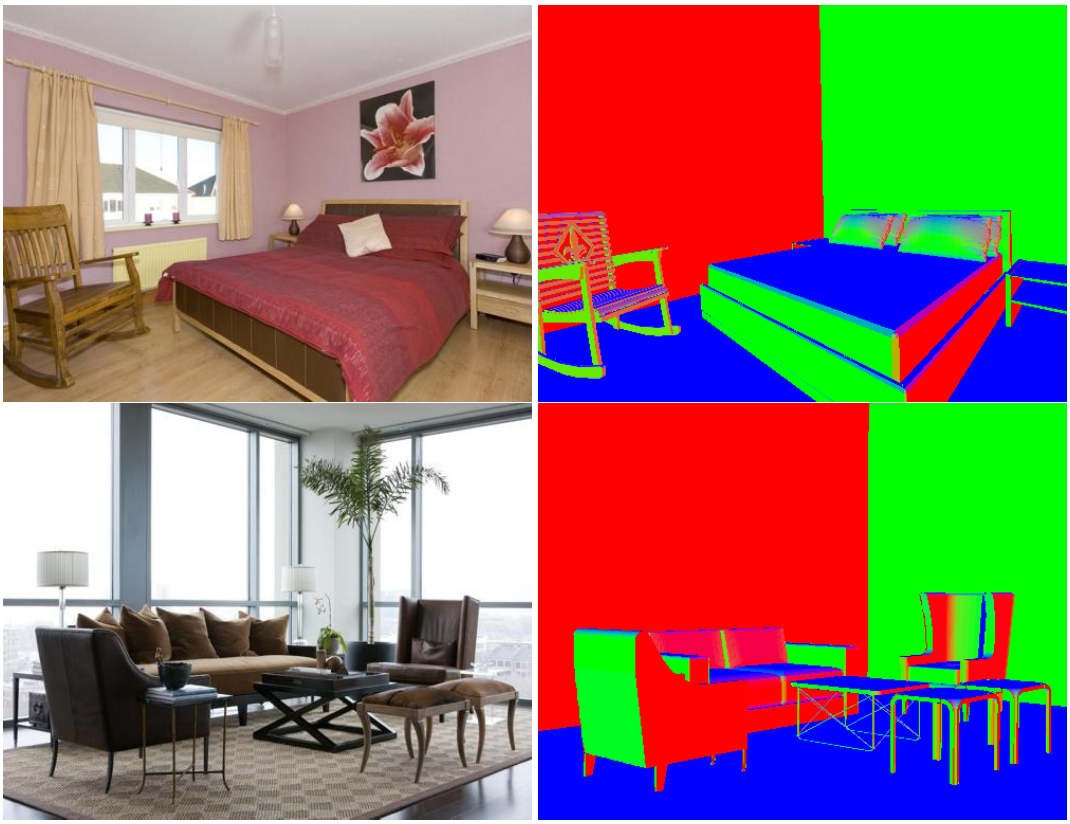


Figure 2.4: Example images and hand-crafted 3D models from our dataset.

2.5 Experimental Dataset and Ground-truth Annotation

Since the problem of monocular geometry estimation is relatively new, there does not yet exist an established dataset of images with detailed ground-truth object geometry and surface normals. Thus, we have created a new dataset with annotated scene geometry building upon the SUN database [Xiao et al., 2010]. Our dataset consists of over 500 images from the categories "bedroom" and "living room." For each image, a detailed 3D model was constructed using SketchUp [Google Inc., 2000]. This software allows users to label vanishing points for camera auto-calibration and insert existing 3D models of objects from the Internet to generate detailed models from an image.

We use these hand-crafted 3D models as ground-truth for training our scene ranker, as well as for evaluating the performance of our geometry estimation algorithm. Figure 2.4 includes example 3D models from our dataset. This dataset has been made publically available as the CMU 3D-Annotated Scene Database and is now being used by vision researchers at multiple universities [Satkin et al., 2012a].

2.6 Proof-of-Concept Scene Matching Results

In this section, we present a series of qualitative and quantitative results to demonstrate the capabilities of our scene matching technique. We partition our dataset into five folds (80% train, 20% test) and learn similarity feature weights for each fold independently to report performance on the entire dataset. Figure 2.5 shows example results of our algorithm. Displayed from left to right are the input images, estimated surface normals from the top-ranking matched 3D model, color-coded object overlays, and depth estimates.

Note that we are able to recover the labels, locations and orientations of objects, even from obscure viewpoints. For example, the image in Figure 2.6 was captured from behind a couch; however, we still correctly identify this unique layout of furniture, recovering the position and orientation of both couches and the coffee table. Moreover, the general style of objects is often matched (e.g., the headboard of each bed). Additionally, although appearance-based object detectors typically fail to detect mostly-occluded objects (such as nightstands), our holistic scene-matching approach is capable of finding these challenging objects.

These results highlight the benefits of using large repositories of 3D data for scene understanding. Unlike traditional approaches which represent objects with simple bounding boxes or cuboids, our algorithm produces detailed geometric representations. This unconventional paradigm for scene understanding requires new metrics to quantify performance. In the following sections, we will describe two families of evaluation metrics for measuring the accuracy of our estimated scene geometries both in the image plane as well as in 3D.

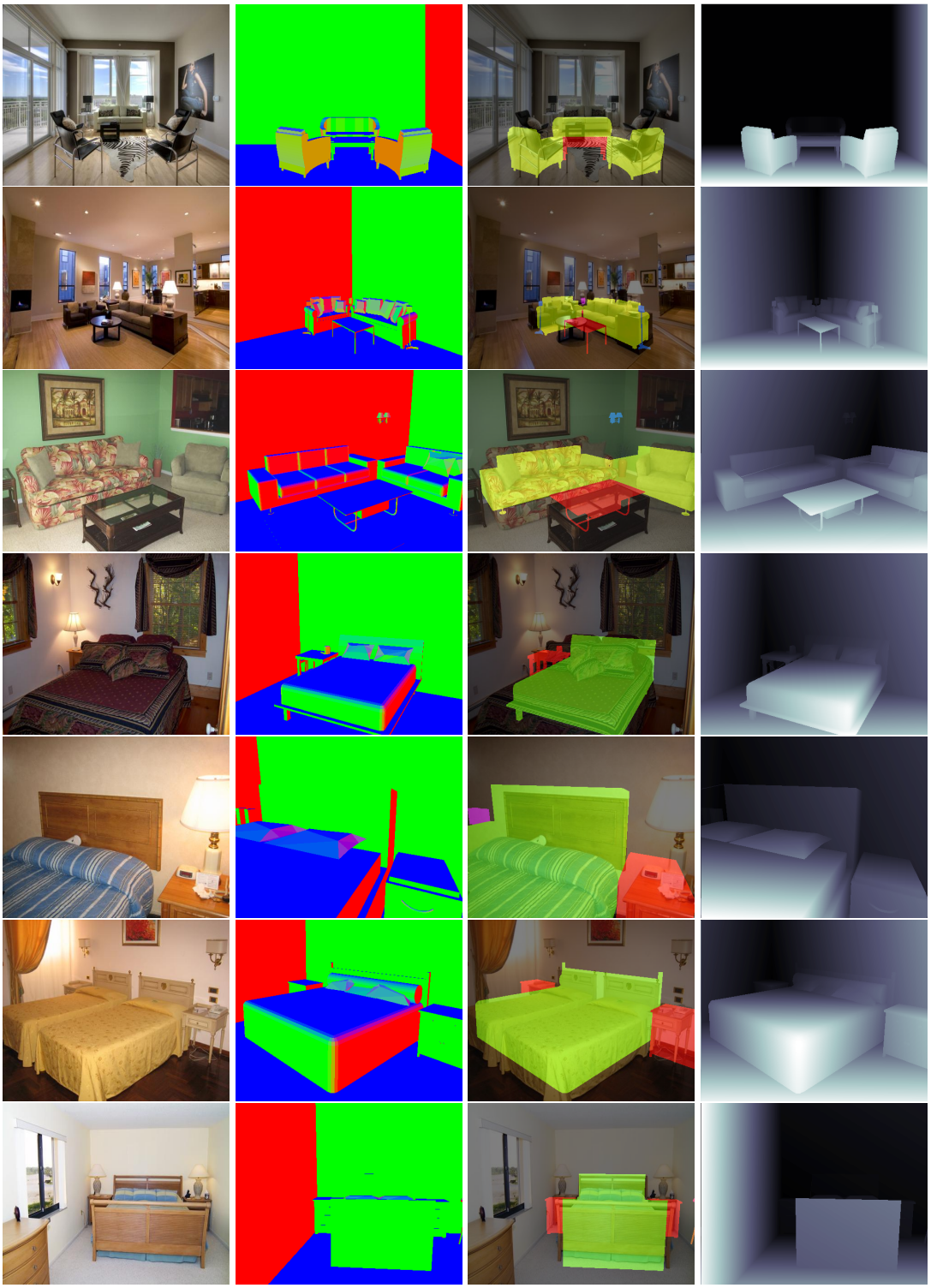


Figure 2.5: Input images, automatically-selected 3D models (surface normals displayed), color-coded object labels (yellow=couch, red=table, green=bed) and depth estimates.

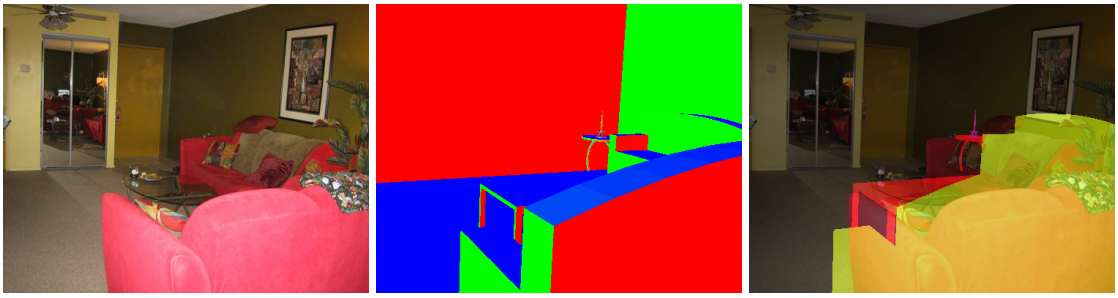


Figure 2.6: Example result showing our ability to recover the geometry of scenes captured from obscure viewpoints.

Figure 2.7: Failure cases of our scene matching system caused by incorrect vanishing point estimation, incorrect room layout estimation and poor hypothesis ranking.

2.6.1 Surface Normal Accuracy

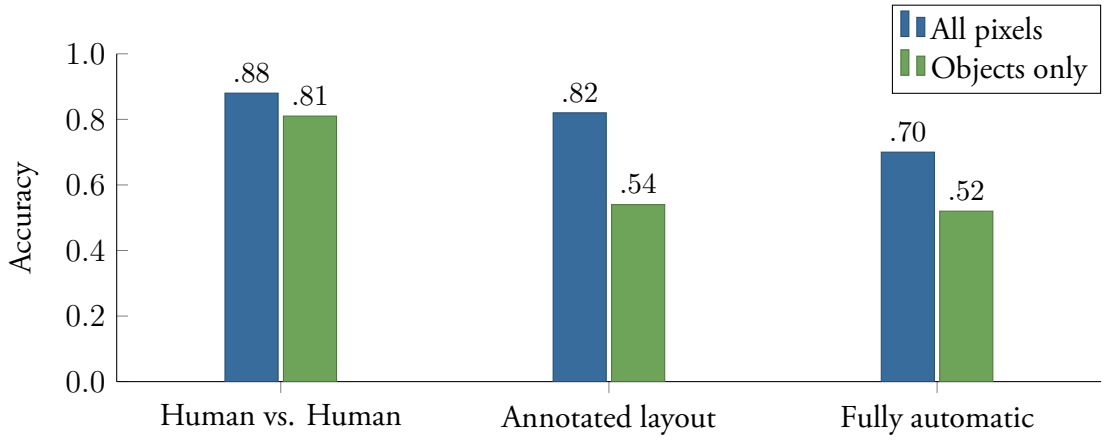


Figure 2.8: Surface normal scores for all pixels, and only pixels belonging to objects.

One way to quantify the quality of our 3D models is to measure how accurately we can predict the orientation of surfaces in the scene. We score our 3D hypothesis by taking the dot product of the ground-truth surface normals (from our annotated models) and the orientations of pixels in our rendered hypothesis, normalized by the number of pixels. This score represents the percentage of the pixels for which we have correctly identified the orientation. Since the majority of pixels in most scenes correspond to walls or floor, which are not informative to the quality of object geometries, we also report the surface normal score for only those pixels which belong to objects.

A fundamental issue with our approach is our reliance on autocalibration and room layout estimation. If this stage of the pipeline fails, we will incorrectly estimate the arrangement of object in a scene. For example, in Figure 2.7, the first scene shows a catastrophic failure due to incorrect vanishing point estimation (See Appendix C for a detailed analysis of vanishing point error.). For the second image, the location of the wall behind the bed is predicted to be too close to the camera. This caused our scene matching process to fit a couch, which has less depth than a bed. The last result shows correct vanishing point and room layout estimation, however our hypothesis ranker failed to select a good scene match. To decouple the effects of room layout estimation from our goal of determining the arrangement of objects, we report results using annotated room layouts and camera parameters as well as fully automatic results incorporating the room layout algorithm of [Hedau et al., 2009].

Two annotators created 3D models for a subset of our images in our dataset. By comparing these different versions of the scene geometries, we can evaluate the subjectivity of the task and annotation process. Figure 2.8 reports three sets of results. The first column measures how consistent humans are when annotating scene geometries. The next column reports the results of our algorithm for determining the arrangements of objects in a scene using annotated room layouts (camera parameters and wall locations). The last column reports the results of our fully-automated algorithm which uses the auto-calibration and room layout estimation techniques of [Hedau et al., 2009].

The large gap in performace between using annotated camera parameters and our fully-automated algorithm indicates a fundamental issue with our reliance on room layout estimation. Thus, in Chapter 3.4, we will present a mechanism for jointly selecting a combination of furniture and camera parameters, which together best match an image. This improvement is designed to increase our robustness to incorrect room layout estimates, narrowing the gap in performance seen in Figure 2.8.

2.6.2 Free Space Estimation

We also evaluate how accurately our algorithm can estimate the free space of a room from a single image. Figure 2.9 shows our ability to recover an architectural floorplan of a room. Note that we are able to identify which regions in space are occupied, estimate the distances between objects, and even make predictions about regions that are not visible in the input images due to occlusions.

To quantify this ability, we compare our estimated object locations to the ground-truth object locations from annotated images. For each square inch of the floor that we predict to be occupied, we compare to the ground-truth occupancy and report precision and recall. We run the pre-trained geometry estimation algorithm of [Gupta et al., 2011], and report their performance as a baseline. In addition, we use the evaluation metric proposed by [Hedau et al., 2012]. Their metric provides a soft-measure of object overlap. Hypothesized objects which are close to ground-truth object locations, but do not overlap, are scored based on their distance to the closest ground-truth object.

Figure 2.10 reports our precision and recall scores as well as [Hedau et al., 2012]'s "δ-precision" and "δ-recall" scores. Additionally, we report the F-measure (harmonic mean of precision and recall) which aims to capture our performance with a single value. We also compute a similar 3D free space evaluation by voxelizing our scenes and computing how precisely we estimate which voxels are occupied. Figure 2.11 reports the results of this free

Figure 2.9: Input images, automatically-selected 3D models, and overhead views (color indicates height: yellow=low, red=high). Results shown use annotated camera parameters.

space evaluation.

We also run our scene matching geometry estimation algorithm on the indoor scenes dataset from [Hedau et al., 2012]. Our algorithm performs comparably on this dataset as on our dataset and outperforms [Hedau et al., 2012]'s algorithm using their δ-precision/δ-recall metric for 2D floorplan occupancy and 3D voxelized accuracy. Figure 2.12 shows our performance (indicated in blue) compared to [Hedau et al., 2012]'s accuracy before and after their geometry refinement stage.[1]

[Guo and Hoiem, 2013] recently demostrated their ability to use context as a prior for inferring the semantic labels of hidden surfaces in a scene. Similarly, the experiments in this section demonstrate our ability to not only recover the geometry of each visible object in a scene, but to push the boundaries of spatial reasoning to estimate the full 3D geometry of a scene.

For example, the bedroom scene in Figure 2.9 contains a large region to the right of the bed which is occluded from the camera's viewpoint. In fact, there are entire objects in the image, such as the far nightstand which have no visible pixels. As humans, we are able to infer the structure of these regions using prior knowledge about the locations and co-locations of objects in different environments. Similarly, by leveraging large repositories of 3D models,

[1]Results from [Hedau et al., 2012] and our evaluation were performed using different code, slight variations in the evaluation process may effect the results. Plots from [Hedau et al., 2012] have been recreated for comparison.

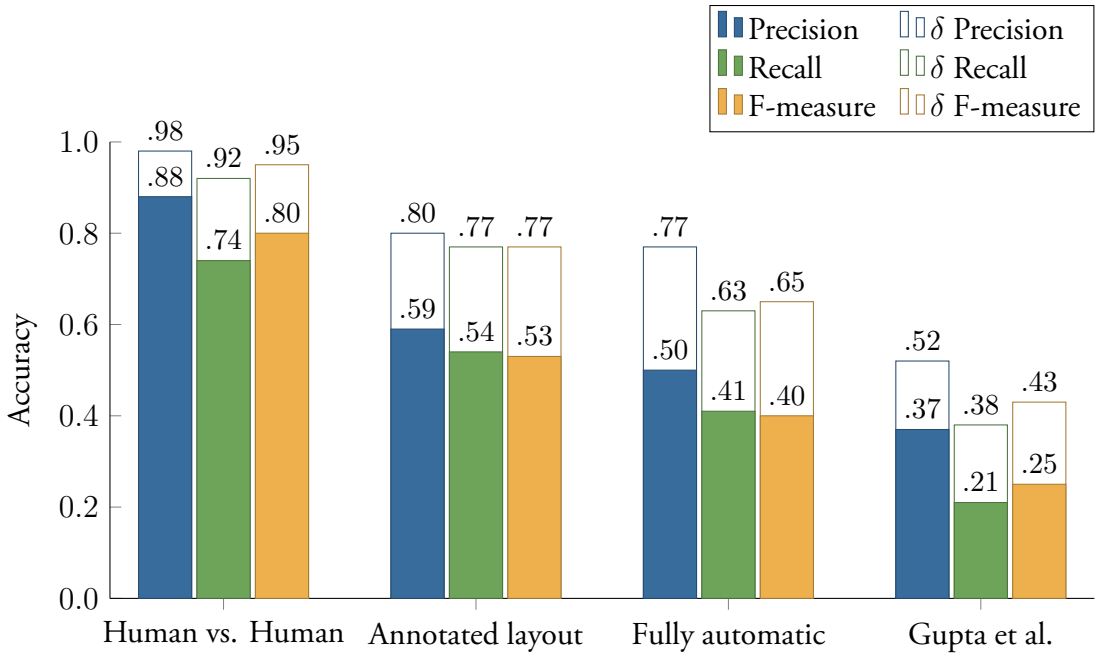


Figure 2.10: 2D floorplan free space evaluation.

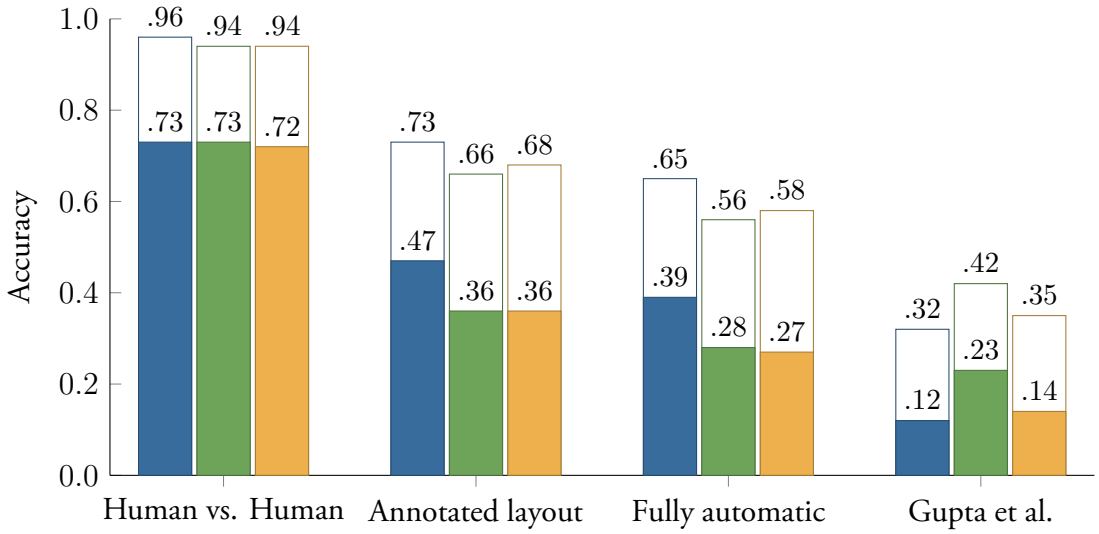


Figure 2.11: 3D floorplan free space evaluation.

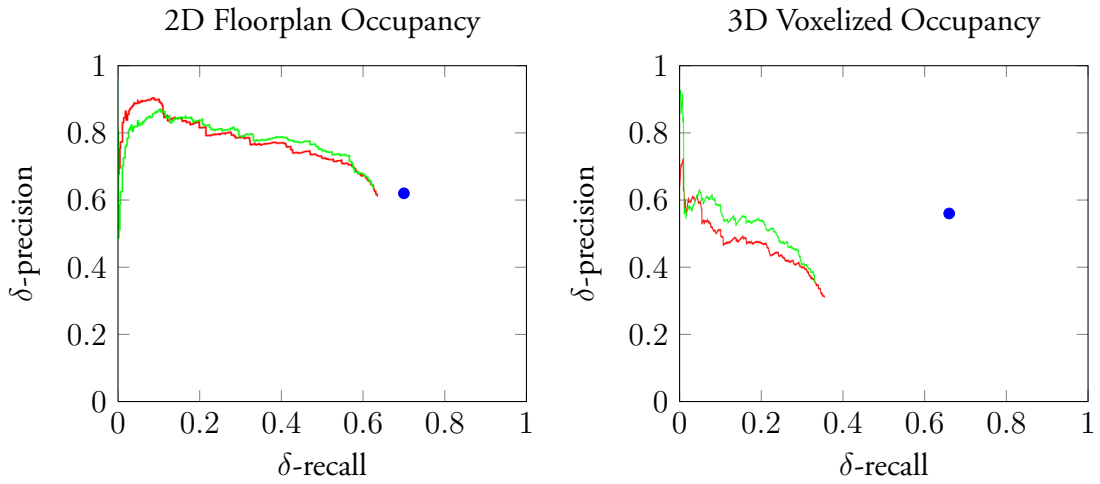


Figure 2.12: A comparison of our accuracy and [Hedau et al., 2012]'s performance on the indoor scenes dataset from Hedau et al. [Hedau et al., 2012]'s performance before and after their geometry refinement stage are indicated in red and green respectively. Our performace is indicated with the blue point on each graph.[1]

our scene understanding algorithm is able to predict what is likely to appear beyond the line of sight. Moreover, we can reason about the free space in an environment, enabling a wide variety of applications. For example, In Chapter V, we present an affordance estimation algorithm which uses this understanding of free space to predict what human poses and actions are possible in an environment.

The ability to perform this type of complex spatial reasoning can be used to enhance existing representation of a scene. For example, although laser scanners and depth cameras can precisely capture the structure of an environment, these sensors record 2.5D representations of the world, not full 3D models. Thus, they do not provide sufficient information to reason about portions of a scene which are occluded from the camera. Our approach could be applied to these modalities to reason about the unseen and recover the structure of objects which are not fully visible to the sensor, thus *inferring what cannot be measured*.

2.7 Discussion

This chapter presented a proof-of-concept algorithm for aligning 3D models with an image, and ranking each model based on how similar it is to the input image. The results of this preliminary system are promising, however there remain many issues to be addressed. We now introduce these issues, which motivate the extensions presented in the following chapter.

The approach described in this chapter depends heavily on [Hedau et al., 2009]'s room layout estimation algorithm. Naturally, if this preliminary step in our pipeline fails, the scene matching stage will fail to find a correct and well-aligned furniture arrangement. Figure 2.7(middle) shows an example of a small error in room layout estimation, for which we can still produce reasonable results; however, there are many situations when room layout errors are substantial enough that we cannot produce a reasonable result. Figure 2.13a shows an example of an image for which estimating the room layout is very challenging. Note that the boundaries between the walls and the floor are almost entirely occluded. Moreover, there are strong gradients along the front edge of the bed which can be mistaken as a wall/floor boundary. This causes [Hedau et al., 2009]'s room layout estimation algorithm to drastically understimate the size of the room as shown in Figure 2.13b. By making hard decisions and committing to the top-ranking room layout we are unable to recover and fit a 3D model to the scene. This issue is precisely why there is a large gap in performance between using annotated room layouts and our fully automatic approach. In Chapter III, we will present a mechanism for viewpoint selection, which does not commit to a single room layout hypothesis. Rather, we will explore many possible viewpoint hypotheses and automatically select the one which enables the best 3D model matching.

Another fundamental limitation of our proof-of-concept scene matching algorithm is that we aim to find a 3D model which has objects in the exact geometric configuration seen in each image. Although our library of models from 3D Warehouse contains a diverse set of object combinations and configurations, our search space is still quite limited. Figure 2.14 shows an example failure case highlighting this issue. Note that our scene matching algorithm is able find a 3D model with two beds and a nightstand; however, the size and position of these objects is imprecise. Thus, in the next chapter, we will present a geometry refinement algorithm which uses a matched 3D model as a starting point and performs an optimization to identify the exact position and styles of objects in the scene.

Even if the room layout estimation process succeeds, and our library of 3D models

(a) Input image (b) Estimated room layout

Figure 2.13: Example failure case due to incorrect room layout estimation. Note the walls are predicted to be too close to the camera, preventing us from properly aligning 3D models.

(a) Input image (b) Estimated object locations

Figure 2.14: Example failure case due to our limited search space. Note that that objects are correctly identified; however, their locations are not accurate resulting in an imprecise segmentation mask.

contains an excellent match for an image, our hypothesis ranker may not be able to identify which model is best if our features fail to accurately encode how similar each model is to the input image. The diversity of object appearances in images makes it difficult to robustly estimate their locations and geometry. For example, upholstered furniture can be virtually any color or texture, making it difficult to generalize. Moreover, the textures on foreground objects can often be found in the backgrounds of indoor scenes. Figure 2.15 shows three example scenes with unusual textures on furniture which also appear on rugs, wallpaper and curtains. These types of images present unique challenges which require robust similarity

Figure 2.15: Example bedroom images from the SUNs Database showing the tremendous diversity in object appearances. Each of these images contains furniture with textures and colors that also appear on the rugs, wallpaper and curtains, making it difficult to estimate the $p(\text{object})$ masks used in our similarity features.

features. The analysis in Appendix D shows that we are limited by our inability to properly rank each hypothesis using our current set of similarity features. This will become more of an issue as we begin to refine geometric hypotheses to for fine-grain alignemt. Thus, in the next chapter, we will present a set of new similarity features which are engineered to more robustly and precisely compare images with 3D models, and analyze how improved similarity features increase the performance of our approach.

CHAPTER III

3DNN: Robust 3D Model Matching

In Chapter II we presented a proof-of-concept method for matching images with 3D models to estimate the geometry of a scene. We experimentally showed this to be a powerful mechanism for inferring the structure of a scene from a single image. However, because this approach aims to match the exact configuration of objects in an image, with an identical furniture configuration from a library of 3D models, the algorithm does not have the flexibility required to precisely reconstruct the diversity of object configurations found in natural scenes.

Thus, rather than limiting ourselves to a fixed set of object configurations, we extend our approach to first begin with a 3D model which closely match an image, and then undergo a two-stage *geometry refinement* algorithm. The first stage of the geometry refinement process adjusts the locations of each object in the hypothesized geometry to produce a result which more precisely aligns with the input image. The second stage takes each object and searches through a library of 3D models to find objects of the same category which more closely matches the size, shape and style of the objects in the image.

This type of fine-grained geometry refinement is challenging, and requires a set of features which are sufficiently discriminative to identify when rendered objects are precisely aligned in the image plane. Thus, we present a *new set of features* which improve the overall accuracy of our scene matching algorithm, enabling this geometry refinement stage.

In addition, we introduce a *viewpoint selection* process which does not commit to a single viewpoint estimate. We consider many camera pose hypotheses and use a learned cost function to select the camera parameters which enable the best scene geometry match. As the number of 3D hypotheses we consider grows, so does the computational complexity. Thus, in Section 3.6 we present algorithms which aim to more efficiently explore the search space, to remain tractable as we consider orders of magnitude more hypotheses.

In this chapter, we describe each of these components as part of the 3DNN (3D Nearest-Neighbor) algorithm, and analyze how each stage contributes to the overall system performance. Lastly, we compare 3DNN with traditional 2D nearest-neighbor approaches, to demonstrate the benefits of viewpoint-invariant scene matching.

3.1 Similarity Features

Many vision researchers have emphasized and demonstrated the importance of high-quality features for various image matching tasks. For example, the SIFT descriptor of [Lowe, 1987] and [Oliva and Torralba, 2006]'s GIST descriptor been shown to outperform many other features for common vision tasks such as object recognition and scene categorization [Douze et al., 2009, Mikolajczyk and Schmid, 2005]. For monocular geometry estimation, there has been less attention on developing and evaluating new features. In fact, the majority of recent algorithms addressing these problems have used the same features: [Lee et al., 2009]'s Orientation Maps and [Hedau et al., 2009]'s Indoor Geometric Context. Given the novelty of 3D scene matching approaches, there still remains substantial room for improvement via feature engineering. Therefore, we develop a series of features which are specifically designed to achieve our goal of precisely detecting objects and delineating their boundaries. We now present three new similarity features, to augment the initial set of features proposed in Chapter 2.2.

Object Locations:

To accurately predict the locations of objects in an image, we train a probabilistic classifier using the algorithm of [Munoz et al., 2010].[1] For each pixel, we estimate the likelihood of an object being present. This $p(\text{object})$ descriptor is compared to hypothesized object locations via rendering to compute a similarity feature indicating how well hypothesized objects align with predicted object locations. This similarity feature is akin to our use of Indoor Geometric Context [Hedau et al., 2009, Hoiem et al., 2007a] in our preliminary scene-matching prototype; however, it is more robust to the diversity of object colors, textures and illumination conditions seen in the SUN database [Xiao et al., 2010].

Figure 3.1 shows an example of a relatively simple scene for which [Hedau et al., 2009]'s Indoor Geometric Context is unable to accurately estimate the locations of objects; however,

[1]Training was performed using 10-fold cross validation on a subset of the SUN Database [Xiao et al., 2010], for which there exist LabelMe annotations [Torralba et al., 2010].

(a) input image (b) original $p(\text{object})$ (c) original scene match (d) new $p(\text{object})$ (e) new scene match

Figure 3.1: Effects of incorrect object location estimation on scene matching. Note in (b) that only a small region near the headboard of the bed is predicted to belong to an object using our preliminary $p(\text{object})$ feature, resulting in a poor scene match (c). However, as seen in (d), our new $p(\text{object})$ feature can more accurately predict the locations of objects, enabling a good sceen match (e).

our new approach succeeds. In these $p(\text{object})$ visualizations, black regions indicate a high likelihood that the pixels correspond to foreground objects, and white pixels are predicted background (walls or floor) locations. Note that the failure to correctly identify which pixels belong to objects (Figure 3.12q) results in a poor 3D model match (Figure 3.12r), which is consistent with the $p(\text{object})$ prediction. However the classifier of [Munoz et al., 2010] correctly identifies the object locations, enabling better scene matching (Figure 3.12t). We incorporate this new $p(\text{object})$ feature into the existing set of similarity features from Chapter 2.2.

Oriented Edges:

In addition, we design another similarity feature that aims to find 3D models which, when projected onto the image plane, produce edges which closely align with edges in the input image. For each hypothesized 3D model, we first analyze its surface normals to identify edges (which we define as discontinuities greater than 20°). We compare the projection of these edges onto the image plane, with edges extracted from an input image using the globalPb algorithm [Arbelaez et al., 2011]. We use an oriented chamfer distance, which matches only edges which are within 30° of each other. This reduces the effects of spurious edges which are spatially close, but not properly oriented in the image. To efficiently compute the oriented chamfer distance, we discretize edges into 12 overlapping bins of 30°

covering the half-circle. This is similar to the directional chamfer matching approach introduced by [Liu et al., 2010], with the addition of overlapping bins to alleviate the effects of quantization artifacts at the boundaries between buckets. We use the same edge penalty truncation approach described in Chapter 2.2 to reduce the influence of outlier edges, resulting in a four-dimensional similarity feature (corresponding to different thresholds).

Surface Normals:

Our third new similarity feature aims to correctly predict the surface normals of patches in an input image, and find 3D models which agree with the predicted normals. We use the Data-Driven 3D Primitive algorithm of [Fouhey et al., 2013], to predict surface normals by detecting patches of the scene that provide information about the underlying surface normals (e.g., corners of rooms, textures in carpets). These patches are found via a collection of primitives consisting of a detector to find the primitive in a new image, a canonical form that represents the underlying surface normals of the primitive, as well as the particular patches from which the canonical form and detector are created. At training time, the collection of primitives is learned from images with color and surface normal information via an iterative approach that alternates between training the detector, finding new instances of the primitive, and updating the canonical form. At test time, the detectors of the primitives are applied to the new image, producing likely locations of each primitive. These detections can yield a sparse interpretation of the scene by transferring the primitive's canonical form to every detection site above a threshold and averaging the results. A dense interpretation can also be obtained by transferring not only the canonical form, but also the regions around the primitive in the training data.

We compare the predicted surface normal orientations to rendered 3D model orientations using the same approach presented in Chapter 2.2 for our similarity feature using [Lee et al., 2009]'s Orientation Maps. We use the dense interpretations from [Fouhey et al., 2013]'s algorithm, as well as the sparse surface normal estimates at eight different levels of sparsity. Each of these nine surface normal estimates are compared to 3D model surface normal renderings to produce a nine-dimensional similarity feature. See Figure 3.2 for a comparison of surface normals estimated using [Lee et al., 2009]'s Orientation Maps, and [Fouhey et al., 2013]'s 3D primivives.

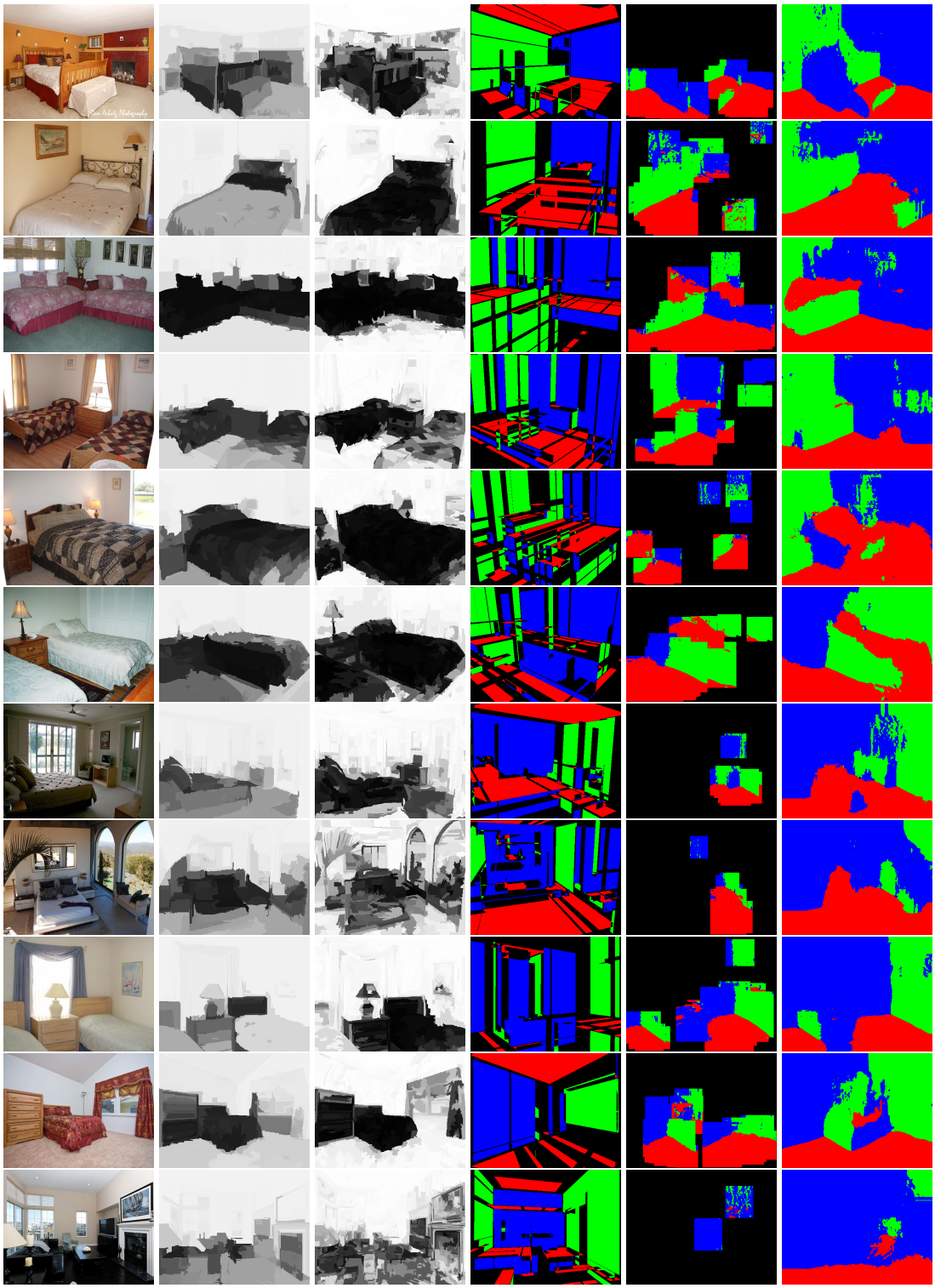


Figure 3.2: Example features. From left to right: input images, $p(\text{object})$ computed using [Hedau et al., 2009]'s Indoor Geometric Context, $p(\text{object})$ computed using [Munoz et al., 2010]'s classifier, surface normals computed using [Lee et al., 2009]'s Orientation Maps, sparse and dense surface normals computed using [Fouhey et al., 2013]'s 3D Primitives.

3.2 Incorporation of New Similarity Features

These new similarity features are combined with our seven original features from Chapter 2.2 to produce an 18-dimensional similarity feature vector. We retrain a support vector ranker [Herbrich et al., 1999] using the approach described in Chapter 2.3 to learn weights for the augmented set of features. Since the newly augmented feature vector now has more than twice the dimensionality, we first perform feature selection by incorporating an ℓ_1 penalty term, enforcing sparseness:

$$\min_{w,\xi} \frac{\lambda}{2}\|w\|_1 + \sum \xi^i_{jk} \text{ s.t.: } w^\top x^i_j \geq w^\top x^i_k + \delta^i_{jk} - \xi^i_{jk},\ \xi^i_{jk} \geq 0. \tag{3.1}$$

Features with negligible weights (less than 1%) relative to the average weight are discarded, and the selected features are re-weighted using the ℓ_2 regularized SVM ranking described in Equation 2.8 from Chapter 2.3.

3.3 Feature Analysis

We perform two experiments to analyze the importance of each feature for 3D model matching. First, we run a standard ablative analysis to see how much each feature contributes to the overall performance of our system. Next, we run our scene matching pipeline using only a single feature (or type of feature), and compare the performance of each feature independently with the performance of the full feature set.

For each of these experiments, we report performance using the two "Pixelwise Surface Normal Accuracy" metrics from Chapter 2.6.1, one measuring how accurately the surface normals of all pixels are predicted, the second evaluating only those pixels which correspond to objects in the ground-truth annotations. Although these metrics are informative for the task of surface normal prediction, they are unable to capture how accurately objects in an image are localized. For example, a horizontal surface corresponding to a bed in an image may be scored as "correct" even if the predicted scene contains no objects. This is because the horizontal floor has the same orientation as the bed's surface. Thus, we present results computed using a new metric, "Matched Objects Surface Normal Accuracy." This is a strict metric which requires two criteria to be met: For each pixel corresponding to objects in the ground-truth annotation, we must first correctly predict that there is an object at that location. We compute the dot product between ground-truth and predicted surface normals only at those pixels for which we "match" an object. Unmatched object pixels receive a

score of zero. This metric is more sensitive to correctly predicting the exact locations and geometries of objects in a scene.

In [Hedau et al., 2012] and [Satkin et al., 2012b], the authors present various metrics for how accurately their algorithms can predict the 3D freespace of a scene. These metrics require rectifying the predicted scene geometry, and are ill-posed when the estimated viewpoint deviates substantially from the ground-truth camera parameters. For example, if the estimated horizon computed from vanishing points is incorrect and intersects the ground-truth floor polygon, the rectification homography (computed using the ground-truth camera parameters) will produce incoherent results with points at infinity being projected to finite locations when applied to the estimated scene geometry. Thus, we develop another new metric to measure freespace prediction in the image plane: "Floorplan Overlap Score." For each object in the scene, we render its "footprint" by setting the height of each polygon to 0. A simple pixel-wise overlap score (intersection/union) of the object footprints can now be used to compare the ground-truth floorplan of a scene with our estimated scene geometry.

Figure 3.3 shows the degradation in performance resulting from the removal of each feature (or groups of features). Note that the removal of edge features has surprisingly little effect on the overall system performance. This is because the edge features were designed for fine-grained matching and alignment, which is not emphasized in our evaluation metrics. Moreover, accurate $p(\text{object})$ masks and orientation estimates can implicitly encode edge information resulting from object boundaries and surface normal discontinuities, compensating for the removed edge features. Interestingly, the removal of either $p(\text{object})$ feature has only a small effect on system performance; howevever, removing both $p(\text{object})$ features results in a dramatic drop in performance. This implies that the two features encode much of the same information, and can compensate when the other is removed. On the contrary, the drop in performance by ablating both surface normal features is approximately additive, indicating that the two features are independent, offering complementary information for scene matching.

Figure 3.4 shows the performance of each feature in isolation, for various metrics. The baseline performance indicated on the left is included for comparison. These graphs provide insight into the performance of our newly engineered features. For example, the oriented edge matching provides a significant improvement over our original non-oriented chamfer matching approach. Interestingly, this analysis shows that there is one feature – [Munoz et al., 2010]'s $p(\text{object})$ masks which perform almost as well as the full feature set (greater than 95% relative performance). This may seem surprising at first; however, figure/ground

perceptual grouping is at the core of Gestalt principals for object recognition [Rubin, 1915], and numerous human psychophysical experiments [Cutzu and Tarr, 1997, Rosch et al., 1976] have shown silhouettes to be sufficient for human vision. Moreover, the computer vision community has demonstrated the capability of classifying objects (or groups of objects) based solely on silhouettes [Belongie et al., 2002, Latecki et al., 2000]. Thus, we should expect accurate $p(\text{object})$ descriptors to not only be necessary, but perhaps sufficient for the task of geometry estimation. See Section 3.7 for further analysis of the benefits of our new set of similarity features.

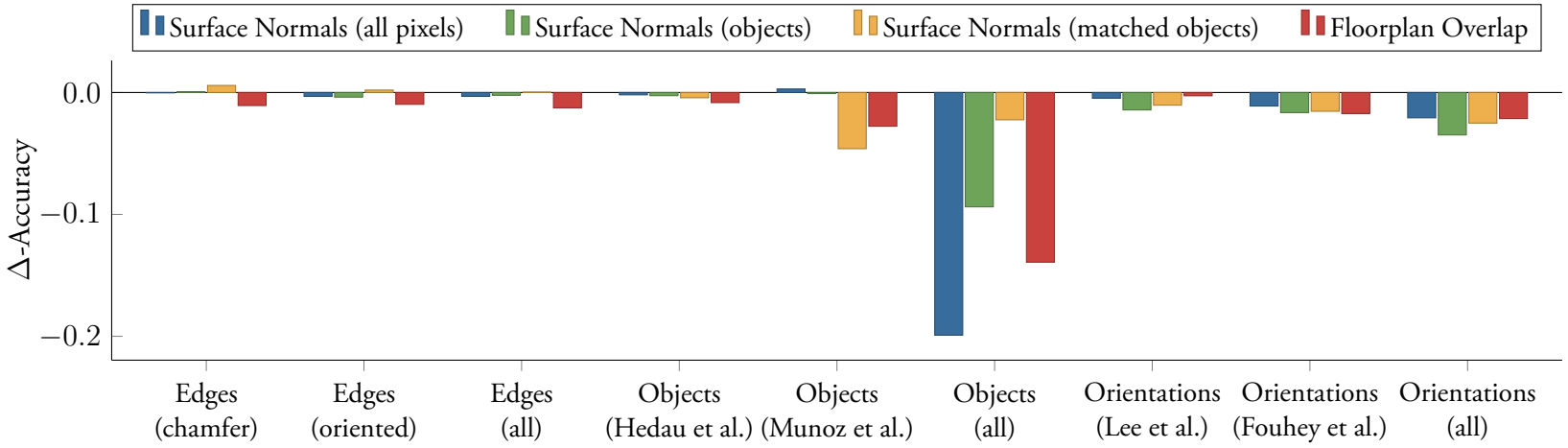


Figure 3.3: Change in performance resulting from the removal of individual features and groups of features.

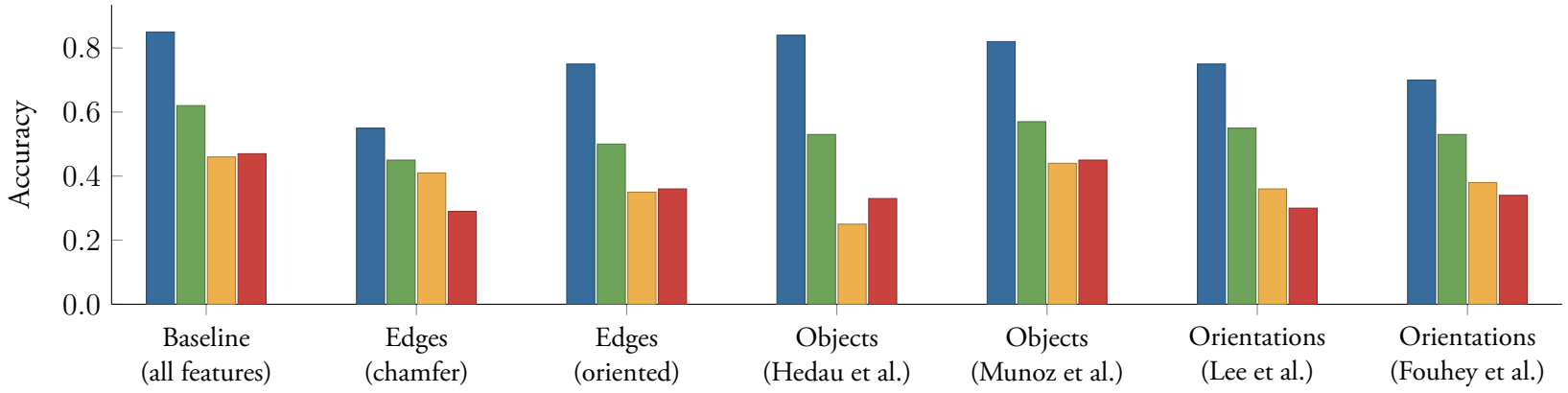


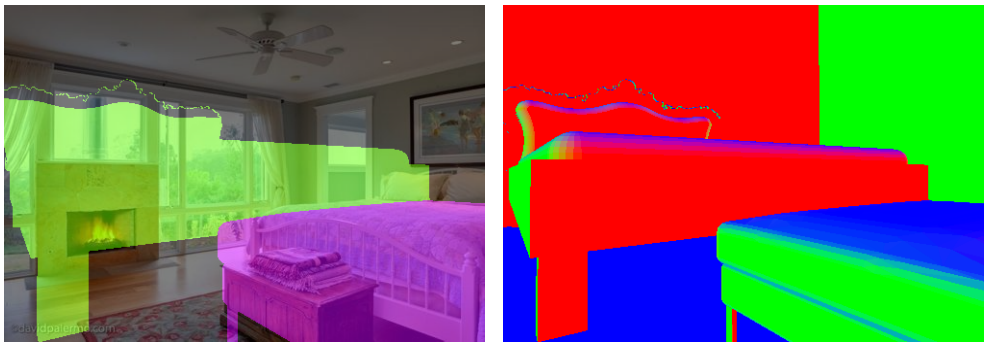
Figure 3.4: Performance of individual features, compared to the baseline (left).

3.4 Viewpoint Selection

The problem of viewpoint estimation is very challenging. Estimating the layout of a room, especially in situations where objects such as furniture occlude the boundaries between the walls and the floor remains unsolved. Recently, researchers such as [Hedau et al., 2010, Lee et al., 2010, Pero et al., 2012] proposed mechanisms for adjusting the estimated locations of walls and floors to ensure that objects (represented by cuboids) are fully contained within the boundaries of the scene. Inspired by these approaches, we aim to intelligently search over viewpoint hypotheses. Intuitively, if we can fit an object configuration using a particular viewpoint hypothesis with high confidence, then that room layout is likely correct (i.e., it allows for objects to be properly matched). By searching over possible viewpoints, we aim to alleviate the brittleness of our baseline scene matching approach which relied on hard decisions for the estimated viewpoint of an image. It should be noted that our geometry estimation algorithm is one of many recent works which rely on accurate viewpoint estimation [Fouhey et al., 2012, Gupta et al., 2011]. These types of geometry estimation algorithms are unable to recover when the room layout estimation process fails.

Thus, we present a framework which does not assume any individual viewpoint hypothesis is correct. Rather, we use our learned cost function to re-rank a set of room layout hypotheses by jointly selecting a combination of furniture and camera parameters, which together best match the image. We search over the top N room layout hypotheses, returned by the algorithm of [Hedau et al., 2009]. For each individual room layout, we use the estimated camera parameters corresponding to that room layout to render every 3D model from. This approach scales linearly with the number of viewpoint hypotheses explored, and is trivially parallelizable. In all our experiments, we consider the top 20 results from [Hedau et al., 2009]'s room layout algorithm. However, our approach is agnostic to the source of these viewpoint hypotheses, and additional hypotheses from [Lee et al., 2009, Pero et al., 2011, Schwing and Urtasun, 2012] or any other algorithm could easily be incorporated to improve robustness.

Figure 3.5 illustrates the benefit of searching over various camera parameters. The top row shows the result of 3DNN using only the top-ranking room layout from [Hedau et al., 2009]. Note that the failure to accurately estimate the height of the camera causes inserted objects to be incorrectly scaled. However, by not limiting ourselves to a single camera parameter hypothesis, we can automatically select a better room layout estimate, enabling a higher-scoring geometry match to be found. Figure 3.5b uses the 10^{th}-ranking hypothe-



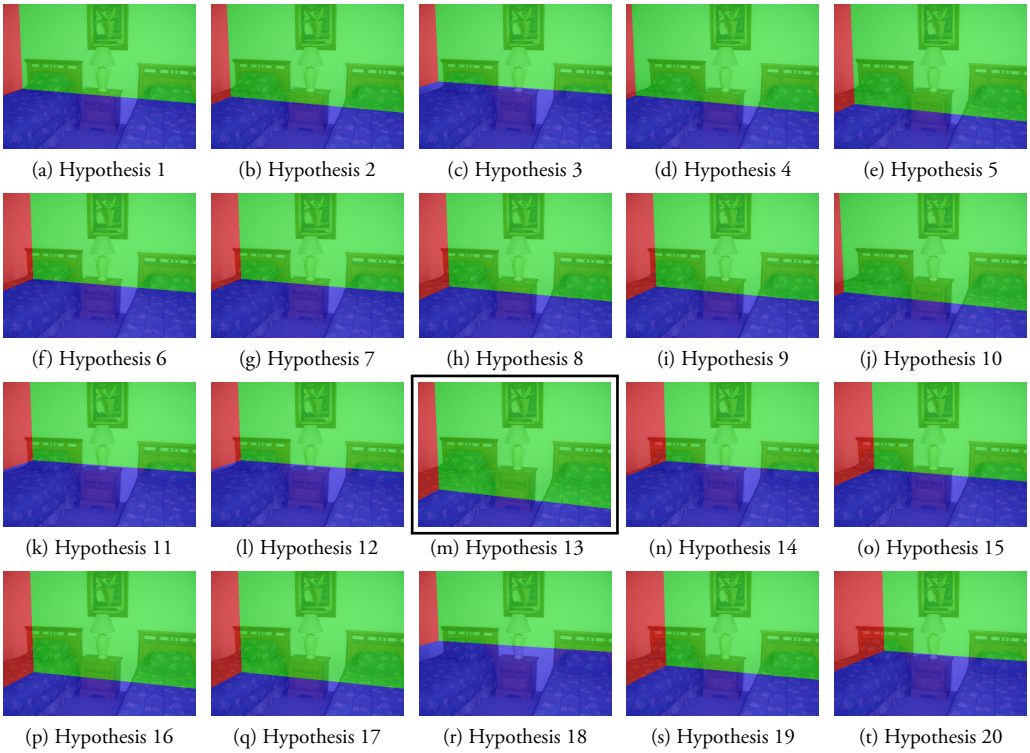
(a) Result using only the top-ranking camera parameters from [Hedau et al., 2009].

(b) Result after re-ranking the top-20 hypotheses from [Hedau et al., 2009].

Figure 3.5: Example results highlighting the benefit of searching over viewpoint hypotheses. The top row shows the best matching scene geometry using the top-ranking room layout hypothesis of [Hedau et al., 2009] (note the incorrect camera height estimate, causing objects to be rendered at the wrong scale). The bottom row shows the best matching scene geometry after intelligently selecting the best room layout.

sis from [Hedau et al., 2009], and has the highest matching score using our learned cost function.

Figures 3.6 and 3.7 show the full set of hypotheses considered for a scene during our viewpoint selection process. Note that the beds and nightstands almost fully occlude the wall/floor boundaries in the image, resulting in an inaccurate room layout estimate (see Figure 3.6a). As shown in Figure 3.7a, our baseline scene matching algorithm does its best to find a 3D model which when rendered from this inaccurate viewpoint aligns with the input image; however, the result is incorrect. Looking through the best matching scene geometries for each viewpoint hypothesis, we see that the more accurate a viewpoint estimate is, the more precisely we can find a matching scene geometry which aligns with the input image. By independently scoring and ranking each of these hypotheses, we correctly identify Hypothesis 13 (Figures 3.6m and 3.7m) as the best viewpoint for this scene.



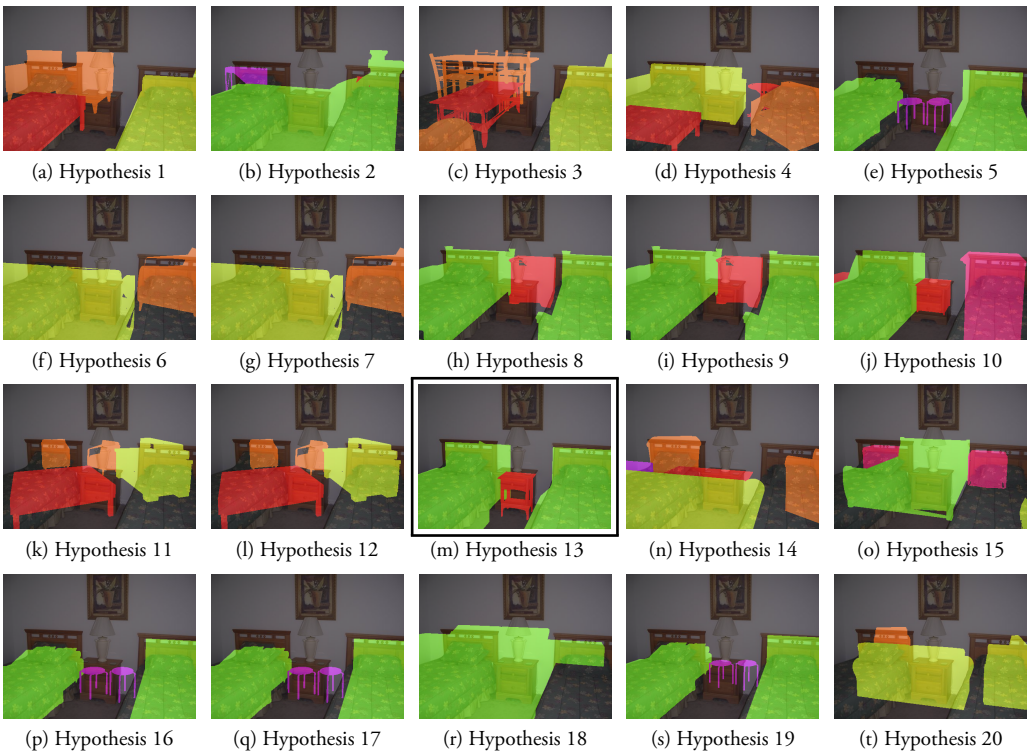
(a) Hypothesis 1 (b) Hypothesis 2 (c) Hypothesis 3 (d) Hypothesis 4 (e) Hypothesis 5

(f) Hypothesis 6 (g) Hypothesis 7 (h) Hypothesis 8 (i) Hypothesis 9 (j) Hypothesis 10

(k) Hypothesis 11 (l) Hypothesis 12 (m) Hypothesis 13 (n) Hypothesis 14 (o) Hypothesis 15

(p) Hypothesis 16 (q) Hypothesis 17 (r) Hypothesis 18 (s) Hypothesis 19 (t) Hypothesis 20

Figure 3.6: Top 20 viewpoint hypotheses from [Hedau et al., 2009] for an input image.



(a) Hypothesis 1 (b) Hypothesis 2 (c) Hypothesis 3 (d) Hypothesis 4 (e) Hypothesis 5

(f) Hypothesis 6 (g) Hypothesis 7 (h) Hypothesis 8 (i) Hypothesis 9 (j) Hypothesis 10

(k) Hypothesis 11 (l) Hypothesis 12 (m) Hypothesis 13 (n) Hypothesis 14 (o) Hypothesis 15

(p) Hypothesis 16 (q) Hypothesis 17 (r) Hypothesis 18 (s) Hypothesis 19 (t) Hypothesis 20

Figure 3.7: Object overlays for the best matching scene geometries given each viewpoint hypothesis.

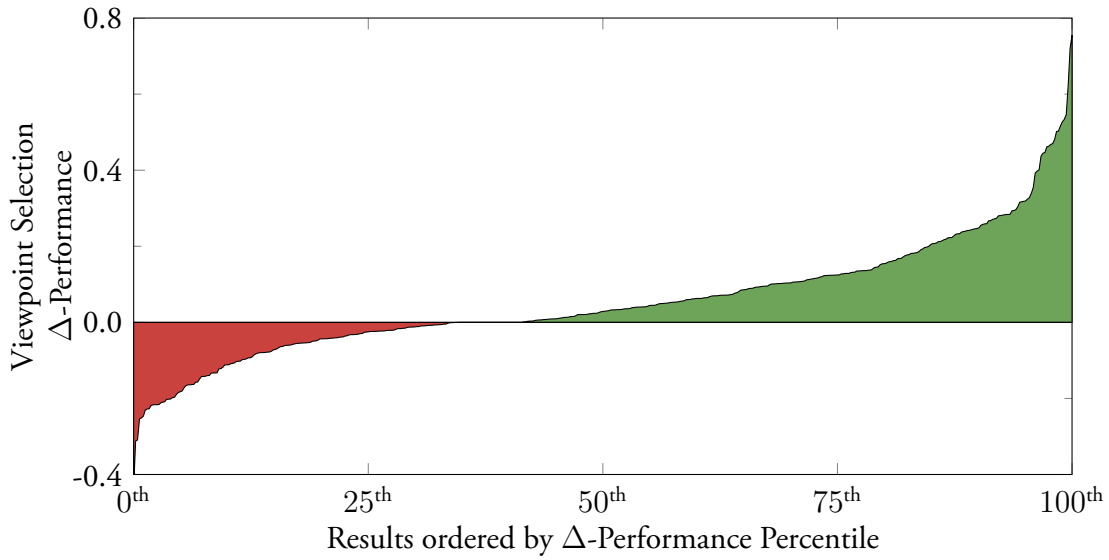


Figure 3.8: Distribution of improvements resulting from the viewpoint selection process. Performance is measured using the matched object surface normal scores. Green indicates a performance increase and examples in red resulted in a marginal performance decrease.

We quantify the benefits of viewpoint selection by comparing the accuracy of our results with and without viewpoint selection. Figure 3.8 shows the distribution of performance gains seen across all images in the CMU 3D-Annotated Scene Database as a result of the viewpoint selection process. The y-axis indicates how much the matched object surface normal score was affected via viewpoint selection. Note that for approximately two-thirds of the images, the viewpoint selection process results in an improved scene geometry (indicated in green). Not only does viewpoint selection result in more accurate object geometries, it also improves the accuracy of room box estimation by re-ranking viewpoint hypotheses based on which room layout affords for the best 3D model matching (14.0% per-pixel error with viewpoint selection versus 16.4% error without viewpoint selection). Additional experiments summarizing the benefits of viewpoint selection are included in Section 3.7.

3.5 Geometry Refinement

In order to accurately segment objects in an image, and reason about their occlusions, we must precisely estimate their positions. However, a fundamental limitation of nearest-neighbor approaches is that their outputs are restricted to the space of object configurations seen in training data. This is a problem which has affected both 2D and 3D non-parametric methods. Recently, algorithms such as SIFT flow [Liu et al., 2008] have been developed to address this issue. The SIFT flow algorithm perturbs a matched image by warping the pixels to better align with the input image. However, because this approach warps pixels in the image plane, there is no longer a coherent 3D interpretation of the result. Thus, we propose a two-stage geometry refinement algorithm which is inherently 3D. Our method first searches for the best location of each object in 3D, such that the projection of these objects best align in the image plane, producing a more precise result. Next, we search through a library of 3D models and try to replace each object in the scene with objects that more precisely match the size and style of the objects in the image. We now describe each of these refinement techniques and demonstrate their effectiveness (both qualitatively and quantitatively).

It should be noted, that our algorithm is not the only work to address 3D geometry refinement. In [Hedau et al., 2012], the authors present a refinement approach which locally adjusts the position of cuboids to better match an image. Similarly, [Pero et al., 2011] perturb the locations of cuboids as diffusion moves of a Markov Chain Monte Carlo optimization, which also includes the addition and removal of cuboids to the scene. The authors have recently extended their algorithm to refine hand-crafted parametric models of objects [Pero et al., 2013]. Our approach differs from these works in our use of vast repositories of non-parametric models. This data allows us to not only detect the positions and sizes of objects, but also their precise styles. Moreover, by not allowing objects to be arbitrarily resized, we ensure that they maintain real-world dimensions. For example, although beds have many possible styles, they cannot be arbitrarily resized; they come in discrete sizes (queen, king, etc.). Resizing household objects may violate real-world distributions, and result in scene geometries which no longer afford for the human actions they were designed to enable. Thus, we sample over discrete object hypotheses from 3D Warehouse to refine scenes while maintaining object functionality.

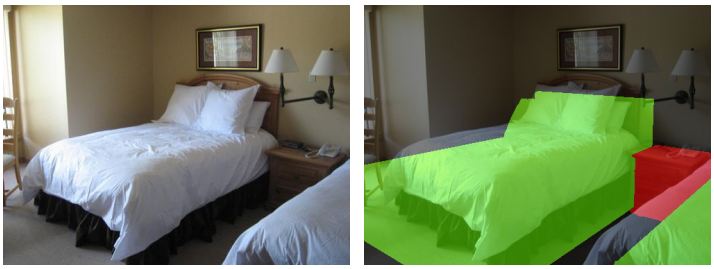
3.5.1 Object Location Refinement

We first search for local refinements of object locations which improve the overall geometric scene matching score. Algorithm 1 details this stochastic geometry refinement process. In each iteration of the refinement, the locations of objects on the x–y plane are perturbed (height off the ground remains fixed), by adding Gaussian noise (σ=1in) to the current objects' locations. If the adjusted objects' locations match the image better than the previous locations, the new locations are saved. This process repeats until convergence. In practice, a few hundred iterations are required to reach a final refined scene geometry.

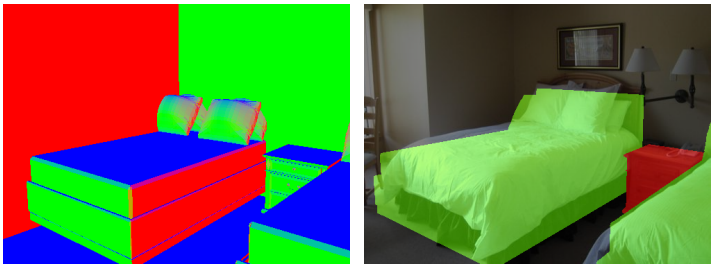
Figure 3.9 highlights the effects of our geometry refinement process. Note the initial object locations in 3.9b, when projected into the image plane do not align with the actual object boundaries. However, after refinement, in 3.9d the objects very precisely align with the image boundaries. The projected objects produce an excellent segmentation mask, and because the scene interpretation is inherently 3D, we can properly reason about occlusions and depth ordering.

Algorithm 1 Stochastic geometry refinement.

```
function REFINE(image, initialGeometry)
    G ← initialGeometry
    S ← SCORE(image, G)
    iteration ← 0
    while iteration < maxIterations do
        iteration ← iteration + 1
        G′ ← G
        for each object in G′ do
            object.x ← object.x + 𝒩(0, σ)
            object.y ← object.y + 𝒩(0, σ)
        end for
        if SCORE(image, G′) > S then
            G ← G′
            S ← SCORE(image, G′)
            iteration ← 0
        end if
    end while
    return G
end function
```



(a) Input image (b) Preliminary object locations



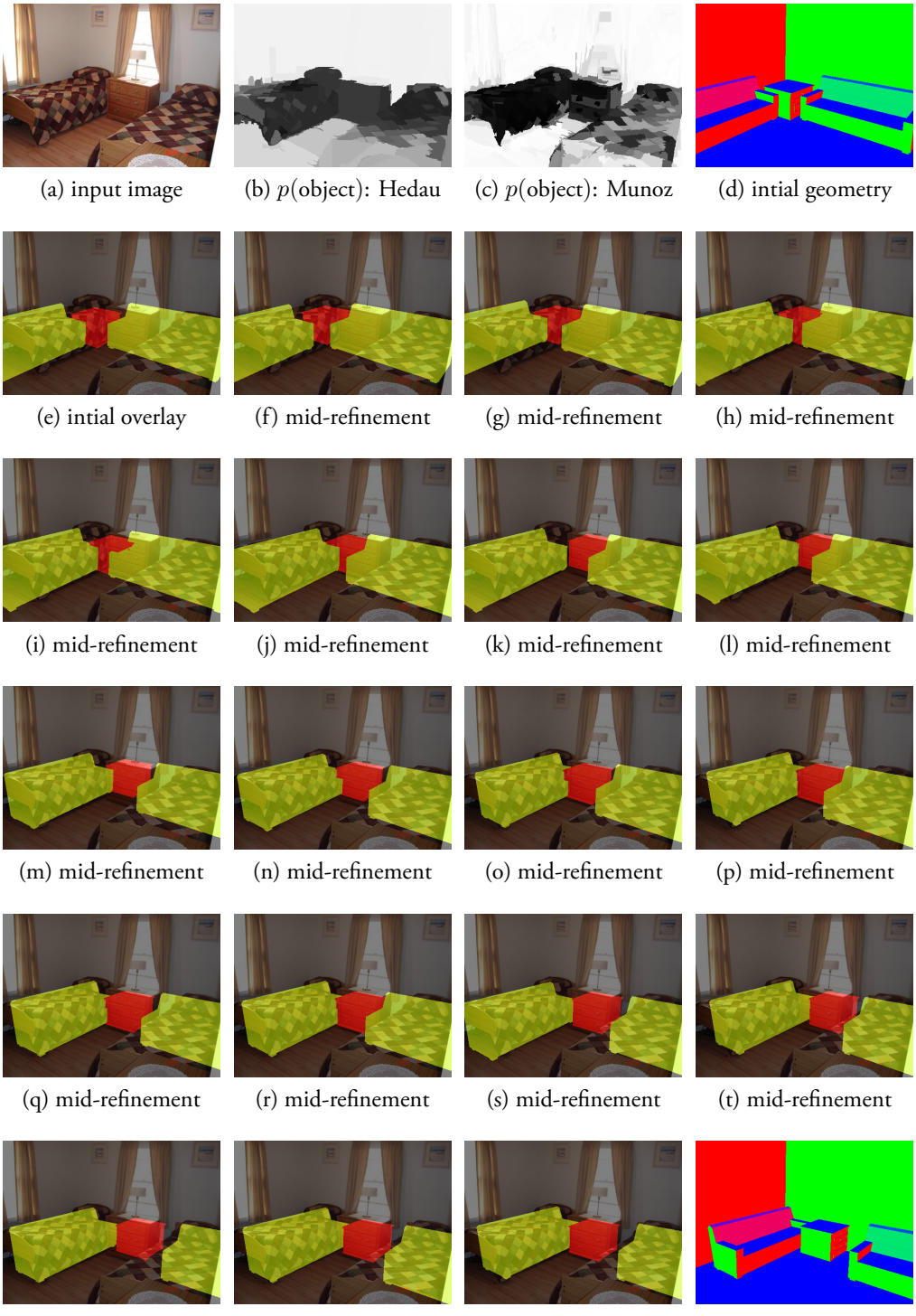
(c) Final refined geometry (d) Final refined object locations

Figure 3.9: Effects of the geometry refinement process. Note that object boundaries are well-delineated after refinement.

Figure 3.10 shows an interesting visualization of the object location refinement process for a scene with a poor initial geometry estimate. Note the incorrect $p(\text{object})$ masks in Figures 3.10b and 3.10c, resulting in the poor initial geometry match shown in Figure 3.10d. The location refinement algorithm slides these objects around until the couch on the left nicely aligns with the bed in the input image, and the end table is positioned where the nightstand appears in the image. This example demonstrates the capability of our refinement algorithm to adjust the locations of objects in 3D, such that their projections best align with the input image.

3.5.2 Object Swapping

The sizes, shapes and styles of objects found in real-world environments is quite diverse. Just as our initial scene matching approach is limited by the the space of object configurations seen in training data, after location refinement, the accuracy of our results are still limited by the set of object geometries found in each matched 3D model. For example, Figure 3.11b shows an initial scene geometry match for which the identities and locations of objects have been correctly predicted; however, the style of these objects are not accurately matched. The image contains a canopy-style bed with a tall metal frame, while our matched 3D model contains a more traditional bed with a rounded headboard and footboard.



(a) input image (b) p(object): Hedau (c) p(object): Munoz (d) intial geometry

(e) intial overlay (f) mid-refinement (g) mid-refinement (h) mid-refinement

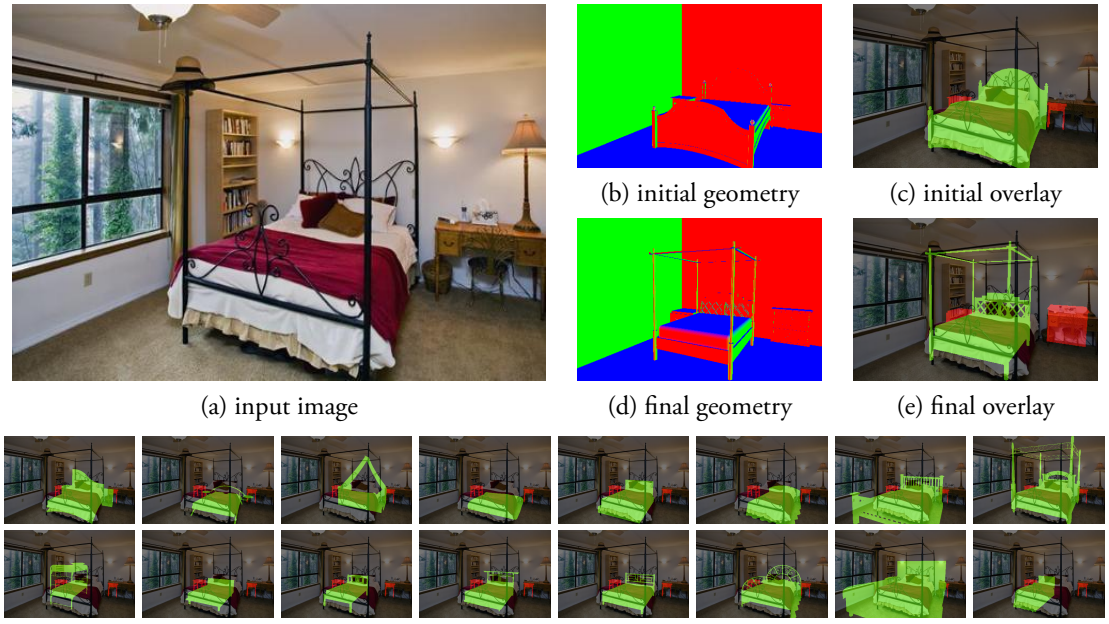
(i) mid-refinement (j) mid-refinement (k) mid-refinement (l) mid-refinement

(m) mid-refinement (n) mid-refinement (o) mid-refinement (p) mid-refinement

(q) mid-refinement (r) mid-refinement (s) mid-refinement (t) mid-refinement

(u) mid-refinement (v) mid-refinement (w) final overlay (x) final geometry

Figure 3.10: Visualization of the geometry refinement process for a scene with a poor initial geometry match. Note the incorrect p(object) masks in (b) and (c), resulting in the poor initial geometry match (d). Given this geometry, the location refinement algorithm slides these objects around until the couch on the left nicely aligns with the bed in the input image, and the end table is positioned where the nightstand appears in the input image.



(a) input image (b) initial geometry (c) initial overlay

(d) final geometry (e) final overlay

(f) Examples of additional beds considered for swapping.

Figure 3.11: Example effects of object swapping. Note that after swapping, the canopy style of the bed is correctly matched. Shown below are additional examples of the hundreds of beds considered during the swapping phase of the geometry refinement process.

The high level of diversity in object styles found in real-world scenes cannot be represented using parametric models. Thus, we leverage the vast collection of models in 3D Warehouse to search for objects which more precisely match the input image. Here, we aim to go from coarse-level object matches to instance-level matches. Our object swapping algorithm is simple and intuitive. For each object in a matched 3D model, we remove it from the scene and replace it with a new object from the same category. When replacing each object, we rotate and position the new models such that they best align with the original objects' positions. We use the same similarity features and learned cost function from Section 3.2 to score each object swapping hypothesis, and select the instance which maximizes this score. Hundreds of swapping hypotheses are considered for each object in the scene, and scored independently.

Figure 3.11d shows the resulting scene geometry after object swapping. Note that after object swapping, the unusual bed style is correctly matched. Figure 3.11f shows the diversity of hypotheses considered during the swapping process. By searching through hundreds of 3D models, we automatically select the instance which most precisely aligns with the input image. See Figure 3.12 for additional examples of object swapping. Note that after swapping, the style and size of each object is more precise than in the initial geometry estimate.

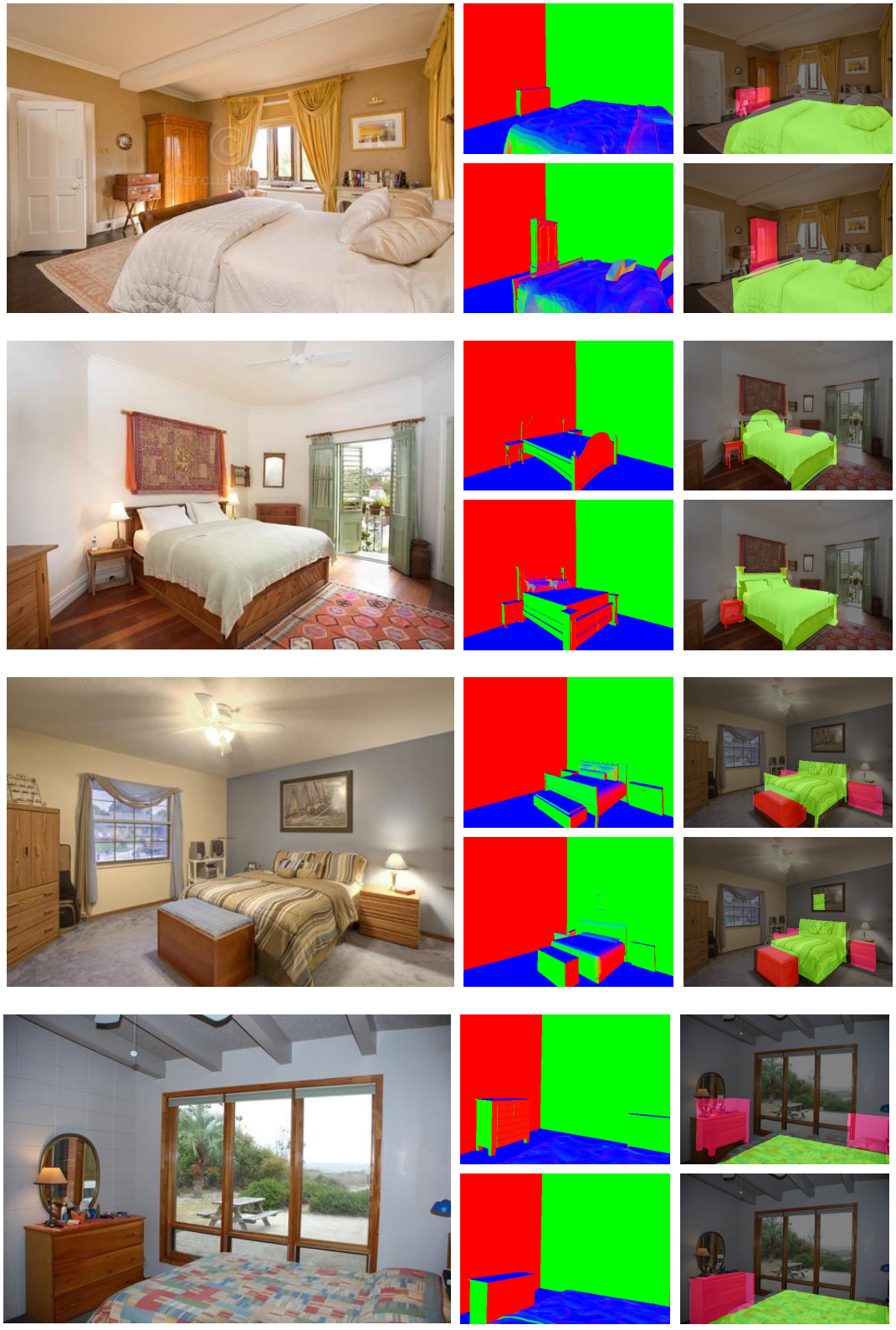


Figure 3.12: Example object swapping results. Besides each input image are the results before (top) and after (bottom) object swapping.

3.5.3 Refinement Evaluation

In Section 3.5.1, we presented an algorithm to refine the locations of objects in 3D, and in Section 3.5.2, we presented an algorithm to replace individual objects from a 3D model with objects which more closely match the input image. We now evaluate how each of these refinement methods affects the overall performance of our scene matching algorithm.

Figure 3.13 shows the distribution of performance gains seen across all images in the CMU 3D-Annotated Scene Database as a result of the geometry refinement process. The dashed line shows the effects of only moving objects and the dotted line shows the effects of only swapping objects. Quantitatively, these two refinement mechanisms perform similarly, resulting in performance gains on approximately two-thirds of the images. However, by combining the two refinement processes, further performance gains can be achieved, as indicated in the green region of the paired error plot.

Figure 3.14 shows bar graphs summarizing the performance of the refinement methods. Quantitatively, the geometry refinement stages of 3DNN result in modest improvements. This is expected, as the refinement process is inherently local and designed to make small modifications, not major changes which would result in dramatic effects on performance. However, qualitatively our results after refinement are markedly better, with objects being well-localized and their styles properly modeled. These effects are most pronounced in the matched object surface normal and floorplan overlap scores, for which precisely localizing and matching the style of objects is emphasized.

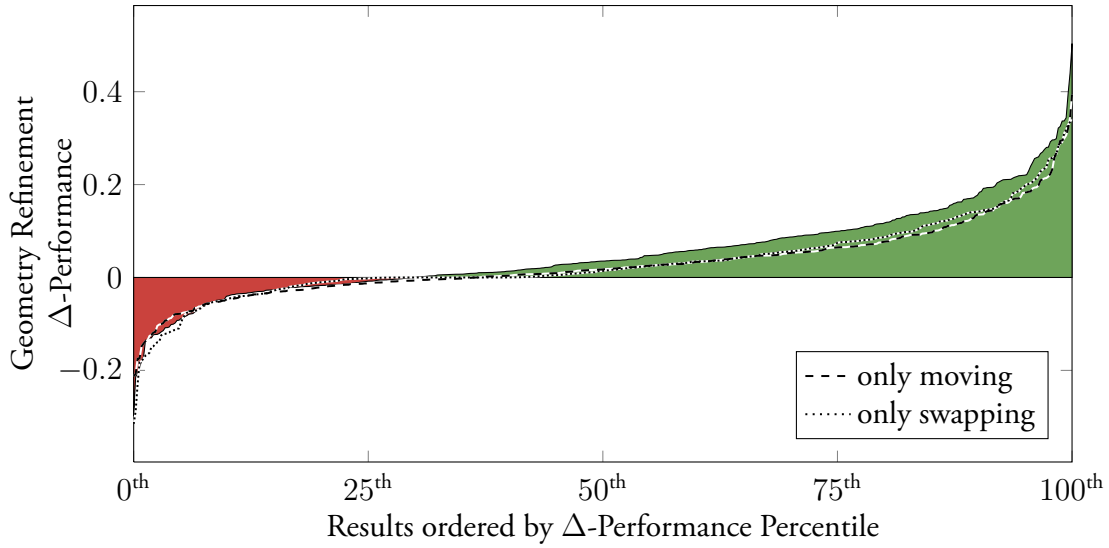


Figure 3.13: Distribution of improvements resulting from the geometry refinement processes. Performance is measured using the matched object surface normal scores. Green indicates a performance increase and examples in red resulted in a marginal performance decrease. The dashed and dotted lines indicate the performance of only moving or only swapping objects, respectively.

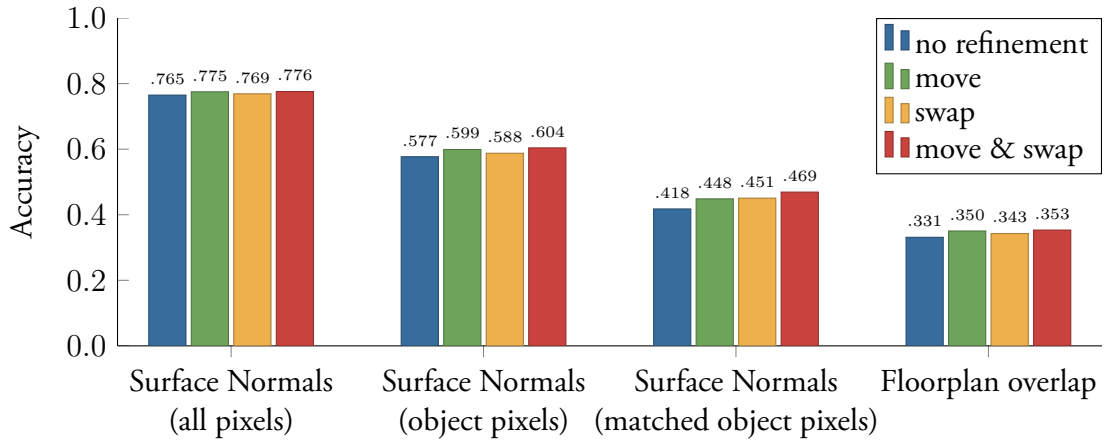


Figure 3.14: Analysis of each component of the geometry refinemnet process.

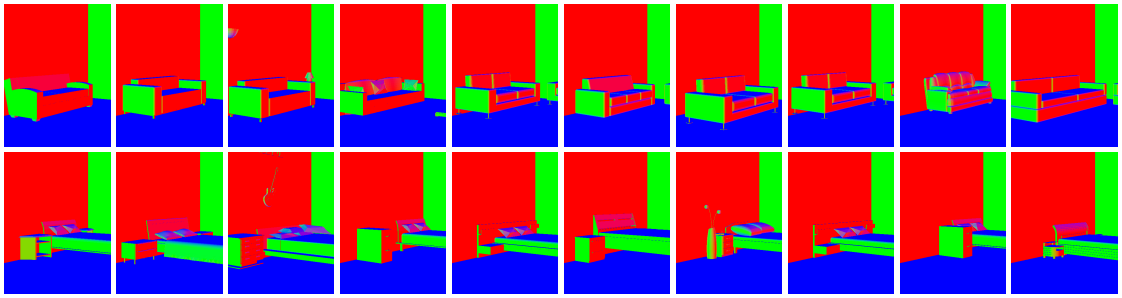


Figure 3.15: Example elements of two manually created "clusters" of 3D models. The top row includes scenes with similar couch geometries; the bottom row includes scenes with similar arrangements of beds and nightstands.

3.6 Efficiency Issues

Our baseline algorithm, presented in Chapter II, scales linearly with the number of scene hypotheses explored. The viewpoint selection and geometry refinement processes further exacerbate this issue by considerably increasing the search space. To remain tractable while exploring orders of magnitude more hypotheses we need an approach which does not naively explore all possible hypotheses in a brute-force manner.

The library of 3D models we downloaded from 3D Warehouse (described in Section 2.4) follows a classic long-tailed Zipfian distribution common to many data repositories of images and videos [Spain and Perona, 2011, Torralba et al., 2010] – there exists a tremendous amount of self-similarity in 3D Warehouse, which we can exploit to more efficiently match images with 3D models.

The number of unique 3D models is a fraction of the total number of models in our database. Figure 3.15 includes example elements of two manually created clusters of 3D models. The top row includes renderings of scenes with similar couch geometries; the bottom row includes scenes with similar arrangements of beds and nightstands. If an image were to match one of the models in a cluster, it will likely match all of the models in the cluster. Conversely, if an image does not match one of the models in a cluster, it likely will not match any models in the cluster. This property suggests ways to efficiently explore the library of 3D models by first sampling models at random from each cluster, and then mining through only those clusters which appear similar to the input image. In this section, we explore two algorithms for reducing the computational complexity of our scene matching approach by exploiting these properties.

3.6.1 Sequence Optimization

Our first algorithm aims to identify a diverse set of 3D models which are likely to perform well for *any* scene. We developed a data-driven method to mine through a set of 3D models, to select a subset which maximizes the performance on validation data. This optimization can be viewed as a submodular set covering problem. Here, each 3D model will perform well on a subset of images and poorly on others. Our goal is to select a subset of the 3D models, such that we maximize performance on all images, without redundancy. This problem is NP-hard, and there exists no polynomial time algorithm which can guarantee optimality, unless P = NP [Karp, 1972]. However, [Nemhauser and Wolsey, 1978], showed that a greedy algorithm will select a subset of a given cardinaliy, which is within $1 - 1/e$ (~63%) of optimal.

Our algorithm begins with an empty set, and iteratively adds the most beneficial 3D models. In each iteration, we select the 3D model $m \in M$ from our library which results in the largest improvement in performance on validation data, when added to the current subset S:

$$\arg\max_{m \in M} f(S \cup \{m\}) - f(S) \tag{3.2}$$

Here, $f(\cdot)$ represents the performance of a set of 3D models on validation data. This algorithm terminates after all 3D models in our library have been ordered by their marginal benefit. Note that this set is ordered, such that the best subset of size n is simply the first n elements of the set. This property enables our scene matching approach to run as an "anytime algorithm," which can report the best solution after any number of hypotheses have been explored.

This algorithm has $O(n^2)$ computational complexity; however, the optimization is only performed once as a precomputation stage before all scene matching is performed. For larger sets of hypotheses, there exists approximations such as [Dey et al., 2012]'s SeqOpt algorithm to more efficiently optimize the sequence of hypotheses explored.

Figure 3.16 shows the performance of our algorithm as a function of the number of hypotheses tested. The solid lines indicate the performace after n random 3D model hypotheses have been tested in a naive manner. The dotted lines indicate the performance of our anytime algorithm, evaluating n 3D models selected using the sequence optimization algorithm. Note that the performance after evaluating a fixed number of 3D models is higher after optimizing the sequence of tested hypotheses, allowing for comparable results to be generated in a fraction of the time.

3.6.2 Hypothesis Factorization and Score Prediction

Another approach to efficiently exploring the set of 3D models is to find correlations in the scores of scene hypotheses. These correlations can be used to derive a basis for the scores of all 3D models. Here, rather than computing the score for all hypotheses given an image, we sparsely sample a set of hypotheses, and use these scores to determine a linear combination of a set of given basis vectors which best reproduce the sampled hypothesis scores. Now, given these weights, we can predict the scores for the remaining untested hypotheses. This algorithm follows the successful "model recommendation" approach of [Matikainen et al., 2012] for efficiently selecting a model from a library of discriminative classifiers for action recognition. In their work, the authors perform a singular value matrix decomposition to determine the top principal components which can be used as bases for estimating the scores for a set of classifiers. The authors show that their method of score prediction for classifiers is quite robust and can actually outperform the brute-force approach of evaluating all classifiers, which is prone to overfitting.

Figure 3.17 shows the efficacy of the score prediction approach for our problem. Here, we first randomly sample 100 scene hypotheses and use the matching scores of these models to predict how well all remaining hypotheses will match the input image. Each additional hypothesis is then tested in order of its predicted matching score. This is a two-stage algorithm which first *explores* the search space with a random sampling of 3D model hypotheses, and then *exploits* knowledge aquired during the first stage to test only those models which are likely to match the input image. Thus, for the first 100 hypotheses tested, this algorithm performs on par with the naive approach; however, during the second stage, the algorithm intelligently selects hypotheses to evaluate, enabling excellent scene matches to be found after a few hundred iterations.

This algorithm optimizes a fundamentally different objective then the sequence optimization approach. The score prediction algorithm aims to find a set of 3D models which maximize the matching score on a specific image. On the contrary, the sequence optimization algorithm aims to find a fixed set of 3D models which *a priori* maximize the matching score for a diverse set of images. Thus, although the sequence optimization approach is able to find fairly good matches after a very small number of iterations, the algorithm is not designed for fine-grained matching and performs on par with the naive approach after many iterations. On the contrary, the score prediction algorithm learns which models are likely to match an image during the first 100 iterations, and then is able to quicky find the best matching model during the second stage of the algorithm.

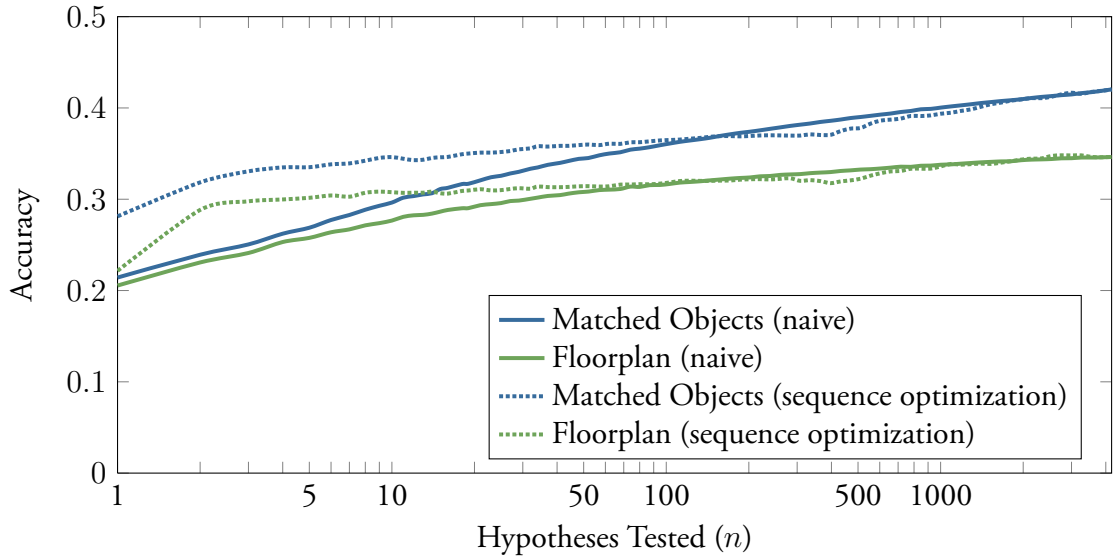


Figure 3.16: Accuracy as a function of the number of 3D model hypotheses tested. Solid lines indicate the baseline performance (random ordering), dotted lines indicate the performance of our sequence optimization approach. Note the logarithmic x-axis. Results are reported for two metrics, matched objects surface normal accuracy and floorplan overlap score.

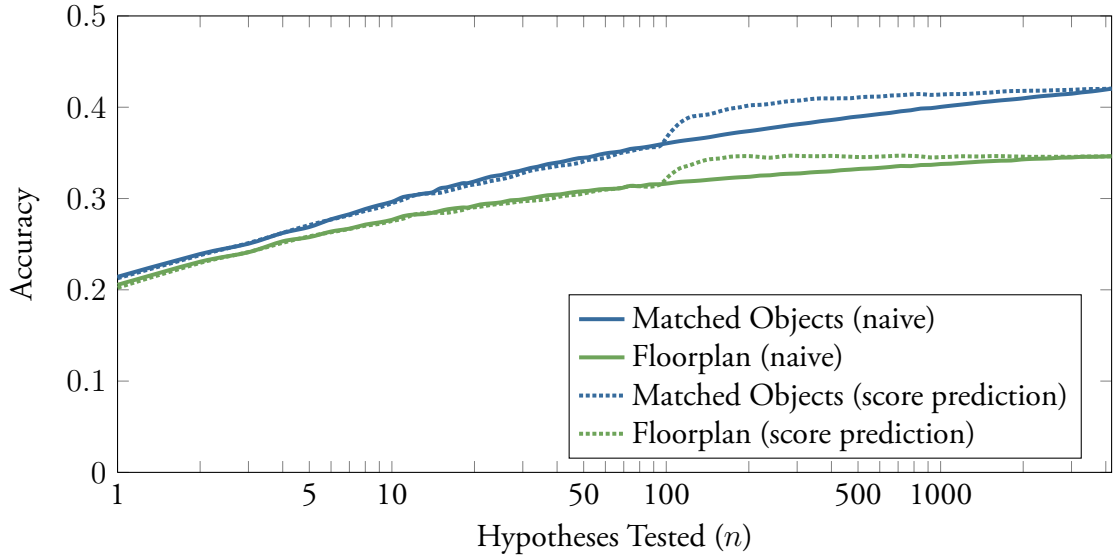


Figure 3.17: Accuracy as a function of the number of 3D model hypotheses tested. Solid lines indicate the baseline performance (random ordering), dotted lines indicate the performance of our sequence optimization approach. Note the logarithmic x-axis. Results are reported for two metrics, matched objects surface normal accuracy and floorplan overlap score.

3.7 Evaluation

We now evaluate the performance of our 3DNN algorithm. The goals of these experiments are two-fold. First, we analyze the added benefit of each component of the 3DNN system: improved similarity features, geometry refinement and viewpoint selection. In the next section, we compare 3DNN with state-of-the-art appearance-based scene matching aproaches, to demonstrate the benefits of viewpoint invariant scene matching.

We perform all experiments using the CMU 3D-Annotated Scene Database [Satkin et al., 2012a], containing 526 images of bedrooms and living rooms. All training was performed using 5-fold cross-validation to evaluate performance on all images in the dataset. Figure 3.18 reports the performance of our scene matching algorithm with each of the improvements proposed in this chapter. The blue bars indicate the baseline performance using the preliminary scene matching algorithm described in Chapter II. The next set of bars (green) indicate the added benefit of using the additional set of similarity features presented in Section 3.1. Note that the additional similarity features account for a large boost in performace. This effect is especially pronounced in the matched object surface normals and floorplan overlap scores, for which accurately predicting the locations of objects is crucial. The yellow bars show the performance by running our geometry refinement process (with the new feature set). The red bars show the performace gains achieved via viewpoint selection (with the new feature set). The purple bars indicate the performance achieved by running the full 3DNN algorithm with new features, viewpoint selection and geometry refinement. Lastly, for comparison, we report in gray the performance of our scene matching approach if we were to use annotated viewpoints (similar to the analysis in Chapter II, Figure 2.8). The gap in performance between the purple and gray bars represents the remaining error due to incorrect viewpoint estimation. Comparing the yellow, purple, and gray bars shows that our viewpoint selection mechanism accounts for a substantial performance gain; however, there still remains considerable room for improvement. This is most pronounced in the all pixel surface normal and floorplan overlap scores, which are sensitive to correctly estimating the locations of the walls and floors in a scene.

These extensions to our baseline scene matching algorithm have resulted in considerable improvements to both the robustness and precision of our results. Figures 3.19 and 3.20 include example results of 3DNN on a wide variety of bedroom and living room scenes. Note that we are able to produce accurate 3D models shown in the surface normal renderings beside each input image. In addition, each object's boundaries are well-delineated due to

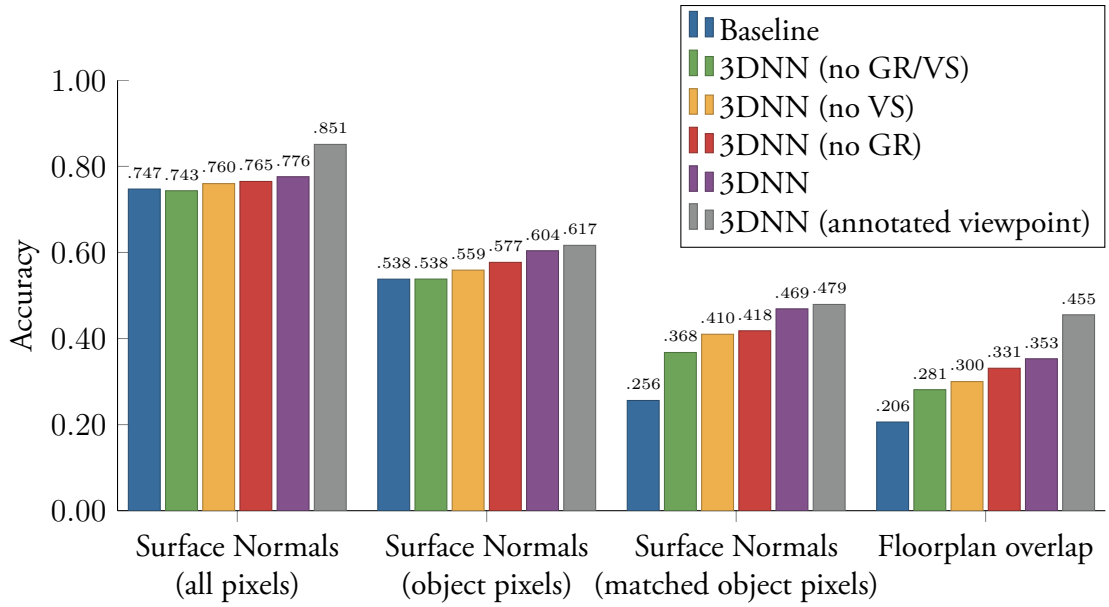


Figure 3.18: Analysis of the benefit of each component in our improved scene matching algorithm.

our geometry refinement stage, as indicated in the overlaid object segmentation masks.

Moving forward, our analysis indicates that continuing to develop new similarity features is likely to result in the most substantial improvements. Appendix D explores the effects of replacing our hypothesis ranker with an oracle capable of selecting the best 3D model for a given viewpoint of an image. These experiments decouple the errors due to our limited library of 3D models, incorrect viewpoint estimation, and poor hypothesis ranking, in essence computing an upper-bound on performance as if we had perfect viewpoint selection or hypothesis ranking. Our inability to robustly rank geometric hypotheses is a result of the challenges inherent in computing the similarity between an image and 3D models. The similarity features used for this are at the core of the 3DNN algorithm; they are required not only to select the 3D model which is most similar to an image, but also to rank viewpoint hypotheses and to determine the location and style of objects during geometry refinement. For example, although using annotated viewpoints (gray bars in Figure 3.18) improves performance relative to 3DNN with viewpoint selection (purple bars), this improvement is not because we do not consider the correct viewpoints; rather, our similarity features are unable to consistently discriminate between each viewpoint hypothesis. This does not come as a surprise, as room layout estimation, and the development of features for the task remain active areas of research within the vision community (e.g., [Ramalingam et al., 2013] and

[Hödlmoser and Micusik, 2013]).

John Lafferty puts it bluntly, "It's the features, stupid!" [Lafferty, 2002]. If our features cannot consistently encode our world in a discriminative manner, then no classifier, no matter how advanced can yield good results. Thus, we believe further development of new similarity features should be a priority for future research. Anything that can be estimated from an image and computed from 3D models can be compared as a similarity feature. For example, algorithms which detect Manhattan junctions [Sugihara, 1986], occlusion boundaries [Hoiem et al., 2007b] and depth gradients [Karsch et al., 2012] from single images could be incorporated into our approach by comparing their outputs to the surface normals and depth buffers of 3D models during rendering. Continued research and development of these types of similarity features can improve our ability to relate images with 3D models, pushing the boundaries of geometric scene understanding.

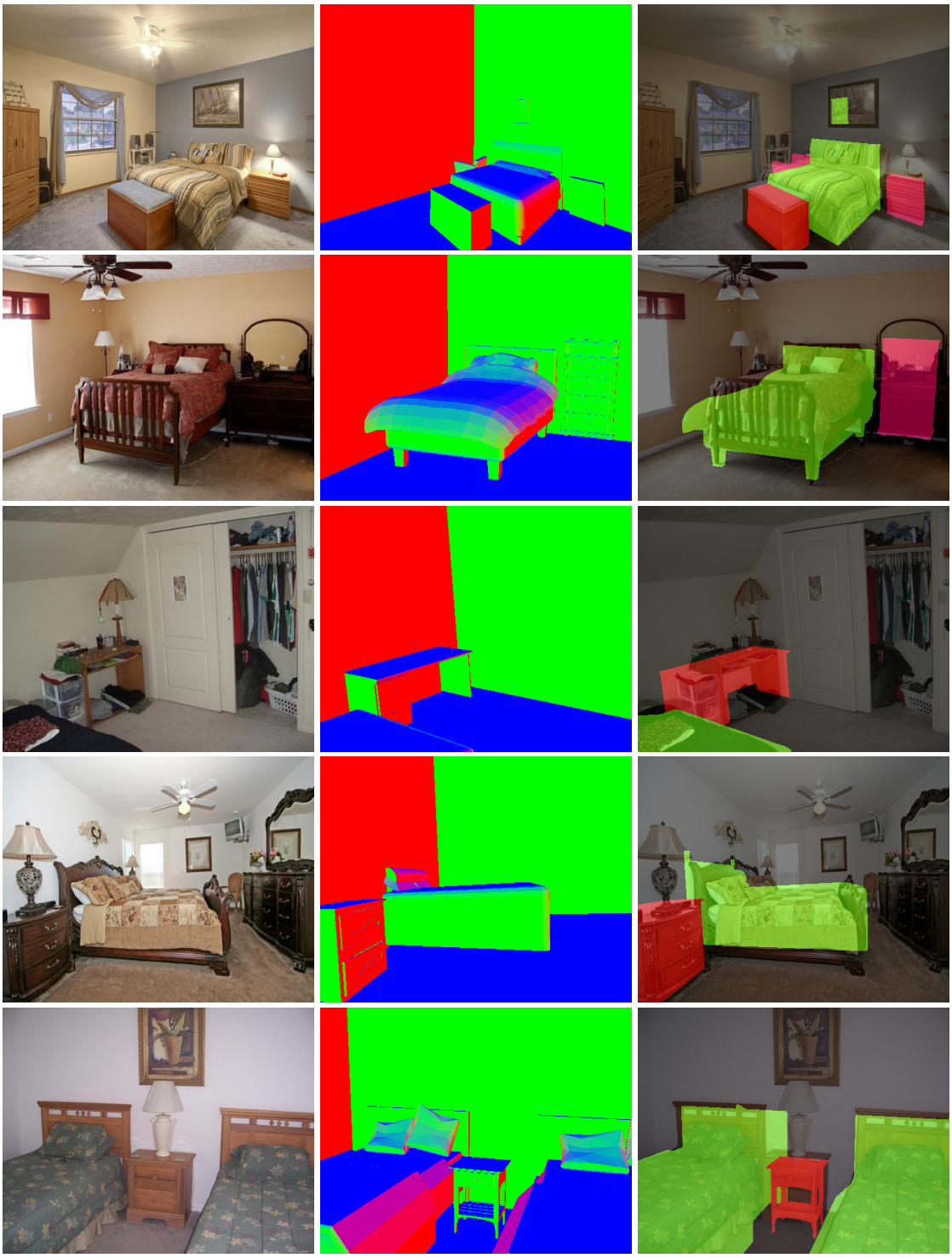


Figure 3.19: Qualitative results of bedroom scenes. From left to right: input images, surface normal renderings and overlaid object segmentation masks.

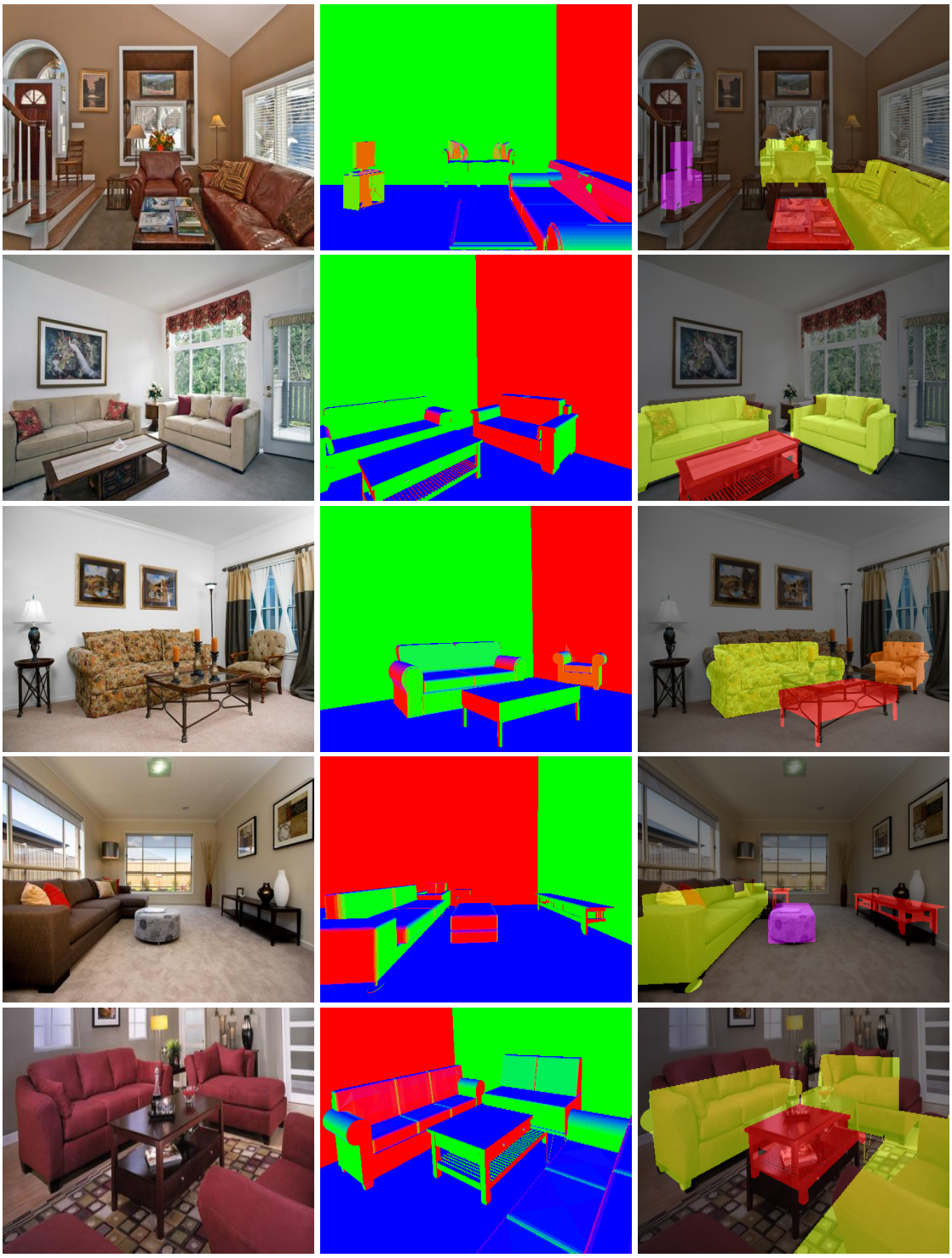


Figure 3.20: Qualitative results of living room scenes. From left to right: input images, surface normal renderings and overlaid object segmentation masks.

3.8 2D vs. 3D Nearest Neighbor

Data-driven scene matching is at the forefront of the computer vision field. Researchers have demonstrated the capability of simple nearest-neighbor based approaches to match an input image (or patches of an image) with a corpus of annotated images, to "transfer" information from one image to another. These non-parametric approaches have been shown to achieve amazing performance for a wide variety of complex computer vision and graphics tasks ranging from object detection [Tighe and Lazebnik, 2010] and scene categorization [Oliva and Torralba, 2006] to motion synthesis [Liu et al., 2008] and even image localization [Hays and Efros, 2008].

Although these 2D nearest-neighbor approaches are powerful, a fundamental limitation of these techniques is the need for vast amounts of data. For a traditional image matching approach to succeed, there must be an image in the recall corpus which is very similar to the input image (i.e., captured from a similar viewpoint, lighting conditions, etc.). This has propelled the growth of datasets, which now measure in the millions of images [Hays and Efros, 2007, Torralba et al., 2008]. Moreover, despite these massive datasets, 2D nearest-neighbor approaches cannot generalize to never-before-scene viewpoints.

Consider the pair of scenes in Figure 3.21. Note the images were captured from drastically different viewpoints. A traditional appearance-based image matching approach such as [Liu et al., 2008, Oliva and Torralba, 2006] would fail to generalize across such extreme viewpoint differences. Although the scenes appear quite different from the viewpoints they were captured, they have a lot in common: both scenes contain a couch facing a fireplace at approximately the same distance from each other. In this section, we show that we are able to automatically match these images by comparing the appearance of one image with the geometry of another. By decoupling the viewpoint and the geometry of an image, we develop a scene matching approach which is truly 100% viewpoint invariant.

3.8.1 Sources of 2D/3D Data

In Chapter II, we demonstrated our ability to match images with 3D models from large online repositories. In this section, we utilize models from the CMU 3D-Annotated Scene Database for matching. This new source of data has an image associated with each 3D model, enabling us to relate pairs of images via their underying geometry. Moreover, because each 3D model has an associated image (unlike data from 3D Warehouse), we are able to transfer any metadata from the source image when parsing an input image, as shown in Figure 3.21.

(a) Input image (b) Matched scene

(c) Automatically transferred object labels

Figure 3.21: Extreme viewpoint differences. Traditional appearance based image matching approaches fail to generalize across such extreme viewpoint differences; however, our approach is able to match the geometry of these two scenes, and transfer object labels.

At the core of our approach is the utilization of datasets of images with corresponding 3D descriptions. This immediately raises natural questions as to where this data is coming from, and whether or not there exists sufficient quantities of images with corresponding 3D models. In fact, the development of such 3D content is exploding, in large part due to the availability of low-cost RGBD cameras, which has been a catalyst for the rapid increase in 2.5D data. Researchers are now working on automated methods for inferring the full 3D geometry of a scene given a 2.5D projection [Shao et al., 2012, Silberman et al., 2012]. As these approaches become more effective, there will be massive amounts of images with associated 3D models, allowing for the first time the exciting possibilities afforded by using the full power of geometric information in conjunction with conventional appearance-based techniques. Our work shows how these emerging new sources of data can be used by quan-

tifying their effectiveness in terms of matching efficiency (dataset size), generalization to unseen viewpoints, geometry estimation, and object segmentation.

3.8.2 Geometry Estimation

We now quantify the performance of 3DNN with a variety of baseline scene matching approaches, including state-of-the-art 2D nearest-neighbor approaches. We compare 3DNN with our baseline scene matching approach from Chapter II as well as two popular 2D nearest-neighbor approaches: GIST [Oliva and Torralba, 2006] and HoG [Dalal and Triggs, 2005] matching.[2] Figure 3.22, reports the results for 3DNN compared to each baseline, for the task of geometry estimation. Note that our baseline 3D scene matching algorithm (indicated in yellow) does not offer substantial improvement with the 2D nearest-neighbor approaches on the more challenging metrics (matched object surface normals and floorplan overlap score); however, 3DNN exhibits dramatic improvement on each of these metrics.

[2]GIST: 4×4 blocks, 8 orientations (code from [Oliva and Torralba, 2006]). HoG: 20×20 blocks, 8 orientations (code from [Vondrick et al., 2013]).

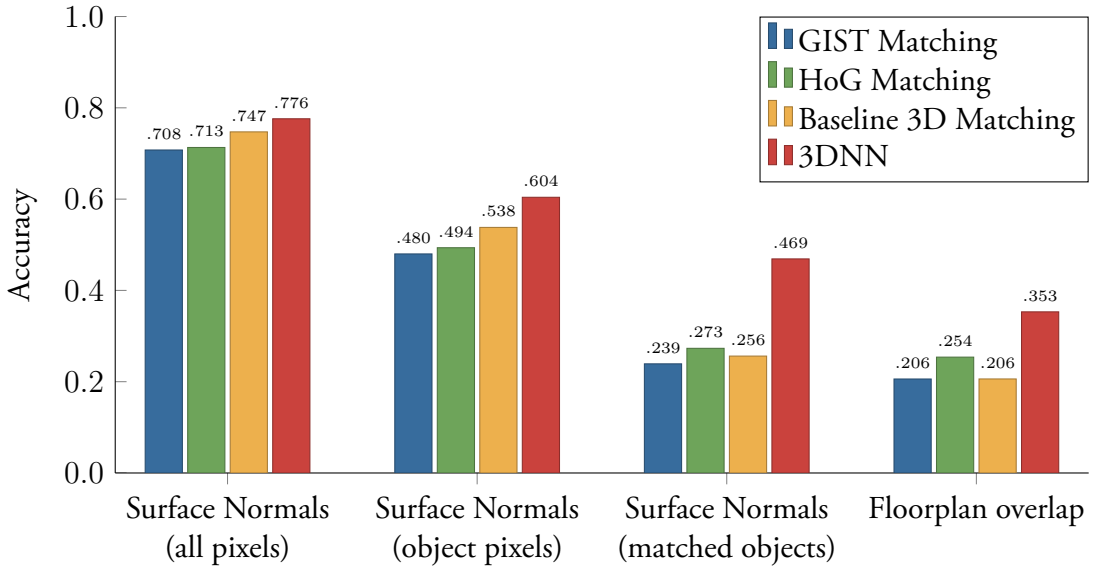


Figure 3.22: Comparison of 3DNN with state-of-the-art 2D nearest-neighbor approaches and the geometry matching algorithm of [Satkin et al., 2012b].

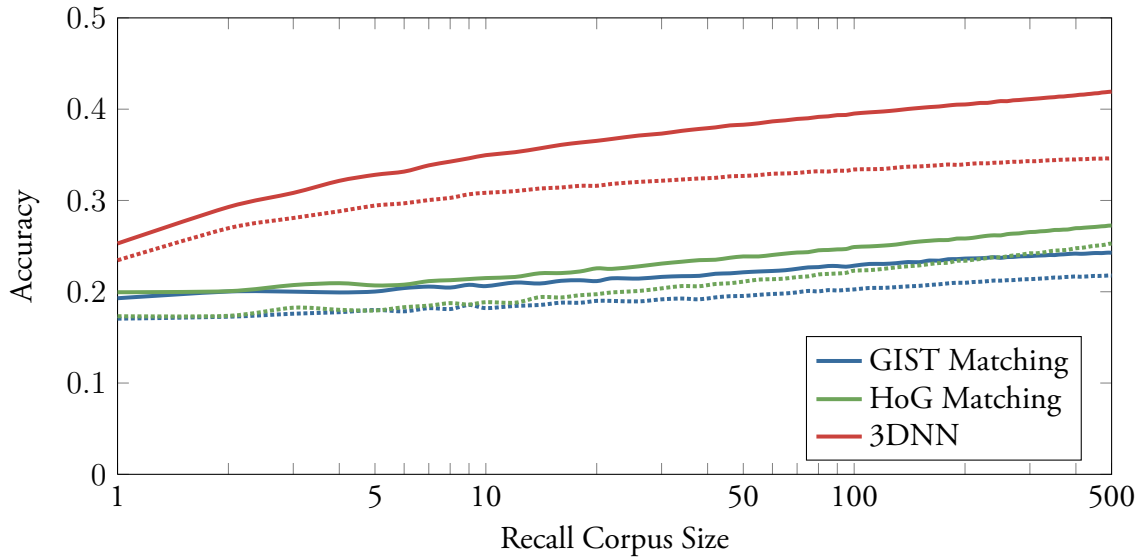


Figure 3.23: Accuracy as a function of dataset size. Solid lines indicate "matched objects surface normal score," dotted lines indicate "floorplan overlap score." Note the logarithmic x-axis.

3.8.3 Dataset Size

It is well known that for appearance-based image matching to be effective, there must be a large recall corpus of images to match with [Hays and Efros, 2007, Torralba et al., 2008]. This is because the data set needs to include recall images captured from a similar viewpoint as the query image. On the contrary for 3DNN, the viewpoint and the geometry of the recall images are decoupled. Thus, each scene provides an exemplar which can be matched to images from any viewpoint.

We evaluate this by experimenting with the size of the recall corpus. Figure 3.23 shows how the performance of 3DNN increases as a function of dataset size, compared to GIST and HoG matching. We report results using two of the more challenging metrics: "matched object surface normal scores" (solid lines) and "floorplan overlap scores" (dashed lines). In these experiments, we consider recall dataset sizes between 1 and 500 images. For each dataset size, we select random subsets of images from the the full recall set, and report the performance of each algorithm on the smaller datasets. Due to the high variance in performance using small recall sets, we average performance across 1000 random subsets of each size. For fair comparison, we do not use our sequence optimization or hypothesis score prediction during these experiments.

There are two important properties of 3DNN we can identify from this graph. Firstly,

note that the red plots for 3DNN start out with a higher accuracy (even for a dataset size of one image). This is because our algorithm starts by estimating the room layout of each image, identifying the locations of floors and walls. On the contrary, GIST and HoG matching do not incorporate this knowledge directly, and must infer the viewpoint of the scene by finding a similar image from the recall corpus.

Secondly, note that the curves for 3DNN are steeper than for the appearance-based approaches. This is because on average, each additional training image provides more information in its geometric form, than the raw pixels used in GIST or HoG matching. This indicates that performance is increasing more quickly as a function of the dataset size, and that fewer training examples are required to achieve the same level of performance using 3DNN compared to a traditional appearance-based 2D nearest-neighbor scene matching approach. Remarkably, 3DNN is able to achieve a noticeable performance boost using a recall set size of only 10 images or fewer, due to the algorithm's ability to generalize across never-before-seen viewpoints.

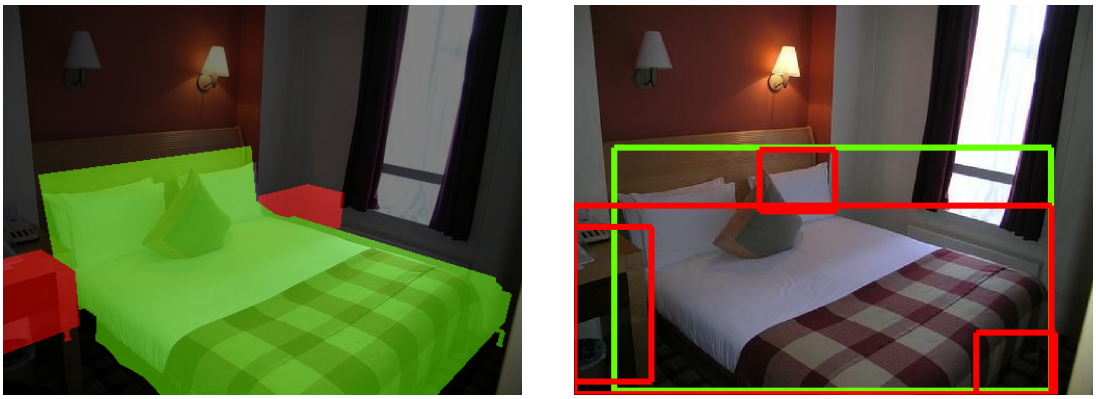
3.9 Conclusion

In this chapter, we presented the 3DNN algorithm which incorporated a series of improvements to our baseline scene matching approach. We described our robust mechanism for simultaneously searching over camera parameters and scene geometries. In addition, we presented an algorithm for refining the locations and styles of objects in 3D to produce more precise results, and the features necessary to achieve this level of fine-grained alignment. Lastly, we showed that 3DNNs can be efficiently computed by intelligently optimizing the order in which geometric hypotheses are evaluated.

We demonstrated the effects of each of the components of the 3DNN algorithm, and compared our approach to traditional 2D nearest-neighbor methods. Because our approach is inherently 3D, we can properly reason about depth ordering and occlusions to produce accurate segmentations of detected objects. In the following chapters, we will demonstrate this ability and explore a series of applications of the 3DNN algorithm.

CHAPTER IV

Application: Object Recognition



(a) Object labels transferred from the top-ranking 3D model using our scene matching approach.

(b) Object detections using Felzenszwalb et al.'s discriminatively trained deformable part model.

Figure 4.1: Comparison of object detection results.

We now explore how our approach to recover the geometry of a scene can be applied to one of the most common computer vision tasks – object recognition. We can transfer the identities of objects by projecting them onto the image plain to produce per-pixel object labels as shown in Figure 4.1a. Additionally, we can intelligently integrate the output of other object detectors to create a more robust result. This approach raises some fundamental questions, which we address in this chapter.

4.1 Object Detection and Segmentation

Our mechanism for inferring the structure of a scene in 3D provides us with rich information about the depth ordering and the occlusions of objects when projected onto the

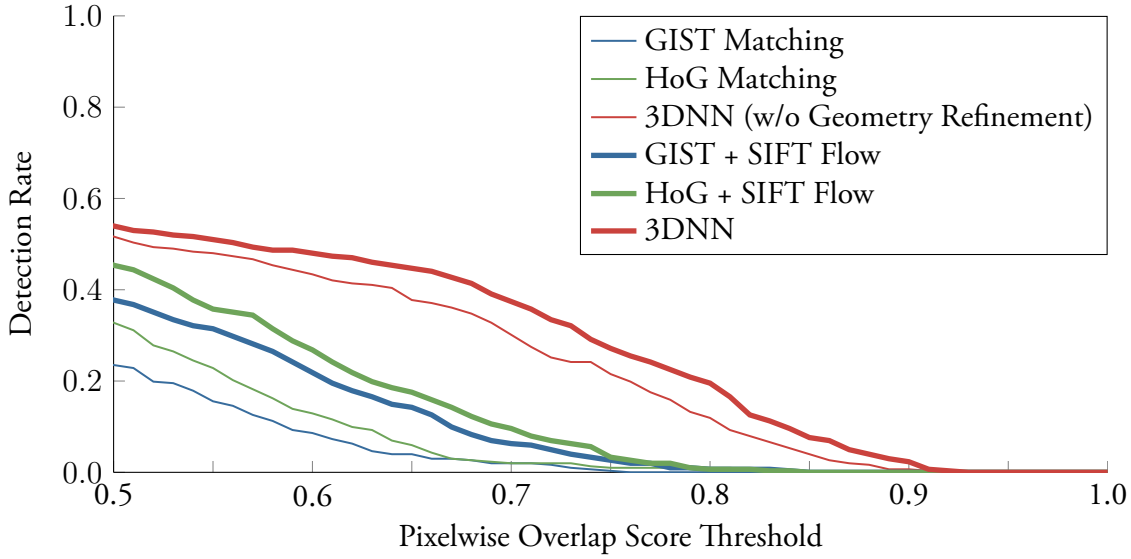


Figure 4.2: Object detection rate as a function of overlap score strictness for the "bed" category.

image plane. Thus, we should be able to not only detect the locations of objects, but also segment their spatial support in the image by precisely identifying their boundaries. To verify that using 3D cues is an attractive alternative for pixel-based object segmentation, we evaluate the per-pixel overlap score of the ground-truth and the object labels estimated by 3DNN.

Figures 4.2 and 4.3 analyze the detection rate of 3DNN, compared to various appearance-based image matching baselines. We measure performance for the "bed" and "couch" categories, two of the most prominent objects in the CMU 3D-Annotated Scene Database. We vary the pixelwise overlap score threshold, and compute what percentage of objects are detected at each threshold. Note that at a stricter threshold of overlap score $\geq .75$, the baseline appearance-based approaches detect very few objects; however, 3DNN still performs well.

Naturally, 3DNN's ability to precisely segment objects is due in part to the geometry refinement stage. To analyze the benefits of this process, we measure the performance of 3DNN with and without the refinement stage. As anticipated, by refining the predicted locations of objects, we achieve a significant (on the order of 5%) boost in detection rate. For fair comparison, we run the SIFT flow algorithm (the state-of-the-art 2D refinement process) as a baseline. The SIFT flow algorithm of [Liu et al., 2008] has been shown to be a robust technique for aligning matched images. By warping each matched scene, SIFT flow refines the location of objects in the image plane, akin to our geometry refinement process.

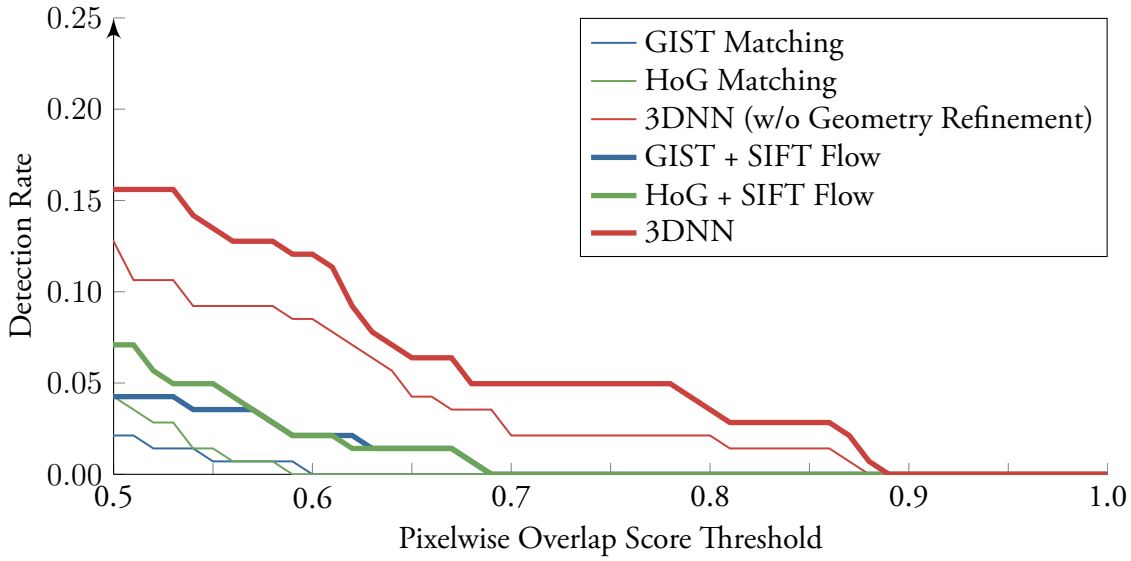
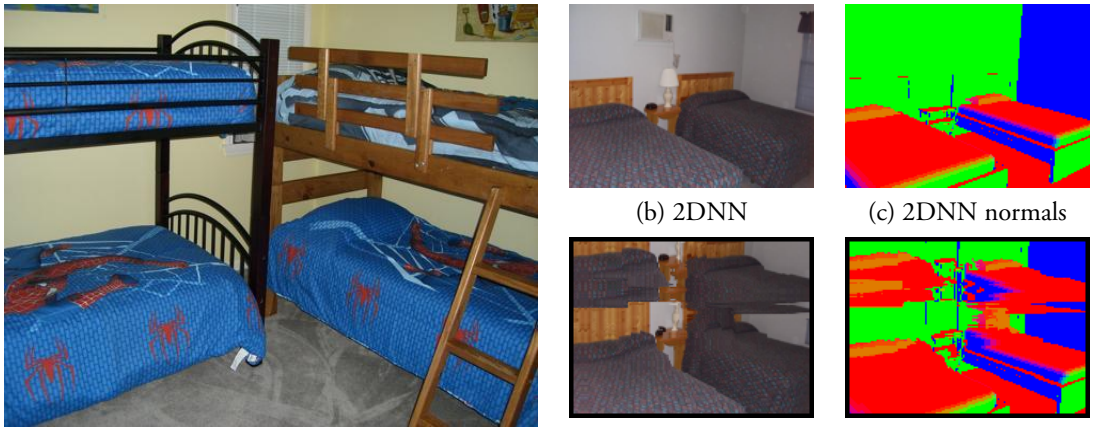


Figure 4.3: Object detection rate as a function of overlap score strictness for the "couch" category.

We apply the SIFT flow algorithm using code provided by [Liu et al., 2008]; this process takes the top-10 scene matches (using either GIST or HoG), warps each matched image, and computes the energy of each warping. We then re-rank the top-10 scene matches according to their SIFT flow energy, and score the top-ranking warped recall image. Although the SIFT flow process yields a significant boost in performance, the algorithm is still not as effective in accurately identifying and segmenting objects compared to 3DNN. Moreover, a key distinction between our geometry refinement process and the SIFT flow algorithm, is that our approach is inherently 3D and produces physically meaningful results. On the contrary, because SIFT flow warps pixels in the image plane, the result no longer has a coherent 3D interpretation as shown in Figure 4.4.

4.2 Integrating Discriminative Object Detections

In recent years, the PASCAL Visual Object Classes Challenge [Everingham et al., 2010] has sparked tremendous progress in object recognition. Bottom-up appearance-based object detectors such as the discriminative deformable parts model of [Felzenszwalb et al., 2008] have been at the forefront of the object recognition field. As a baseline for comparison, we train [Felzenszwalb et al., 2010b]'s part-based detector using code provided by [Girshick et al., 2012]. It is important to note that object detectors are in a fundamentally different



(a) input image (b) 2DNN (c) 2DNN normals (d) SIFT flow (e) SIFT flow normals

Figure 4.4: Visualization of the SIFT flow algorithm [Liu et al., 2008]. Note the incoherent output which no longer has a plausible geometric interpretation.

class of algorithms than our approach. Most discriminative object detectors have a tunable parameter which can be adjusted to achieve arbitrarily high recall by predicting the object at all locations and scales in the scene (at the cost of reducing precision). However, our approach produces a single 3D model for each image, and uses physical constraints which do not allow objects to by hypothesized at arbitrary locations. Thus, our output represents a single point on the precision recall curves.

Many researchers have successfully demonstrated the ability to integrate the output of multiple object recognition systems to create a more robust detector. For example, in [Hedau et al., 2010], the authors combine the scores from their "boxy object detector" with [Felzenszwalb et al., 2010a]'s discriminatively trained deformable part model to produce a state-of-the-art bed detector. Hedau et al. showed that while each of the two approaches perform well on their own, the integration of the two algorithms results in a significant performance boost. Their approach assumes the outputs of each algorithm are statistically independent and multiplies the scores from their boxy object detector with the scores from the deformable parts model.

Following the paradigm of [Hedau et al., 2010], we integrate our object detections with the bounding-box detections from [Girshick et al., 2012], as a post-processing step. For each object detection, we boost the DPM detector's confidence if our result is in agreement. Specifically, we compare each of the bounding boxes from our results with those from the DPM. If their overlap score is greater than a set threshold, we increase the DPM's confidence by a fixed amount. To determine the best values for this threshold and the amount by which

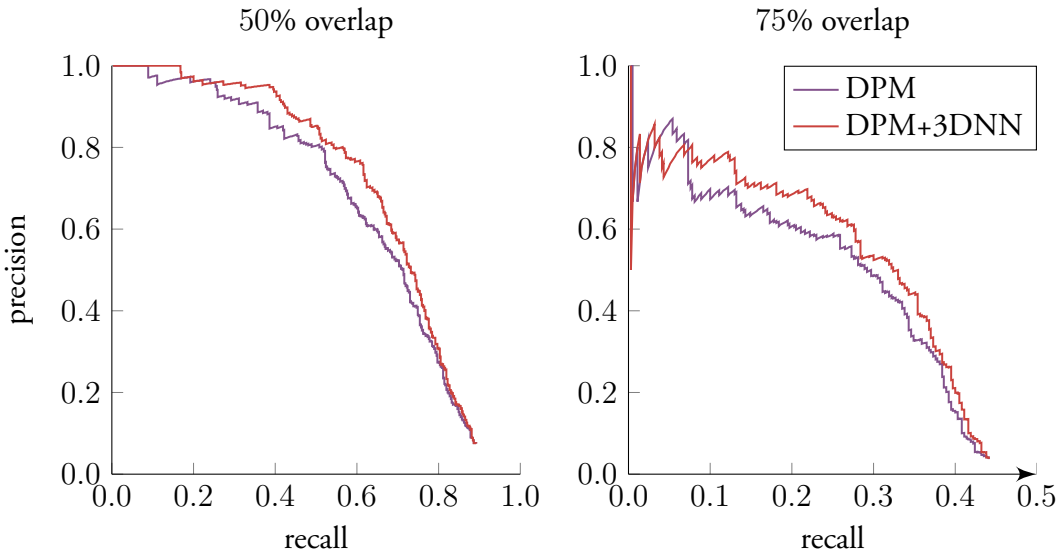


Figure 4.5: Object detection precision recall curves for the "bed" category. Our approach provides complementary information to DPM; thus, integrating both results improves performance.

to boost the DPM's confidence, we perform a grid search after partitioning the dataset into 5 folds (80% train, 20% test).

Figures 4.5 and 4.6 show the performance of [Felzenszwalb et al., 2010b]'s deformable parts model (DPM) with pairs of precision recall curves for the bed and couch categories, respectively. The plots on the left use the traditional 50% overlap score measure. On the right, we measure performance using a 75% overlap score threshold; this is a stricter metric, which requires objects to be more precisely localized. The purple plots report the performance of the DPM detector and the red plots show the performance of integrating the DPM's detections with our object predictions. Note that the combined detection results provide a modest improvement (~10% relative increase in average precision) over the baseline DPM accuracy. This indicates that our approaches to object classification and geometry estimation provide complementary information. Experimentally, we achieved similar results to [Hedau et al., 2010]. In isolation, the authors' cuboid detector does not perform as well as [Felzenszwalb et al., 2010a]'s DPM; however, by intelligently integrating the results of their cuboid detector and the DPM's results, they achieve state-of-the-art performance.

Quantitatively, the DPM out-performs our approach when measured using a liberal 50% overlap score detection threshold. However, as the overlap score threshold is increased, our approach shows a modest improvement over DPM. This result is not surprising – the features

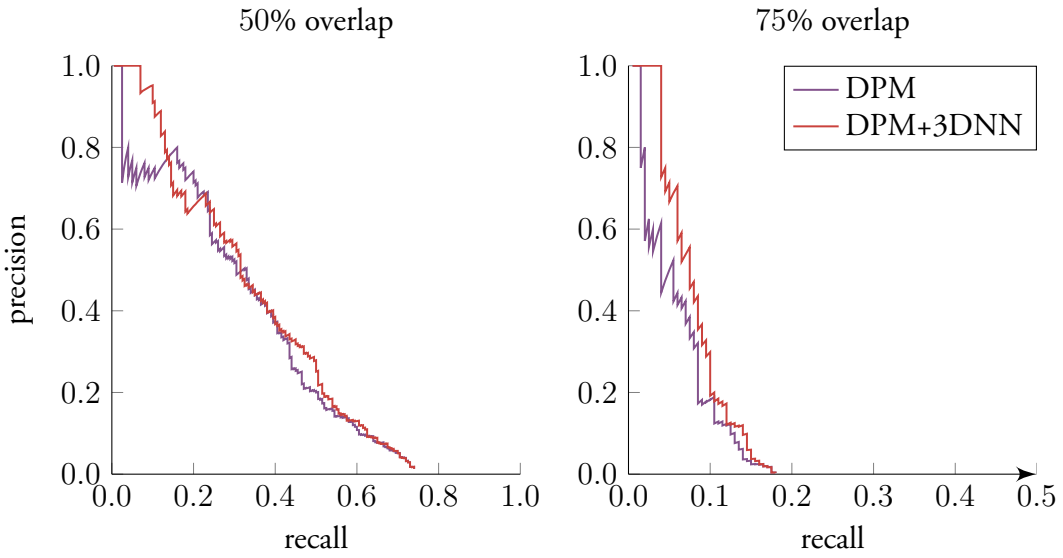
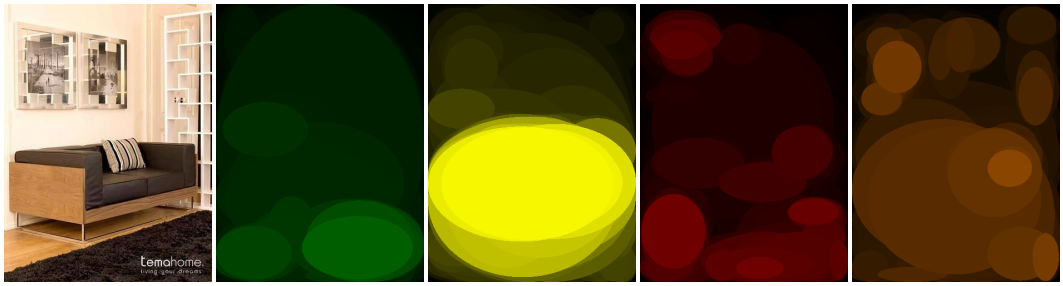


Figure 4.6: Object detection precision recall curves for the "couch" category. Our approach provides complementary information to DPM; thus, integrating both results improves performance.

and model used in [Felzenszwalb et al., 2010b]'s detector are discriminative in nature and explicitly aim to classify each object in the scene. On the contrary, our approach uses features which intentionally ambiguate between object categories. For example, the $p(\text{object})$ feature simply predicts whether or not an object is present, not the identity of the object. This suggests a new family of features which could be integrated into the 3DNN algorithm.

Here, we propose a mechanism for incorporating the outputs of an object detector into a similarity feature to leverage the performance of these successful discriminative approaches in a manner similar to [Li et al., 2010]'s Object Bank features. Figure 4.7 shows an input image, and heatmaps indicating the probability of a specific object category appearing in the image, computed using the deformable parts model of [Felzenszwalb et al., 2008]. Because the bounding-box detections are not well-suited for localizing and segmenting objects, we place ellipses at each object detection location, with values proportional to the detection confidence. To convert the raw heatmaps output from [Felzenszwalb et al., 2008]'s algorithm to features which represent the likelihood of each object appearing at specific locations, we perform a nonlinear scaling of the result using a sigmoid function. The parameters of the sigmoid are learned using a logistic regressor trained on validation data. A similarity feature can be computed for each object category, by calculating the normalized dot product between the each $p(\text{object})$ calibrated heatmap (scaled to be in the range $[-1, 1]$) and the



(a) input image (b) p(bed) (c) p(couch) (d) p(table) (e) p(chair)

(f) example object category renderings

Figure 4.7: An input image (a), and heatmaps (a)-(e) indicating the probability of a specific object category appearing at each pixel in the image, computed using the deformable parts model object detection algorithm of [Felzenszwalb et al., 2010a]. Example object category renderings of 3D models (f), aligned to the input image. Each color represents a different object category (e.g., couch=yellow, table=red, orange=chair).

corresponding object label rendering (as shown in Figure 4.7f) for that object class.

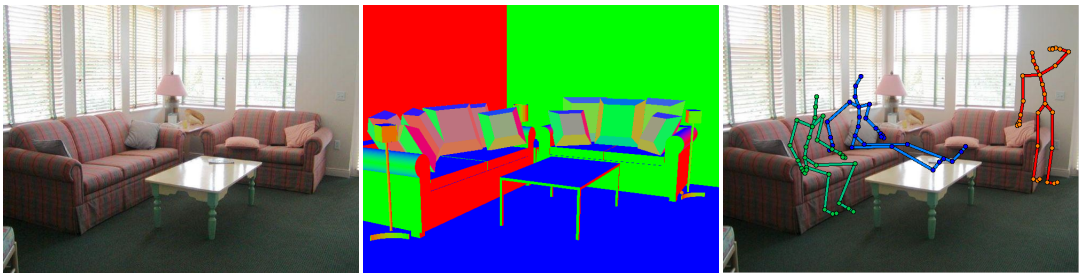
Our initial experimentation with this similarity feature shows that this method can successfully constrain our search to select models which are in agreement with the output of the low-level object detections. However, if too much weight is put on these features, the resulting 3D models we select will have the same errors as the object detectors. Our current max-margin training routine aims to optimize the geometry and locations of objects in each scene. This yields excellent performance for tasks like freespace and affordance estimation; however, optimizing classification accuracy is a fundamentally different objective, which is beyond the scope of this thesis. Future work could address, "What is the correct way to balance the tradeoffs between object class discrimination and 3D reconstruction when ranking scene hypotheses?"

4.3 Scene Matching as a Prior for Object Locations

Not only can we use the labels of components in each 3D model to predict the identity of objects in a scene, this information can also serve as a geometric prior to aid in the detection of unidentified objects. For example, we can ask, "Where would we be likely to find a cup, pillow or lamp in this image?" Given the location of tables, beds or nightstands, and known co-location priors for these object categories, we could potentially identify locations in an image where we would expect these objects to appear. This type of co-occurrence prior has been studied in many contexts. For example, [Desai et al., 2009] model the spatial co-occurrence of objects in the image plane to improve non-maxima suppression. Additionally, [Fisher et al., 2011] characterize structural relationships in scenes directly from 3D data. The ideas from each of these works can be integrated with our scene matching approach to model where objects are likely to appear using 3D co-occurrence statistics. See Chapter VI for an example application which uses object detections and co-occurrence statistics to predict realistic locations for objects to be added to a scene.

CHAPTER V

Application: Affordance Estimation



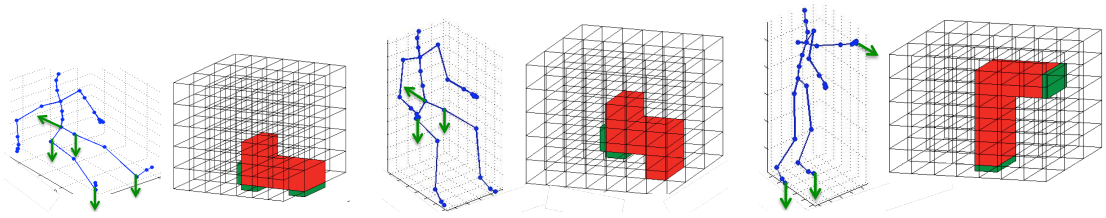
(a) Input image (b) Geometry estimation (c) Our human-centric representation

Figure 5.1: (a) An input image, (b) a geometry estimated using the approach presented in Chapter II, (c) our human-centric representation.

Given a detailed geometric representation of a scene, there are many possible higher-level interpretations that we can generate. In this chapter, we summarize our work in affordance estimation [Gupta et al., 2011]. This work leverages the availability of motion capture data and derives a novel human-scene interaction model capable of predicting the locations of possible human poses from a single image. Our affordance estimation process utilizes an intermediate geometric representation of a scene. We show the importance of high-quality geometry estimates for this problem, and incorporate our data-driven geometry estimation algorithm to improve upon baseline approaches.

5.1 Algorithmic Overview

This work is an attempt to marry 3D scene understanding with human action modeling. We present a novel qualitative scene representation that combines 3D scene geometry with



(a) Sitting Reclined (b) Sitting Upright (c) Reaching

Figure 5.2: Qualitative Representation of Human Poses: Each pose is represented by the occupied blocks in discretized 3D space (shown in red) and the required surfaces of interaction (shown in green).

a set of possible human actions, to create a joint space of human-scene interactions. A key insight is to note that there are only two constraints on a 3D human pose that are relevant for embedding it within a given geometry: 1) the 3D space (volume) the pose occupies, and 2) the surfaces it is in contact with. We divide the space around the human actor into blocks (Figure 5.2) and associate each block with a 0 or 1 based on whether the block is occupied or not. In addition, each block may require an external support in a particular direction. For example, in the sitting pose (with back support), we need a horizontal surface below the pelvic joint to support the body and a vertical surface to rest the spine (Figure 5.2b). In a similar manner, for the "reaching" action (Figure 5.2c) a horizontal support is required at the feet and a vertical surface of interaction is required to represent the point of contact of the hands.

By discretizing the scene geometries and human poses into an occupancy matrix, we can efficiently search for locations and poses which satisfy the free space and support constraints. We slide the discretized human blocks around the scene occupancy matrix using a binary correlation operation. Intuitively, we are searching for locations for which the human pose does not intersect any objects. Additionally, the locations must have the appropriate supporting surfaces to afford each pose. Both of these constraints can be satisfied using simple correlation operations. To account for the deformation of furniture and the human body, we perform an erosion and dilation process on a scenes's occupancy matrix before performing the correlation operations. See [Gupta et al., 2011] for a detailed description of this algorithm.

We demonstrate the capabilities of our human-centric representation by running our human-scene interaction model on a few synthetic 3D indoor scenes downloaded from the Google 3D Warehouse. Figure 5.3 shows the locations where a human can sit for two

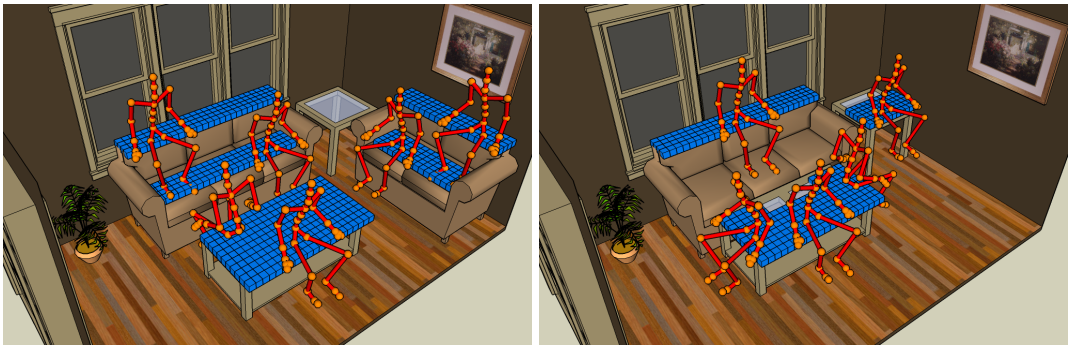


Figure 5.3: Human-centric representation on a synthetic 3D scene: given an example "sitting" pose, we visualize where a human can sit in the environment (blue mask shows all possible pelvis locations). Note how rearranging a few objects within the scene can have a big influence on the estimated human workspace.

possible furniture configurations. It can be seen how our representation captures the spatial arrangements of objects and how affordances change for the same objects under varying configurations. While in Figure 5.3(left) a person can sit on the left couch, the same couch is no longer accessible for sitting in Figure 5.3(right) because the table is moved too close to it. Similarly, the side table becomes accessible in the bottom figure, once an obstruction is removed.

5.2 Qualitative Results

We now evaluate the utility of our 3D geometry estimation algorithm for determining the affordances of a scene. Figure 5.4 shows the results of this approach. To visualize the whole range of possible poses, we overlay colored masks indicating the locations of pertinent joints for a given pose. For example, we show in blue the locations where the pelvic joint makes contact with a valid surface of support for the "sitting reclined" task. We also indicate in cyan the locations where the back makes contact with a vertical support. Example human stick figures (extracted from the mocap data) show representative valid poses in each scene. As is evident from the stick figures, our approach predicts affordances that cannot be represented by basic object categories. For example, on the "sitting reclined" pose, our approach combines the vertical surface of the bed with the horizontal surface of the ground to predict human poses. Similarly, for "sitting upright" our approach finds valid pose locations that cannot be predicted by object-level categories such as chairs or couches. For example, as shown in Figure 5.5, we can predict unusual sitting poses (on the back of the chair and atop

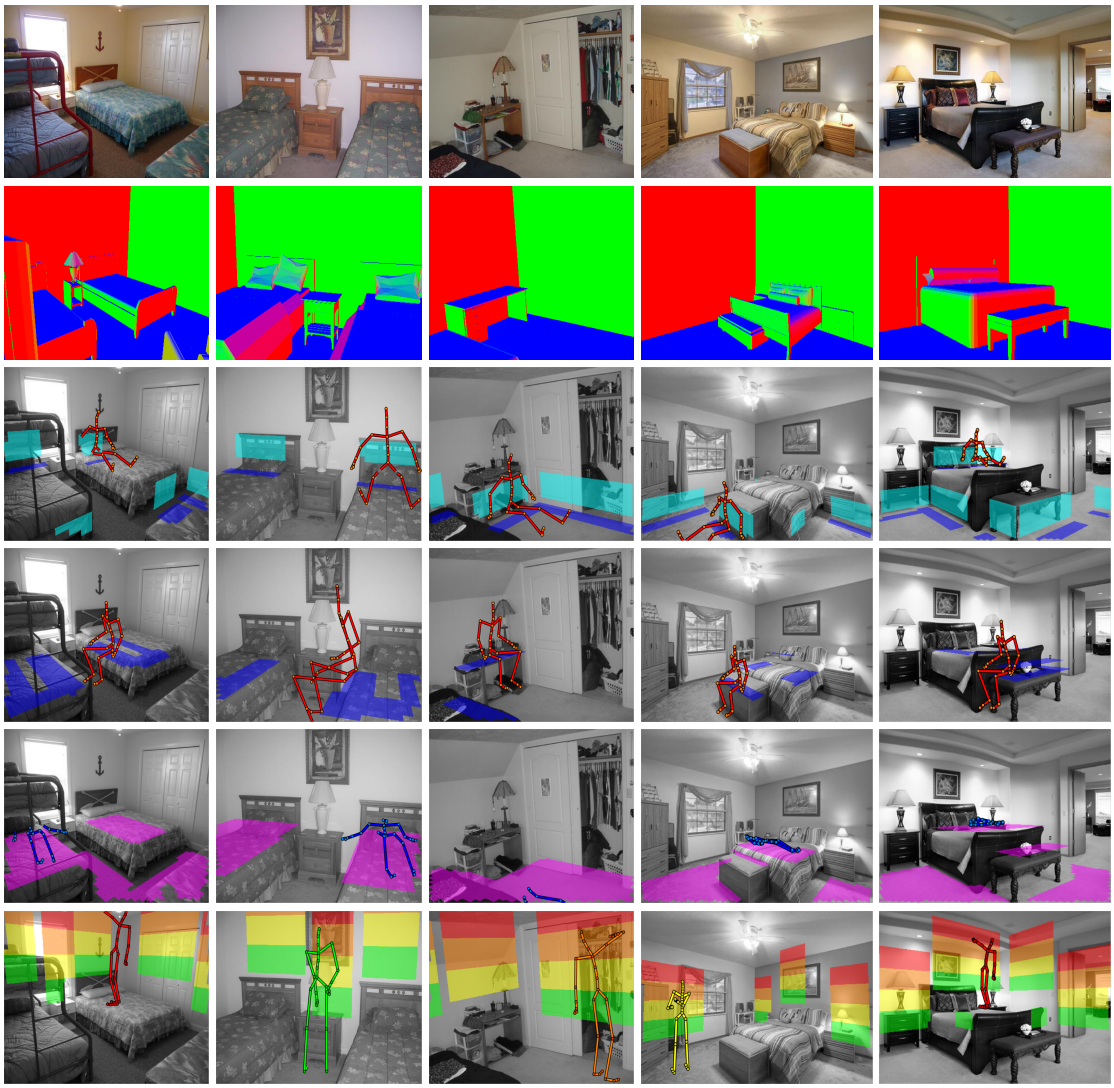


Figure 5.4: Qualitative performance of our affordance estimation approach. The images in the first row are the input to our algorithm. The second row shows our estimated 3D scene geometry. The third row shows the possible pelvic joint and back support locations in blue and cyan respectively for the "sitting reclined" pose. The fourth row shows the possible pelvic joint locations in blue for the "sitting upright" pose. The fifth row shows the locations where a human's back can rest when "laying down." The last row shows the vertical surfaces a person's hand can touch from a standing position for the "reaching" pose, color coded to indicate the corresponding pose. Each scene also includes a representative stick figure for each pose.

Figure 5.5: Examples of unusual, yet physically valid poses computed using our geometry and affordance estimation approach.

the television), which despite being unlikely, are physically valid.

5.3 Quantitative Evaluation

To quantify the performance of our affordance estimation algorithm, we measure how accurately we can predict the possible locations of human poses in each scene. We evaluate our affordance estimation algorithm using the data-driven scene matching approach for geometry estimation presented in this thesis, as well as the geometry estimation approach from [Gupta et al., 2011]. The approach of [Gupta et al., 2011] is a precursor to our geometry estimation algorithm, which uses a simplified geometry estimation process to find a set of cuboids which model the arrangement of furniture in a scene. This approach begins with the same autocalibration technique discussed in Section 2.1; however, it considers a limited number of geometry hypotheses, commits to a single viewpoint estimate, and uses only the original $p(\text{object})$ similarity feature for scoring each hypothesis.

We also compare our algorithm with a standard appearance-based baseline; training a separate classifier for each task. These methods have shown good performance for different pixel labeling tasks, such as object categorization and qualitative geometry estimation. Each pose classifier uses appearance features computed from an image to label the pixels where a relevant body joint can appear for that human pose. For example, the "sitting upright" classifier predicts where a person can sit by indicating where the pelvic joint could rest in an image when the person is sitting. Specifically, we use the image features and multiple segmentations classifier of [Hoiem et al., 2007a], and 50 training images for each classifier.

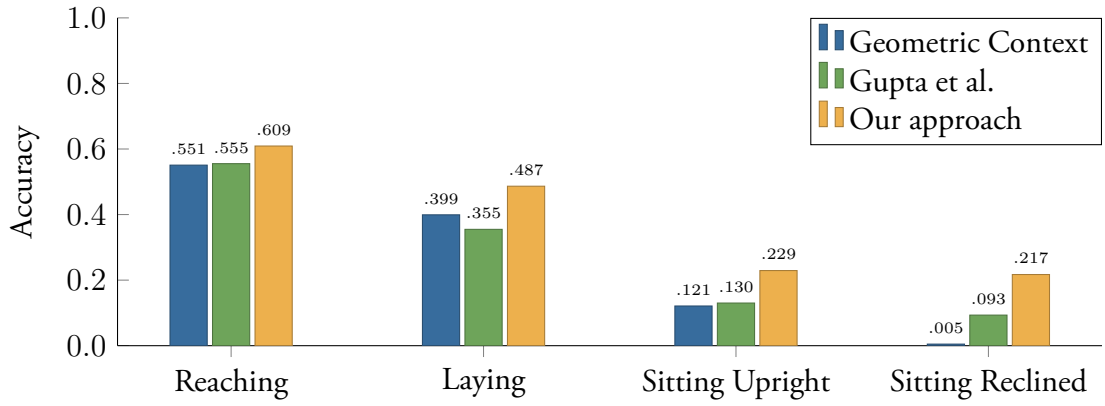


Figure 5.6: Quantitative comparison of our affordance estimation algorithm with two baselines.

We manually labeled 50 test images for four poses:

(a) Locations a pelvic joint can rest while sitting upright,

(b) Locations a pelvic joint can rest while sitting reclined,

(c) Locations a human's back can rest when laying down,

(d) Locations a hand can reach on a vertical surface.

Using these annotations, we compute the pixelwise overlap score between each algorithm's predicted pose locations and the ground-truth. Figure 5.6 shows the performance of our approach compared to the baseline appearance classifier using geometric context, and affordances computed using the coarse geometry estimates from [Gupta et al., 2011] on each of the four classes.

Note that our approach outperforms the two baselines for all poses. While the baseline approaches do a decent job in prediction valid locations for "reaching" and "laying down," their performance is markedly lower for the "sitting upright" and "sitting reclined" poses[1]. Intuitively, this is because detecting locations for reaching and laying simply requires identifying the locations of walls and floors in an image, which spatial priors can easily encode. This is akin to easier categories such as ground and sky in pixel classification literature. However, accurately detecting possible sitting poses requires precise estimates of scene geometry and cannot be captured via appearance-based approaches.

[1] Note: In [Gupta et al., 2011], their approach outperformed the appearance baseline. This is due to the selection bias in creating the dataset for that paper, such that only images with accurate autocalibration estimates from [Hedau et al., 2009] were used. However, for these experiments an unbiased sample of images were selected.

CHAPTER VI

Application: Geometry-Aware Object Insertion

Many graphics applications involve the insertion of new objects into existing images or videos. In order to achieve photorealistic results, the geometry and lighting of a scene must be known (or estimated). Without accurate models of a scene it is impossible to properly reason about how objects should be positioned and oriented, which portions of these objects are visible or occluded, and how complex lighting interactions such as reflections and shadows should be properly rendered.

In this section, we explore existing algorithms for synthetic object insertion and demonstrate how our ability to automatically estimate the detailed geometry of a scene enables photorealistic object insertion.[1]

6.1 Background

[Lalonde et al., 2007]'s "Photo Clip Art" begins to address this problem. In their work, the authors present a system which reasons about camera pose and lighting to allow the realistic insertion of objects into an image. Their approach begins by first estimating the viewpoint of the scene (camera height and horizon position) using the algorithm of [Hoiem et al., 2006]. This allows their system to properly reason about perspective, select objects which are correctly oriented, and automatically place them on the ground plane at the right scale, based on their distance from the camera. The authors also present "illumination context," a set of color histogram based features which allows them to automatically select objects for insertion which appear to be captured from similar lighting conditions as the input image (alleviating the need for relighting). Lastly, the authors present a context-sensitive

[1]Note: The object insertion renderings included in this chapter were created by Kevin Karsch at the University of Illinois at Urbana Champaign during a collaboration with the author.

blending algorithm for realistically compositing the new objects into the image and transferring their shadows. The authors integrate each of these algorithms into their "Photo Clip Art" application which enables users to quickly insert objects (such as people and cars) into a scene and produce photo-realistic results without worrying about matching the illumination conditions.

In [Karsch et al., 2011], the authors show that better geometric models of scenes allow for more precise estimation of illumination, enabling photorealistic insertion of synthetic objects into legacy photographs. The authors presented a framework which allows a user to quickly annotate the geometry of a scene, and roughly specify the locations of directed light sources. Their system then automatically generates a physical model of the scene including the position, shape and intensity of light sources. Given an estimated scene geometry (via user annotation), the authors present an algorithm to recover the material properties and illumination conditions in the scene. Their method uses intrinsic image decomposition to determine the albedo, direct illumination and indirect illumination of a scene by building upon [Grosse et al., 2009]'s Color Retinex algorithm and [Guo et al., 2011]'s shadow detection and removal algorithm. After estimating the geometry, lighting and material properties of the scene, [Karsch et al., 2011] allow users to position new 3D objects into the scene which are photorealistically rendered using the additive differential rendering method of [Debevec, 1998] and the spectral matting algorithm of [Levin et al., 2008]. The rendered results of their approach are high-fidelity. In fact, the results are often photorealistic enough that humans often cannot differentiate between objects which were originally in the image and objects which were inserted.

6.2 Approach

A fundamental limitation of [Karsch et al., 2011]'s approach is their reliance on geometry. This currently requires users to manually annotate each scene's geometry, a process which is both painstaking and imprecise (geometry is usually approximated with cuboids). However, by integrating our geometry estimation algorithm, we can overcome the need for human geometry annotation. Specifically, we can incorporate our automatically estimated camera parameters, surface normals, occlusion masks and depth orderings rather than using human annotations.

Figure 6.1 shows an example scene with three synthetically inserted objects: a dresser behind the bed, a lamp on one nightstand, and a small kinetic sculpture on the other night-

(a) Input image. (b) Synthetically inserted objects.

Figure 6.1: Examples of synthetically inserted objects. Note the accurate occlusion and depth ordering of the dresser, and the realistic reflections.

stand. By using our estimated camera parameters, the added objects are automatically scaled and oriented correctly, and the perspective effects of these objects are in correspondence with the scene's vanishing points. The estimated scene geometry enables objects to be automatically positioned to lay upon horizontal surfaces. Moreover, our precise scene geometry enables automatic occlusion reasoning (as seen with the dresser behind the bed). Not only is the scene geometry useful for positioning each object, the estimated surface normals and depths are critical for ensuring proper lighting effects. For example, the reflection of the bed is visible in the inserted mirror, and the light emitted from the lamp is realistically scattered off the edge of the bed.

Our estimated object categories and orientations can also be used to automatically position new objects in a scene at realistic locations using co-occurrence priors. For example, if we know the position and orientation of a couch in a scene, we can infer the most likely location of a coffee table relative to the couch's position. See Appendix B for a description of our data-driven approach for computing object co-occurrence priors.

6.3 Application: Augmented Reality Product Catalog

We now demonstrate a proof-of-concept application which incorporates our autocalibration and geometry estimation approach with [Karsch et al., 2011]'s synthetic rendering algorithm to create an augmented reality product catalog. The goal of this application is to allow users to see how furniture would look in their home. We run our geometry estimation

(a) IKEA Granås (b) IKEA Hol

(c) IKEA Lack (d) IKEA Liatorp

(e) IKEA Hemnes (f) IKEA Strind

(g) IKEA Vejmon (h) IKEA Lack

Figure 6.2: Photorealistic synthetic furniture insertions from the IKEA catalog.

(a) IKEA Bankas (b) IKEA Klubbo

(c) IKEA Lack (d) IKEA Liatorp

(e) IKEA Hemnes (f) IKEA Strind

(g) IKEA Vejmon (h) IKEA Lack

Figure 6.3: Photorealistic synthetic furniture insertions from the IKEA catalog.

Figure 6.4: Mockup of an augmented reality product catalog iPad application.

pipeline on images and identify the positions and categories of each object present in the scene. Then, using our known co-occurrence priors, we recommend objects which have a high likelihood of co-occurring with objects currently in the scene. Not only can we recommend what object could be added to a scene, we automatically position and orient the objects relative to other objects in the scene. For example, a coffee table should be centered in front of a couch with approximately 1.5ft in between the objects.

Figures 6.2 and 6.3 show example scenes for which we automatically insert coffee tables from the IKEA catalog [IKEA, 2013], using texture mapped 3D models from [Polantis, 2010]. Note that the coffee tables are automatically sized and positioned at realistic locations in the scenes. In addition, the high-fidelity rendering approach of [Karsch et al., 2011] produces photorealistic results with accurate lighting effects (reflections, scattering, shadows, etc.). Figure 6.4 shows a mockup of our object insertion and photorealistic rendering algorithms being incorporated into an iPad application. This technology could enhance a user's shopping experience, by allowing them to see how products would fit and appear in their homes, alleviating their uncertainty.

CLOSING THOUGHTS

"Where there is matter, there is geometry." – Johannes Kepler (1602)

Recovering the 3D structure of a scene from a single 2D projection is an inherently ill-posed problem, and remains a tremendous challenge for vision researchers. However, we believe it is an important problem to address, as determining the geometry of an environment provides a complete representation which can be used to answer virtually any question about our world, ranging from object categorization and localization to freespace and affordance estimation.

To tackle this problem, we leveraged the recent growth and availability of 3D data to integrate geometric reasoning and machine learning. The methods presented in this thesis should not be thought of as a final algorithm, rather they represent a general framework which can be extended and adapted for different tasks. There are four key parts of this framework:

1. A mechanism for aligning 3D data with images via autocalibration
2. A data-driven means for generating geometric hypotheses
3. A means for comparing each hypothesis to an input image via rendering
4. An algorithm for ranking each hypothesis using similarity features

Different environments, such as outdoor settings, would require an alternative mechanism for alignment and similarity features which are tuned for the task; however, the overall 3DNN architecture can be directly adapted.

APPENDICES

APPENDIX A

System Applicability

Throughout this thesis, experiments were conducted on images of bedrooms and living rooms. These scene categories are a small, but representative sample of [Xiao et al., 2010]'s SUN database. A natural question to pose is, "Can this algorithm be applied to images of any scene?"

When creating a dataset for this research, we selected living rooms and bedrooms for three reasons. Firstly, these scenes are the two most common categories in the SUN database. Secondly, these scenes contained large numbers of 3D models in Google Warehouse. Lastly, bedroom and living room scenes have different levels of diversity, enabling us to explore the performance of our approach on images with varying degrees of difficulty.

Bedrooms tend to have limited variation; the diversity of objects found in bedrooms and their relative positions can be clustered into a few canonical configurations. On the contrary, living rooms are less structured and contain a wider diversity of object categories, sizes and configurations. Fundamentally, our approach (like all non-parametric matching algorithms) requires there be a scene in our library of 3D models which is similar to the input image. Intuitively, the more unique a scene is, the less likely we are to find a good geometric match in our library of 3D models. We verify this hypothesis by analyzing the accuracy of our system as a function of each input image's uniqueness.

We compute the "uniqueness" of each scene by comparing its ground-truth geometry to the geometry of every other scene in our dataset. Using the floorplan overlap score, we first compute how similar a given scene is to each other scene's geometry. If an image has a large number of scenes with which it has a high floorplan overlap score, the scene is not very unique. Thus, we quantify a scene's uniqueness using one minus the 95% percentile

(a) 1st (b) 25th (c) 50th (d) 75th (e) 99th

Figure A.1: Images sorted by increasing uniqueness, from left to right, the 1st to 99th percentiles.

of floorplan overlap scores with the remaining scenes in the dataset. Figure A.1 shows examples of scenes from our dataset, sorted by increasing uniqueness. Note that the first few examples include bedroms in canonical configurations; however, the examples towards the right include object configurations which are less frequently seen.

Figure A.2 analyses the performance of our system for increasing levels of image uniqueness. The accuracy of our approach is highly correlated with the uniqueness of the test images: linear correlation $\rho = -0.6033, p = 1.0654^{-49}$ [Pearson, 1896]. The more unique the scene is, the less accurate our result will be. Note that the living room scenes (indicated with orange marks) are significantly more unique than the bedrooms, providing us with a challenging set of scenes for experimentation.

This analysis provides insight into the peformance of our algorithm when applied to new scene categories. Using this regression model, we can mine through annotated scene geometries for these new categories and compute each image's uniqueness to predict the accuracy of our approach when applied to the data. Thus, although we have only presented experiments on two scene categories, we have explored how effective our 3DNN algorithm is on relatively simple (bedroom) and difficult (living room) scenes. Moreover, we have provided a data-driven mehanism capable of determining how effective our geometry estimation approach would be on any indoor scene.

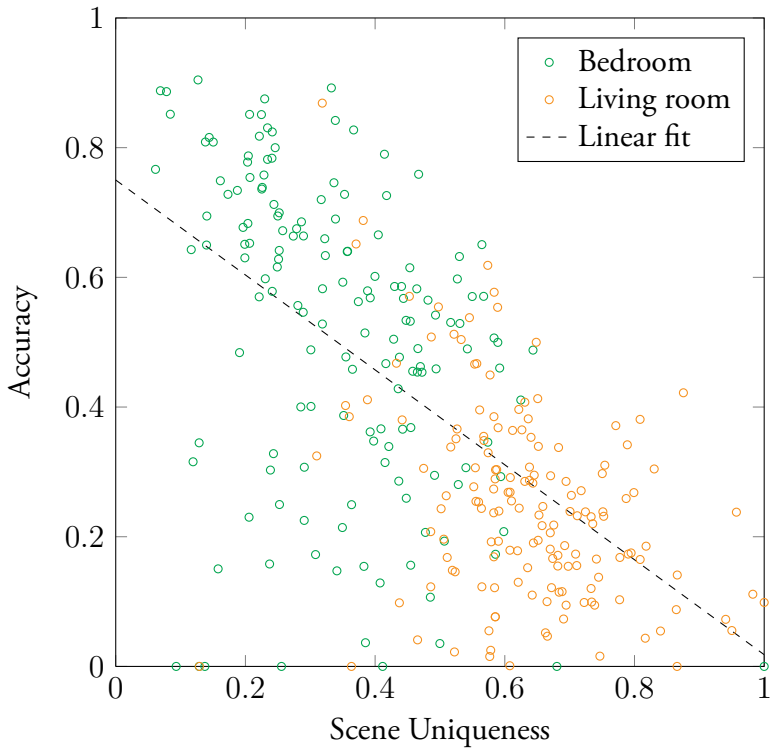


Figure A.2: Scatter plot of Scene Uniqueness vs. Accuracy (floorplan overlap score).

APPENDIX B

Object Co-occurrences

In this appendix, we summarize our data-driven approach for computing detailed object co-occurrence priors. We leverage the vast repositories of 3D models now available to harvest statistics regarding how often different object types appear together, and in what relative configuration they are most likely to appear. This information enables us to not only recommend new objects to be added to a scene, but also predict exactly where the objects are likely to appear, as demonstrated in Chapter VI.

We take a Bayesian approach to computing these co-occurrence statistics. For each pair of object types, we mine through our library of models from 3D Warehouse [Trimble Inc., 2012], and aggregate statistics regarding the relative positions the objects. Most types of furniture are not functionally invariant to rotation. Beds, chairs, couches, and other common household objects have fronts and backs which dictate how they are used. Moreover, each object pair has an archetypal configuration (or set of archetypal configurations), which define how these objects are likely to co-occur in scenes. These configurations cannot be accurately modeled with simple distributions, such as the distance between the objects. For example, not only are nightstands likely to appear very close to beds, they are typically positioned beside beds, near their headboards, such that their corners are abutting. To accurately model these relations, we must first determine the orientation of each object.

Object orientations are identified using two methods. This process begins by manually annotating a small subset of objects to label their front, back, left, and right faces.[1] These annotated templates are used as references to automatically determine the orientation of the

[1]Note: The labels assigned to each face are simply a convention. Any face could be assigned any label, as long as they are consistent within each category.

(a) back (b) right (c) front (d) left (e) top

Figure B.1: Example face heightmaps used for determining the orientations of objects. Yellow indicates low depth, and red indicates higher depths.

remaining objects. Here, we aim to select the rotation $\theta \in \{0°, 90°, 180°, 270°\}$ which will bring an object into closest alignment with the set of annotated templates. The similarity between an object and a template is computed geometrically by comparing five 2D heightmaps corresponding to the front, back, left, right, and top faces of each object pair using normalized correlations. Figure B.1 shows an example of these heightmaps used for computing object orientations. The intuition behind this approach is that there are certain structural characteristics unique to different object categories which encode their orientations. For example, the backs of chairs and and sofas tend to be higher than their fronts, resulting in a distinctive "h" shape when viewed from the left side. Similarly, the headrests of beds tend to be higher than their feet, and pillows tend to be positioned towards the heads of beds, as shown in Figure B.1.

This approach fails for highly symmetric boxy objects, such as tables or nightstands. Each face of these objects is geometrically similar, making it difficult to determine the objects' orientations based solely on shape. However, the position of these objects in an environment can disambiguate their orientations. For example, dressers and nightstands tend to be positioned such that their back side is against a wall. Thus, for object categories which are mostly cuboidal (nightstand, table, dresser, etc.), we simply identify the face closest to a wall as the back of the object.

After determining the orientation of each object in our 3D model library, we can now compute pairwise co-occurrence statistics. Unlike previous approaches such as [Choi et al., 2013]'s 3D Geometric Phrases, which model the distances between centroids of objects, our priors are finer-grained and model particular spatial relations. For each pair of object categories, we consider 12 possible relations, shown in Figure B.2. Each of the red points on the front, back, left, and right sides of the object are used as anchors, and the distribution of distances from each anchor point to the nearest corner or edge on the second object is aggregated in two-dimensional histograms corresponding to distances in the u and v directions.

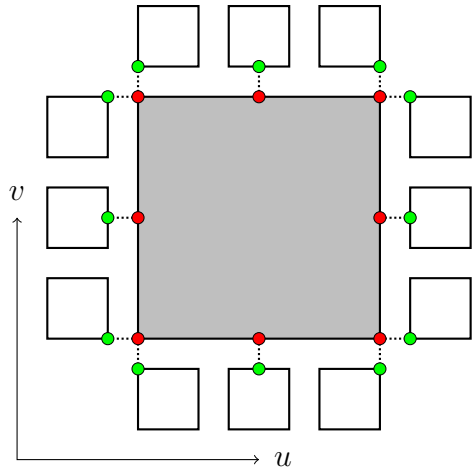
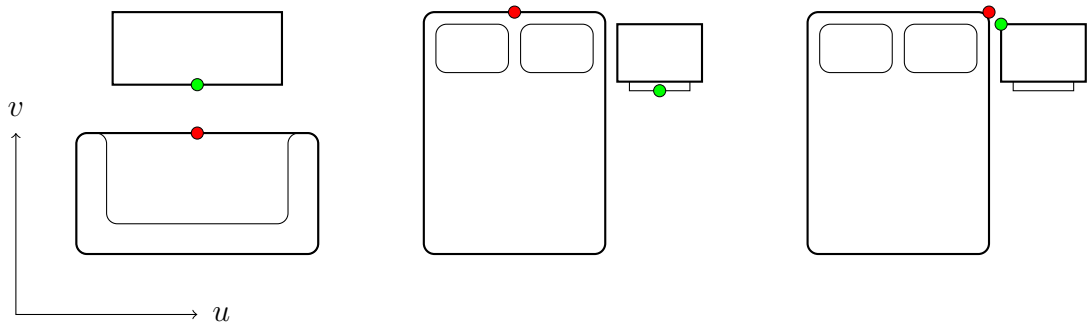


Figure B.2: Visualization of the 12 anchor points considered when computing pairwise object co-occurrence statistics.



(a) couch and coffee table, front-middle anchor

(b) bed and nightstand, front-middle anchor

(c) bed and nightstand, front-corner anchor

Figure B.3: Visualization of anchor points used for computing pairwise object co-occurrence statistics.

When mining through each object co-occurrence, we do not record distances to objects which are on the opposite side of the anchor point. For example, if a chair is positioned to the right of a table, we do not record its distance to the left edge anchor points on the table.

A common configuration of objects seen in home environments is a couch and coffee table. Typically, coffee tables are centered in front of couches with around 1.5-2ft of space in between them, leaving enough room for a person to sit comfortably. The histograms in Figure B.4 show the distribution in distances between a couch and a coffee table when anchored to the front center of a couch, as depicted in Figure B.3(a). Note that the distribution of u-distances measuring how centered tables are relative to the middle of a couch has

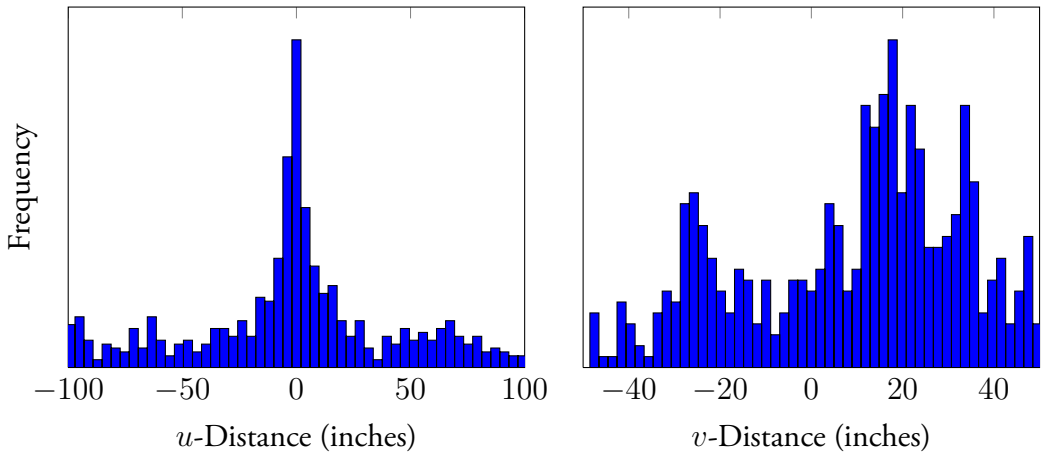


Figure B.4: Distribution of distances between couches and coffee tables, using the front middle of the couch as an anchor, as depicted in Figure B.3(a).

a strong mode at 0, indicting that coffee tables are most commonly aligned to the midpoint of a couch. The v-distance has a mode at 18in, the most frequent distance between couches and coffee tables. The negative v-distances seen in the histogram on the right occur in two situations. First, tables are sometimes positioned besides couches, not in front of them. This not only explains the negative v-distances, but also the large variance in u-distances, since the side tables will be far from their couch midpoint anchors. Another situation in which v-distances can be negative is for "L-shaped" couches. These objects are non-convex, and tables are typically positioned within their bounding boxes.

Although using the front midpoint of a couch as an anchor was quite informative for modeling the relative locations of tables, this anchor point may be less informative for other object pairs. For example, Figure B.5 shows distribution of nightstand positions relative to the front middle anchor of a bed, as depicted in Figure B.3(b). The bimodal distribution of u-distances indicates that unlike a couch and table configuration, nightstands are almost never centered in front of a bed; rather, they are positioned to either side. Because beds come in different sizes, the positions of nightstands relative to the beds' midpoints has substantial variance, making it difficult accurately predict the most likely locations of the objects relative to each other. However, if we use the front right corner of a bed as the anchor, as depicted in Figure B.3(c), the distribution of relative nightstand locations can be modeled with tight Gaussians, as shown in the histograms in Figure B.6. Here, there are strong peaks at 0 for both the u-distance and v-distance, indicating that nightstands are most frequently positioned such that their corners abut the corners of beds. We compute the distributions of

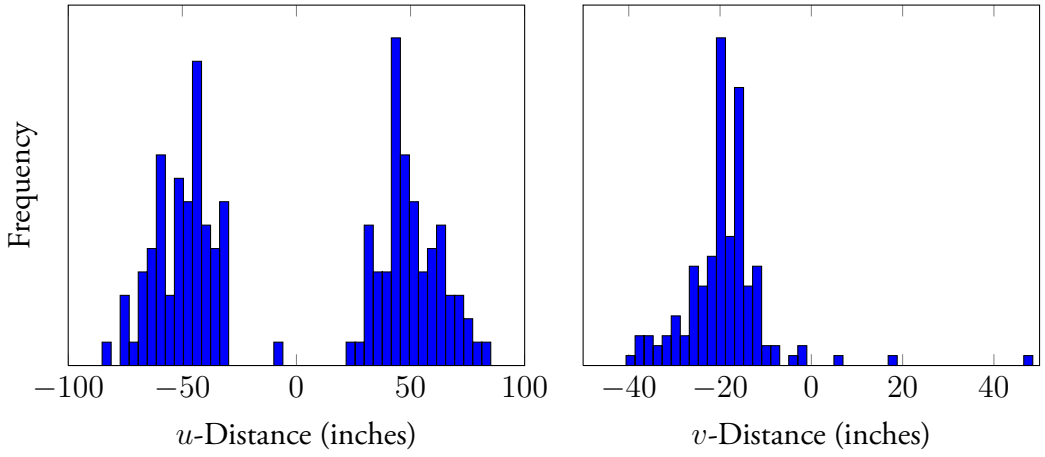


Figure B.5: Distribution of distances between beds and nightstands, using the front middle of the bed as an anchor, as depicted in Figure B.3(b).

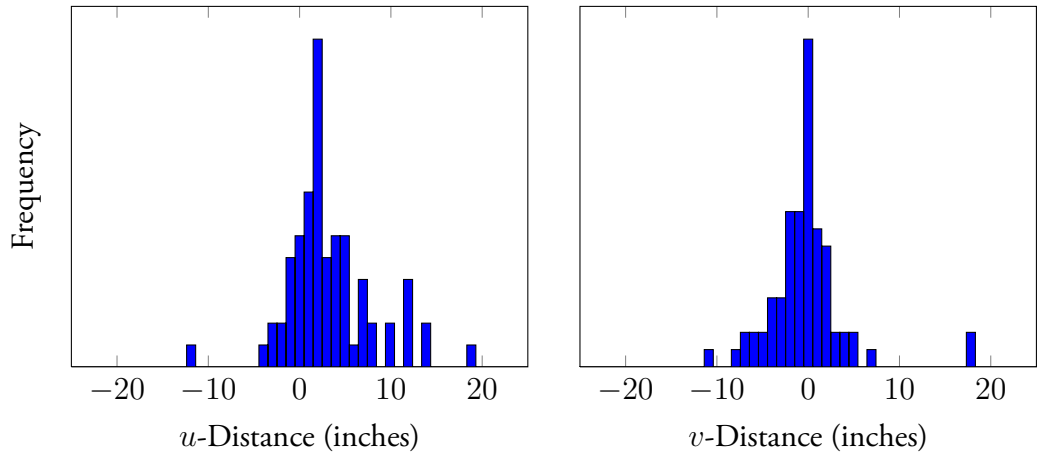


Figure B.6: Distribution of distances between beds and nightstands, using the front right corner of the bed as an anchor, as depicted in Figure B.3(c).

distances using all 12 anchor points for each object pair, and select the ones with the lowest variance. This allows us to discard uninformative relationships and keep only those which accurately model the spatial co-occurrences between objects.

These types of co-occurrence statistics have many practical applications. For example, researchers have demonstrated the utility of object co-occurrence statistics for a wide variety of tasks ranging from context-based model search [Fisher and Hanrahan, 2010] and automated interior design [Merrell et al., 2011] to scene parsing [Choi et al., 2013]. We

incorporate these statistics in Chapter VI to predict what objects are most likely to appear in a scene, as well as their most probable position and orientation relative to other objects. Future work may incorporate these co-occurrence statistics into the geometry refinement process, to preserve these relationships while refining the positions of objects in a scene.

APPENDIX C

Monocular Autocalibration Error Analysis

In this appendix, we analyze the various sources of error for our system. Understanding what causes our geometry estimation system to fail allows us to identify what stages of the pipeline need improvement. A fundamental limitation of our approach to geometry estimation is the reliance on a single vanishing point estimate. This hard-decision can result in unrecoverable errors which affect room layout estimation and each additional stage of our pipeline which builds upon our initial vanishing point estimate.

We now analyze the accuracy of the vanishing point estimation algorithm we use. For each annotated image in our dataset, we have hand-labeled vanishing point locations. Since the locations of vanishing points can be close to infinity, computing the absolute error with respect to pixel locations is not numerically stable nor does it truly reflect the error of our overall calibration process. To compare predicted vanishing point locations to the annotated vanishing point locations, we analyze the rotation of camera relative to the three orthogonal axis defined by the vanishing points.

Our intrinsic camera matrix is computed using the following orthogonality constraint on vanishing points:

$$V_1'(K^{-\top}K^{-1})V_2 = 0 \tag{C.1}$$

$$V_2'(K^{-\top}K^{-1})V_3 = 0 \tag{C.2}$$

$$V_3'(K^{-\top}K^{-1})V_1 = 0. \tag{C.3}$$

The rotation matrix representing the rotation of the camera relative the principal axis of the scene is then computed as follows:

$$R = \left(\frac{K^{-1}V_1}{\|K^{-1}V_1\|}, \frac{K^{-1}V_2}{\|K^{-1}V_2\|}, \frac{K^{-1}V_3}{\|K^{-1}V_3\|} \right). \tag{C.4}$$

To compare our estimated rotation matrix to the ground-truth rotation (from annotation), we compute the geodesic distance on the 3D manifold of rotation matrices as follows:

$$\Theta = \frac{1}{\sqrt{2}} \| \log(R_1^\top R_2) \|_F. \tag{C.5}$$

Figure C.1 shows the cumulative rotational error based on vanishing point estimation. Approximately 40% of scenes had vanishing point estimates which yielded a geodesic distance less than 5° from the human-annotated rotations. Approximately 75% of scenes were within 10° of the annotated locations. For a subset of images we had two annotators label vanishing points. This allows us to measure the subjectivity of the annotation task. The standard deviation of the geodesic distance between the rotations from pairs of human annotated vanishing points was 3%.

We also perform a similar error analysis on our focal length estimates. Figure C.2 shows the results of this experiment, comparing autocalibration focal length estimates with focal lengths from human annotations. This analysis showed that we are unable to accurately estimate the scale of the scene (based on focal length) for a significant percentage of images. Approximately 40% of images had focal length estimates that were 20% or more off of the ground-truth focal lengths. This suggests that searching over possible focal lengths may boost the performance of our system.

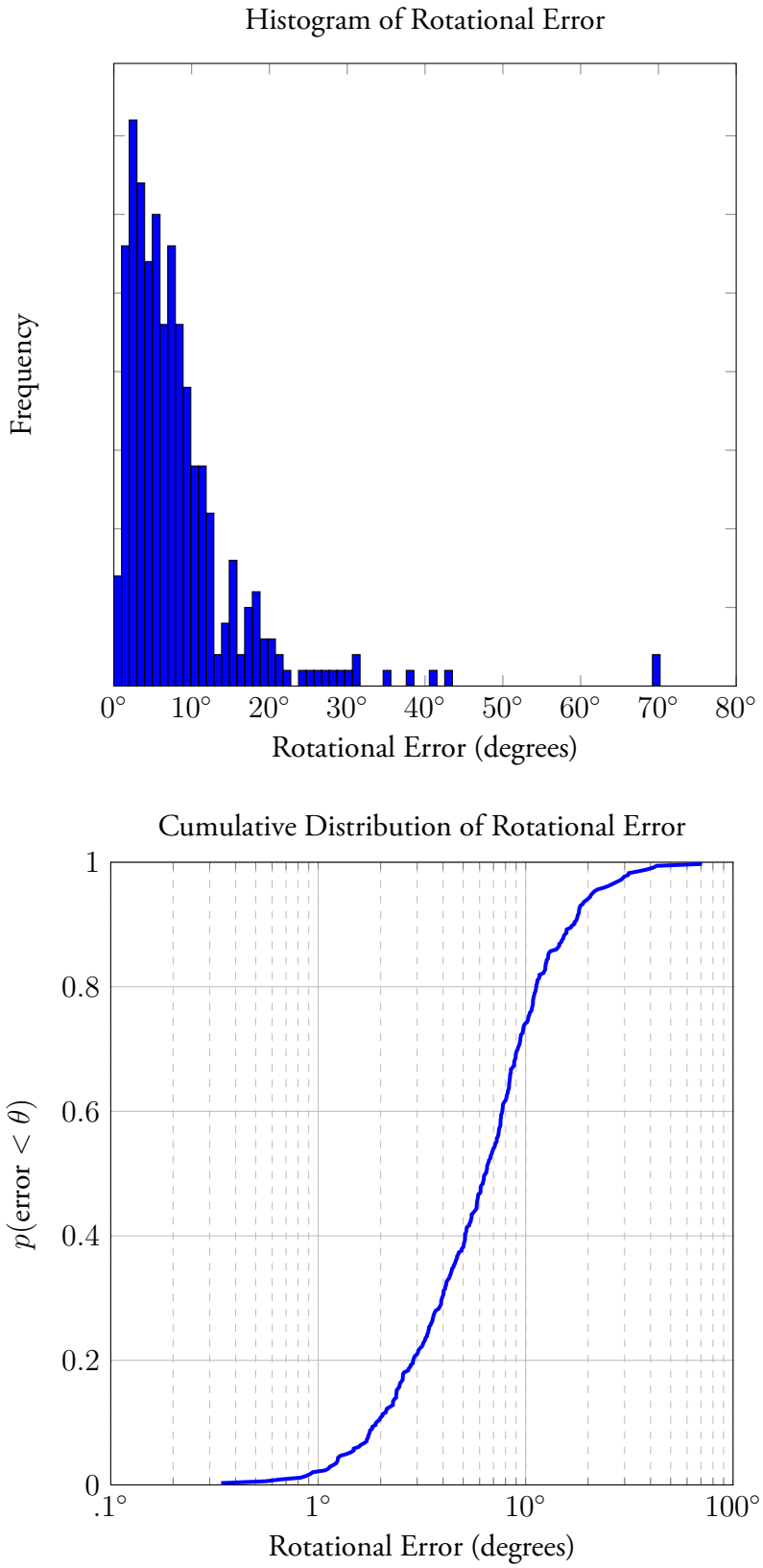


Figure C.1: Analysis of rotation error based on camera autocalibration.

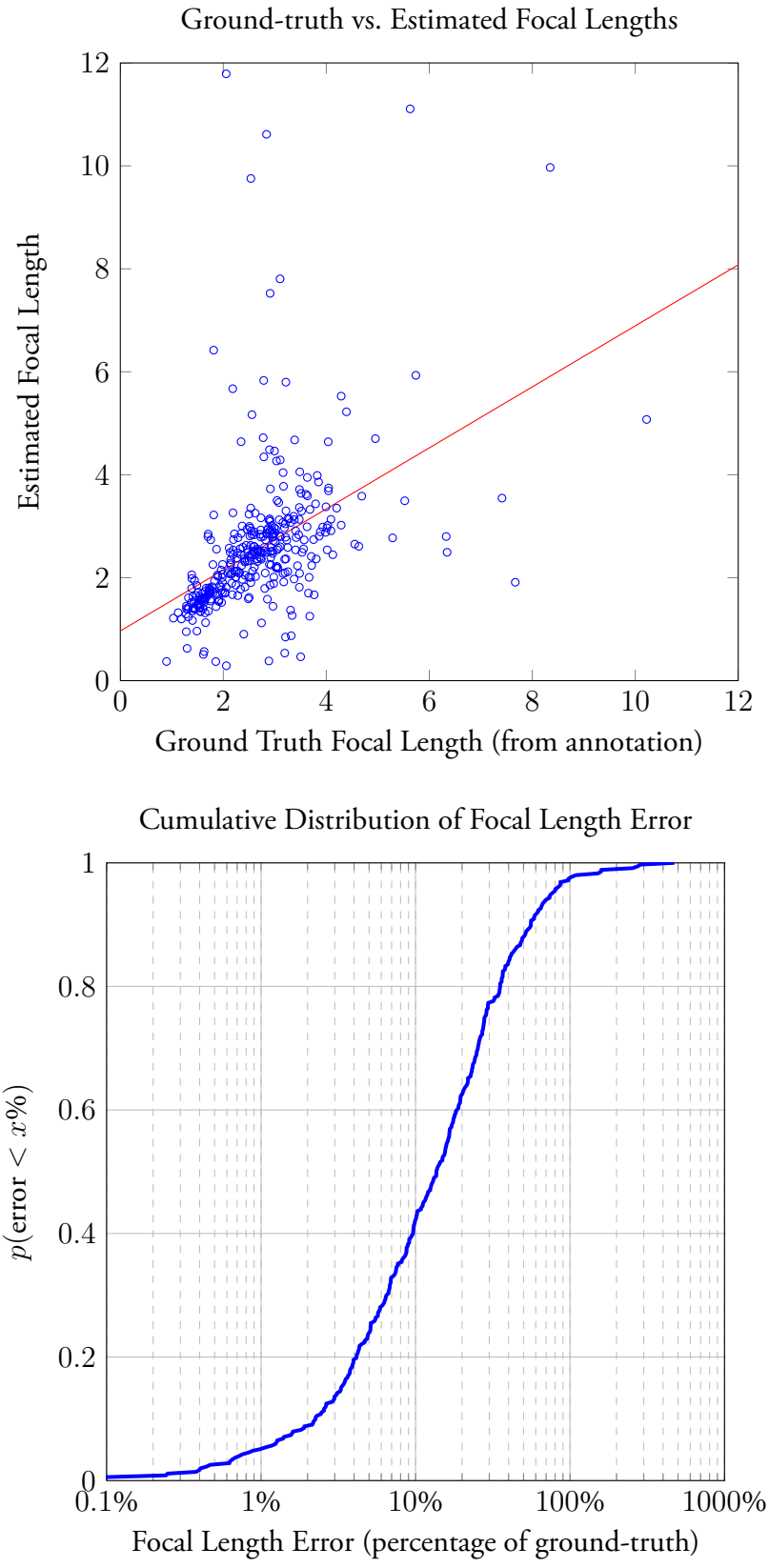


Figure C.2: Error analysis of focal length estimation.

APPENDIX D

3D Model Library Size Analysis

In this appendix, we explore issues with the size of our 3D model library. Vision researchers such as [Torralba et al., 2008, Hays and Efros, 2007] have demonstrated the importance of large datasets for data-driven approaches. The performance of our preliminary scene matching system could potentially be limited by the relatively small number of 3D models in our library. To estimate the effect of dataset size on the performance of our algorithm, we ran experiments restricting the number of scene hypotheses considered.

Figure D.1 shows how our performance increases as the size of the 3D model library grows. Figure D.2 illustrates the same trend; however, an oracle is used to automatically select the best-performing geometry hypothesis. This removes the effect of poor hypothesis ranking, and essentially computes an upper-bound on performance. Note that each of these plots is increasing sublogarithmically, indicating that our library of 3D models is approaching saturation in size – extrapolating forward indicates that it may take orders of magnitude more data to achieve a substantial gain in performance.

This trend may be due to the fact that we incorporate 3D data into our model of a scene, this makes our approach inherently invariant to rotations. Traditional appearance-based recognition system are trained on 2D images and require training examples from all unique viewpoints; however, our approach utilizes 3D models capturing all possible rotations. Moreover, unlike many environments which have a high variability in object arrangements, bedrooms and living rooms are very well-structured. This trend is present in both the 3D models we collect from 3D Warehouse [Trimble Inc., 2012] (See Section 3.6) as well as the images randomly selected from the SUN database [Xiao et al., 2010] for our dataset. Thus, there are diminishing returns with each 3D model we add to our library.

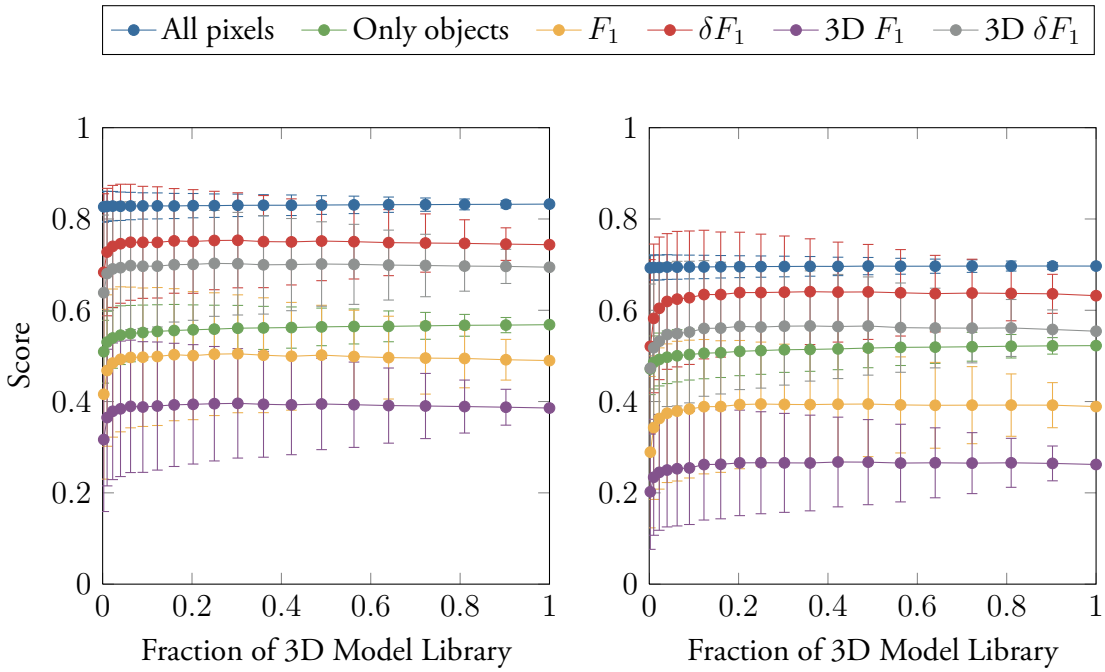


Figure D.1: Analysis of dataset size using our learned feature weighting: (left) incorporating ground-truth room layouts from annotation, (right) fully-automatic approach. Note: results are computed with our preliminary scene matching approach presented in Chapter II.

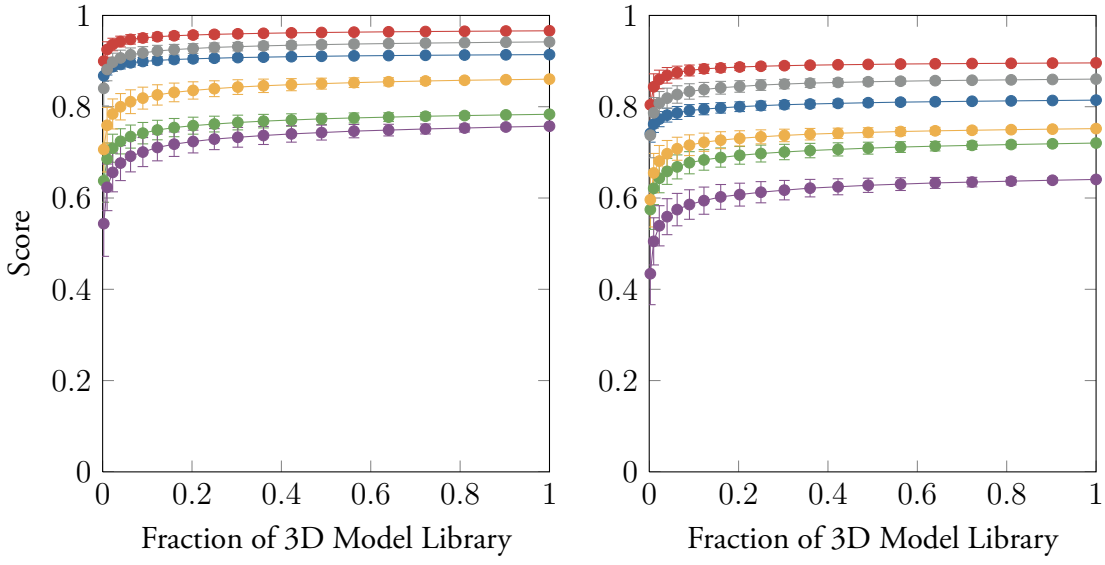


Figure D.2: Analysis of dataset size using an oracle to select the highest-scoring geometry hypothesis: (left) incorporating ground-truth room layouts from annotation, (right) fully-automatic approach.

WORKS CITED

[Arbelaez et al., 2011] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(5):898–916.

[Bao et al., 2010] Bao, S. Y., Sun, M., and Savarese, S. (2010). Toward coherent object detection and scene layout understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):509–522.

[Brooks, 1981] Brooks, R. (1981). Symbolic reasoning among 3D models and 2D images. In *Artificial Intelligence*, volume 17.

[Choi et al., 2013] Choi, W., Pantofaru, C., and Savarese, S. (2013). Understanding indoor scenes using 3d geometric phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Cutzu and Tarr, 1997] Cutzu, F. and Tarr, M. (1997). The representation of three-dimensional object similarity in human vision. In *SPIE Proceedings from Electronic Imaging: Human Vision and Electronic Imaging II, 3016*.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Debevec, 1998] Debevec, P. (1998). Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Desai et al., 2009] Desai, C., Ramanan, D., and Fowlkes, C. (2009). Discriminative models for multi-class object layout. In *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Dey et al., 2012] Dey, D., Liu, T. Y., Sofman, B., and Bagnell, J. A. D. (2012). Efficient optimization of control libraries. In *26th Conference of Association for Advancement of Artificial Intelligence*.

[Douze et al., 2009] Douze, M., Jégou, H., Sandhawalia, H., Amsaleg, L., and Schmid, C. (2009). Evaluation of gist descriptors for web-scale image search. In *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR)*.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338.

[Felzenszwalb et al., 2008] Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Felzenszwalb et al., 2010a] Felzenszwalb, P. F., Girshick, R. B., and McAllester, D. (2010a). Discriminatively trained deformable part models, release 4. `http://people.cs.uchicago.edu/~pff/latent-release4/`.

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

[Fisher and Hanrahan, 2010] Fisher, M. and Hanrahan, P. (2010). Context-based search for 3d models. *ACM Transactions on Graphics*, 29:182:1–182:10.

[Fisher et al., 2011] Fisher, M., Savva, M., and Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. *ACM Transactions on Graphics*, 30:34:1–34:12.

[Fouhey et al., 2013] Fouhey, D., Gupta, A., and Hebert, M. (2013). Data-driven 3d primatives. In *In submission to the Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Fouhey et al., 2012] Fouhey, D. F., Delaitre, V., Gupta, A., Efros, A. A., Laptev, I., and Sivic, J. (2012). People watching: Human actions as a cue for single-view geometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Girshick et al., 2012] Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. (2012). Discriminatively trained deformable part models, release 5. `http://people.cs.uchicago.edu/~rbg/latent-release5/`.

[Google Inc., 2000] Google Inc. (2000). Google SketchUp. `http://sketchup.google.com/`.

[Grabner et al., 2011] Grabner, H., Gall, J., and Gool, L. J. V. (2011). What makes a chair a chair? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1536.

[Grimson et al., 1990] Grimson, W., Lozano-Pérez, T., and Huttenlocher, D. (1990). *Object recognition by computer*. MIT Press.

[Grosse et al., 2009] Grosse, R., Johnson, M. K., Adelson, E. H., and Freeman, W. T. (2009). Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2335–2342.

[Guo et al., 2011] Guo, R., Dai, Q., and Hoiem, D. (2011). Single-image shadow detection and removal using paired regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Guo and Hoiem, 2013] Guo, R. and Hoiem, D. (2013). Beyond the line of sight: labeling the underlying surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Gupta et al., 2010] Gupta, A., Efros, A. A., and Hebert, M. (2010). Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Gupta et al., 2011] Gupta, A., Satkin, S., Efros, A. A., and Hebert, M. (2011). From 3d scene geometry to human workspace. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Hays and Efros, 2007] Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3).

[Hays and Efros, 2008] Hays, J. and Efros, A. A. (2008). im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Hedau et al., 2009] Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Hedau et al., 2010] Hedau, V., Hoiem, D., and Forsyth, D. (2010). Thinking inside the box: Using appearance models and context based on room geometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Hedau et al., 2012] Hedau, V., Hoiem, D., and Forsyth, D. (2012). Recovering free space of indoor scenes from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Herbrich et al., 1999] Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support vector learning for ordinal regression. In *Proceedings of the International Conference on Artificial Neural Networks (ANN)*, volume 1, pages 97 –102.

[Hödlmoser and Micusik, 2013] Hödlmoser, M. and Micusik, B. (2013). Surface layout estimation using multiple segmentation methods and 3d reasoning. In *Pattern Recognition and Image Analysis*, volume 7887, pages 41–49.

[Hoiem et al., 2006] Hoiem, D., Efros, A. A., and Hebert, M. (2006). Putting objects in perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2137 – 2144.

[Hoiem et al., 2007a] Hoiem, D., Efros, A. A., and Hebert, M. (2007a). Recovering surface layout from an image. In *International Journal of Computer Vision (IJCV)*, volume 75.

[Hoiem et al., 2008] Hoiem, D., Efros, A. A., and Hebert, M. (2008). Closing the loop on scene interpretation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Hoiem et al., 2007b] Hoiem, D., Stein, A., Efros, A., and Hebert, M. (2007b). Recovering occlusion boundaries from a single image. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[IKEA, 2013] IKEA (2013). IKEA Catalog.

[Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

[Karsch et al., 2011] Karsch, K., Hedau, V., Forsyth, D., and Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. ACM.

[Karsch et al., 2012] Karsch, K., Liu, C., and Kang, S. B. (2012). Depth extraction from video using non-parametric sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Kepler, 1602] Kepler, J. (1979/1602). *On the More Certain Fundamentals of Astrology*. Translated from Latin by M.A. Rossi.

[Lafferty, 2002] Lafferty, J. (2002). Machine learning and language processing. `http://galton.uchicago.edu/~lafferty/pdf/lti-ic02.pdf`.

[Lai and Fox, 2009] Lai, K. and Fox, D. (2009). 3D laser scan classification using web data and domain adaptation. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

[Lalonde et al., 2007] Lalonde, J.-F., Hoiem, D., Efros, A. A., Rother, C., Winn, J., and Criminisi, A. (2007). Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3):3.

[Latecki et al., 2000] Latecki, L. J., Lakämper, R., and Eckhardt, U. (2000). Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Lee et al., 2010] Lee, D., Gupta, A., Hebert, M., and Kanade., T. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*.

[Lee et al., 2009] Lee, D., Hebert, M., and Kanade., T. (2009). Geometric reasoning for single image structure recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Levin et al., 2008] Levin, A., Rav-Acha, A., and Lischinski, D. (2008). Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(10):1699–1712.

[Li et al., 2010] Li, L.-J., Su, H., Xing, E. P., and Fei-Fei, L. (2010). Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Neural Information Processing Systems (NIPS)*.

[Liu et al., 2008] Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. (2008). SIFT Flow: dense correspondence across different scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Liu et al., 2010] Liu, M.-Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. (2010). Fast directional chamfer matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Longuet-Higgins, 1981] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293.

[Lowe, 1987] Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. In *Artificial Intelligence*, volume 31.

[Matikainen et al., 2012] Matikainen, P., Sukthankar, R., and Hebert, M. (2012). Model recommendation for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Merrell et al., 2011] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. *ACM Transactions on Graphics (SIGGRAPH 2011)*.

[Microsoft Corporation, 2010] Microsoft Corporation (2010). Kinect for Xbox 360.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630.

[Munoz et al., 2010] Munoz, D., Bagnell, J. A., and Hebert, M. (2010). Stacked hierarchical labeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Nemhauser and Wolsey, 1978] Nemhauser, G. and Wolsey, L. A. (1978). An Analysis of Approximations for Maximizing Submodular Set Functions - I. *Mathematical Programming*, 14:265–294.

[Oliva and Torralba, 2001] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision (IJCV)*, 42:145–175.

[Oliva and Torralba, 2006] Oliva, A. and Torralba, A. (2006). Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*.

[Pearson, 1896] Pearson, K. (1896). Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:253–318.

[Pero et al., 2012] Pero, L. D., Bowdish, J., Fried, D., Kermgard, B., Hartley, E., and Barnard, K. (2012). Bayesian geometric modeling of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Pero et al., 2013] Pero, L. D., Bowdish, J., Hartley, E., Kermgard, B., and Barnard, K. (2013). Understanding bayesian rooms using composite 3d object models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Pero et al., 2011] Pero, L. D., Guan, J., Brau, E., Schlecht, J., and Barnard, K. (2011). Sampling bedrooms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Polantis, 2010] Polantis (2010). Polantis 3D Catalog. `http://www.polantis.com/ikea`.

[Ramalingam et al., 2013] Ramalingam, S., Pillai, J. K., Jain, A., and Taguch, Y. (2013). Manhattan junction catalogue for spatial reasoning of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Rosch et al., 1976] Rosch, E., Mervis, C. B., Gray, W. D., M, D., and Boyes-braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*.

[Rubin, 1915] Rubin, E. (1958/1915). Figure-ground perception. In *Readings in Perception*. Translated from German by M. Wertheimer.

[Satkin et al., 2012a] Satkin, S., Lin, J., and Hebert, M. (2012a). CMU 3D-Annotated Scene Database. `http://cmu.satkin.com/bmvc2012/`.

[Satkin et al., 2012b] Satkin, S., Lin, J., and Hebert, M. (2012b). Data-driven scene understanding from 3D models. In *Proceedings of the 23rd British Machine Vision Conference (BMVC)*.

[Saxena et al., 2009] Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(5):824–840.

[Schwing and Urtasun, 2012] Schwing, A. G. and Urtasun, R. (2012). Efficient Exact Inference for 3D Indoor Scene Understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Shao et al., 2012] Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., and Guo, B. (2012). An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics*, 31(6):136:1–136:11.

[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Sing and Košecká, 2013] Sing, G. and Košecká, J. (2013). Nonparametric scene parsing with adaptive feature relevance and semantic context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Spain and Perona, 2011] Spain, M. and Perona, P. (2011). Measuring and predicting object importance. *International Journal of Computer Vision (IJCV)*, 91(1):59–76.

[Sugihara, 1986] Sugihara, K. (1986). *Machine Interpretation of Line Drawings*. MIT Press.

[Tighe and Lazebnik, 2010] Tighe, J. and Lazebnik, S. (2010). Superparsing: scalable nonparametric image parsing with superpixels. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(11):1958–1970.

[Torralba et al., 2010] Torralba, A., Russell, B. C., and Yuen, J. (2010). Labelme: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484.

[Trimble Inc., 2012] Trimble Inc. (2012). Trimble 3D Warehouse. `http://sketchup.google.com/3dwarehouse`.

[Vondrick et al., 2013] Vondrick, C., Khosla, A., Malisiewicz, T., and Torralba, A. (2013). Inverting and visualizing features for object detection. In *MIT Technical Report*.

[Wang et al., 2010] Wang, H., Gould, S., and Koller, D. (2010). Discriminative learning with latent variables for cluttered indoor scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Xiao et al., 2010] Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Yu et al., 2008] Yu, S., Zhang, H., and Malik., J. (2008). Inferring spatial layout from a single image via depth-ordered grouping. In *Proceedings of the IEEE Workshop on Perceptual Organization in Computer Vision (CVPR-POCV)*.

[Zhao and Zhu, 2013] Zhao, Y. and Zhu, S.-C. (2013). Scene parsing by integrating function, geometry and appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.