TOP-DOWN BAYESIAN MODELING AND INFERENCE FOR INDOOR SCENES

by

Luca Del Pero

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2013

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Luca Del Pero, titled Top-down Bayesian modeling and inference for indoor scenes, and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Kobus Barnard	Date: 17 May 2013
Paul Cohen	Date: 17 May 2013
Alon Efrat	Date: 17 May 2013
Clayton Morrison	Date: 17 May 2013

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director: Kobus Barnard	Date: 17 May 2013

STATEMENT BY AUTHOR

SIGNED: Luca Del Pero

ACKNOWLEDGEMENTS

I will add the acknowledgments as a last thing.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

People can understand the content of an image without effort. We can easily identify the objects in it, and figure out where they are in the 3D world. Automating these abilities is critical for many applications, like robotics, autonomous driving and surveillance. Unfortunately, despite recent advancements, fully automated vision systems for image understanding do not exist. In this work, we present progress restricted to the domain of images of indoor scenes, such as bedrooms and kitchens. These environments typically have the "Manhattan" property that most surfaces are parallel to three principal ones. Further, the 3D geometry of a room and the objects within it can be approximated with simple geometric primitives, such as 3D blocks. Our goal is to reconstruct the 3D geometry of an indoor environment while also understanding its semantic meaning, by identifying the objects in the scene, such as beds and couches

We separately model the 3D geometry, the camera, and an image likelihood, to provide a generative statistical model for image data. Our representation captures the rich structure of an indoor scene, by explicitly modeling the contextual relationships among its elements, such as the typical size of objects and their arrangement in the room, and simple physical constraints, such as 3D objects do not intersect. This ensures that the predicted image interpretation will be globally coherent geometrically and semantically, which allows tackling the ambiguities caused by projecting a 3D scene onto an image, such as occlusions and foreshortening.

We fit this model to images using MCMC sampling. Our inference method combines bottom-up evidence from the data and top-down knowledge from the 3D world, in order to explore the vast output space efficiently. Comprehensive evaluation confirms our intuition that global inference of the entire scene is more effective than estimating its individual elements independently. Further, our experiments show

that our approach is competitive and often exceeds the results of state-of-the-art methods.

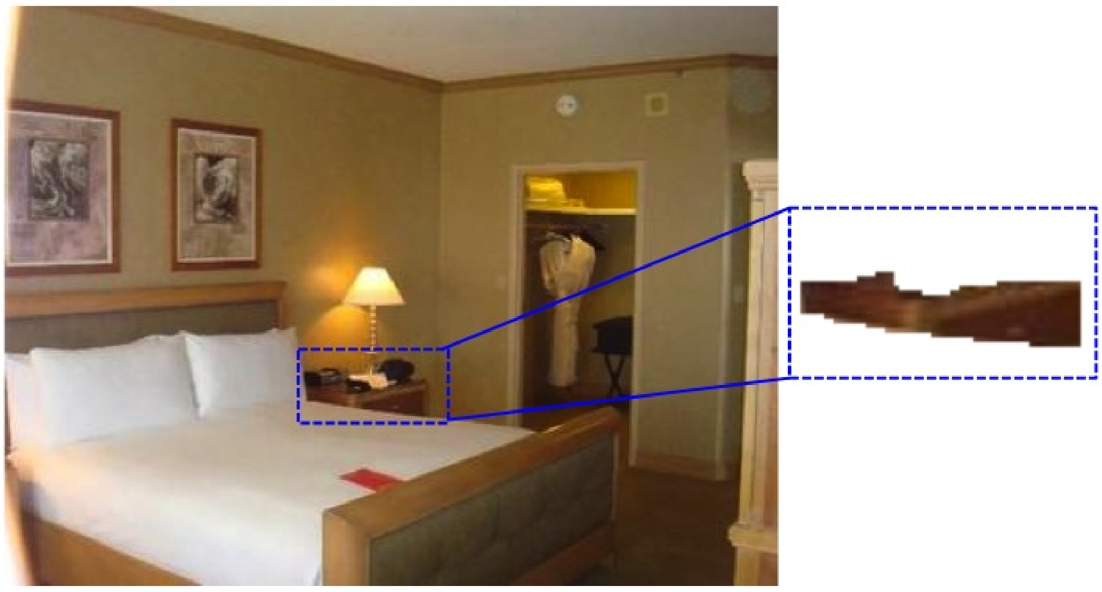

Figure 1: Understanding an image requires global reasoning. Most people have no trouble identifying the nightstand in the left image, but this would be challenging if we considered only local evidence, highlighted on the right. This ambiguity is solved when we look at the entire image, as the bed and the reading lamp provide the context required to identify the nightstand. From this flat image, we can also understand the 3D scene, for example it is easy to infer that a possible path to the door of the closet goes around the bed. This task requires a global interpretation of the image, as we need to consider the interdependencies among the objects and the camera in 3D. Automatically recovering a global interpretation of such an image is precisely our goal. Given a single image of an indoor environment, like the one above, we want to simultaneously recover the objects in the 3D scene, their interdependencies, and the camera. Our intuition is that pursuing a global solution is more powerful than finding each element individually, as illustrated by the two examples above.

CHAPTER 1

Introduction

Understanding the content of images is an effortless task for most people. We can in fact quickly identify most of the objects in a picture, or reason about the 3D structure of the scene shown in the flat image. For example, it is easy to recognize the bed in Fig. 1, and to conclude that it is closer to the viewer than the door of the closet. Further, we can typically infer the interplay among elements in the scene, for instance we understand that most of the nightstand is hidden because the bed is in front of it.

Being aware of these interactions among objects in the context of the overall scene is indeed very important, as there is strong evidence from psychology and cognitive science that context plays a central role in scene understanding. Interpreting the group of pixels highlighted in Fig. 1 as a nightstand would be in fact very challenging, if we were to consider them in isolation. However, we can easily recognize this object thanks to its context, since it sits alongside the bed, and it supports items typically found on a nightstand, such as the reading lamp.

Despite major advancements over the last decades, current computer vision systems cannot yet achieve this deep level of image understanding. While there has been much progress towards solving individual tasks, like object recognition, 3D reconstruction, or pose estimation to name a few, a fully functioning vision system is still beyond reach. Such a system would need to identify all the objects in an image, and at the same time capture their interplay in the context of the overall scene. This process would also need to address the additional complexities introduced by the imaging process. In fact, the contextual relationships among scene elements are more meaningful in the 3D world than on the 2D image plane where they have been projected by a camera, whose parameters are not known.

The tight interconnections among objects in the scene and the camera inspired much recent work aiming at solving these problems jointly, with the goal of providing a global interpretation of an image. Of particular interest is the work of Hoiem et al. (2006), which proposed a framework for placing local object detection in the context of the 3D scene, by jointly estimating objects, scene geometry and camera parameters. In their work, they demonstrated the cyclical nature of the problem, by showing that object hypotheses can refine the estimated scene geometry and vice-versa.

In this dissertation we explore the idea of providing a global interpretation of an image in the domain of indoor scenes, such as bedrooms and living rooms. Such environments have a very rich and well defined 3D structure. For example, most surfaces in the scene are aligned with three principal orthogonal directions. Further, the set of objects found indoor is typically restricted to pieces of furniture with a relatively simple geometric structure, like beds and tables, which are often arranged in recurring patterns, such as a set of chairs around a table, or a footstool in front of a couch. In this setting, a global approach to understanding an image is potentially very powerful, given the rich and informative relationships among scene elements, which are more meaningful in the context of the entire scene.

We thus want to provide a comprehensive parsing of an indoor image that is globally consistent in 3D both geometrically and semantically, by taking into account the interdependencies of objects, camera, and our prior knowledge of indoor geometry. More specifically, given a monocular image, our goal is to jointly infer: 1) the parameters of the camera; 2) the **scene layout**, defined by the set of orthogonal surfaces enclosing the environment (walls, floor and ceiling); 3) the number of objects in the scene, which is not known a priori; and 4) the size, position, and shape of all the objects, as well as their identity (e.g. bed, couch, door, etc.).

We propose to do so using a Bayesian generative approach relying on a strong 3D model, where we separately model the objects, the camera, and an image likelihood. As summarized in Fig. 1.1, we assume that image features are generated statistically by projecting the objects in the 3D scene with the camera. The Bayesian framework is indeed very powerful in this scope, as it separates the modeling and the inference, thus allowing us to encode the complex indoor relationships in a principled way in the modeling phase. During inference, we search the parameter space by sampling from the Bayesian posterior distribution, which includes a likelihood component to compare the features predicted by the model with the image data, and a prior distribution to constrain 3D hypotheses to realistic configurations (Fig. 1.1).

We emphasize that the first key aspect of this work is that we solve for all the unknowns **jointly**, by exploiting their interdependencies to our advantage, as opposed to seeing them as an additional complexity in the inference process. Having a global understanding of the scene also helps address ambiguities arising from the imaging process, such as occlusions and clutter, in a top-down fashion. For example, occlusions can be explained in the context of the entire scene by considering the 3D position of objects with respect to the camera. Similarly, chairs and tables are hard to localize individually, because they heavily occlude each other, and this is why we suggest finding them together. Bayesian inference is again very suitable towards this goal, as it enables joint inference over model parameters without committing to partial solutions.

As a second important point of our method, we rely on a strong 3D model, where the statistical variations of the camera and those of the objects are modeled separately. Importantly, object variations are modeled in 3D, where the geometry is strongly constrained. Instead, 2D representations are more ambiguous, since changes in 2D appearance could be due to both variations in the view and in the 3D objects. For example, corners of tables are usually orthogonal in 3D, but their projections on the image are not right angles, since the perspective transformation does not preserve orthogonality (Fig. 1.2). Similarly, a typical bed is a rectangular box, and is 2-3 feet high, but the shape and size of its 2D projection have a much larger variance. Separate modeling of the camera and of the 3D geometry helps avoiding this kind of ambiguities. Further, this separation is particularly powerful in the context of indoor scenes, where most objects have a similar orientation, and thus have to agree on a single camera configuration.

The proposed 3D model is characterized by a simple geometric structure, which

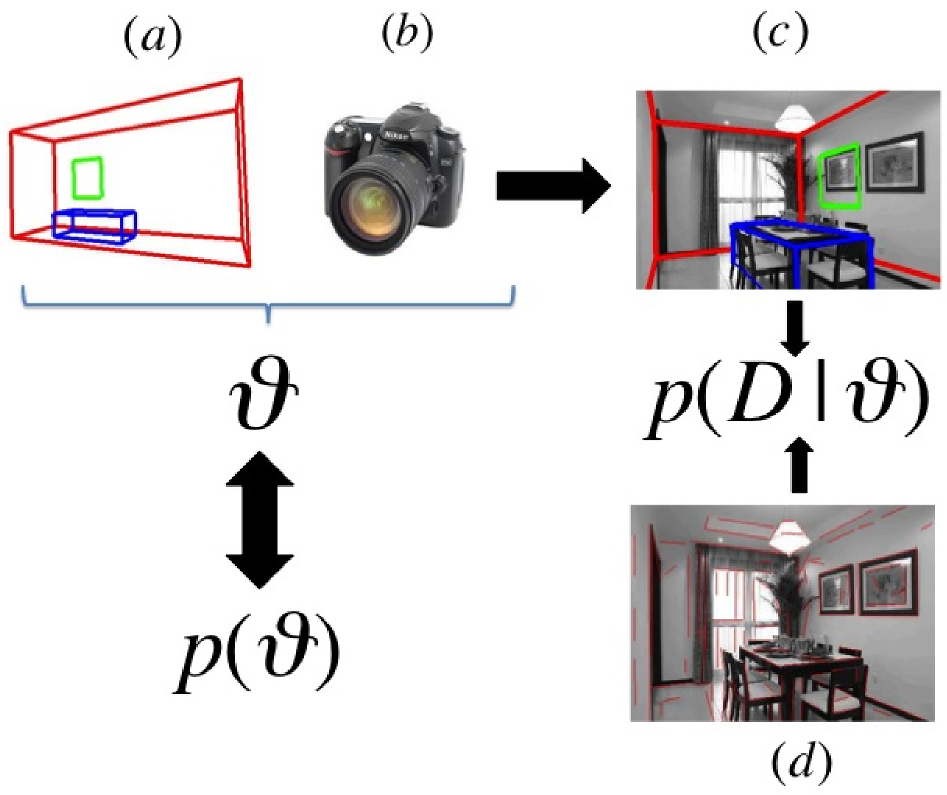

Figure 1.1: A visual summary of the proposed Bayesian approach. An instance of the model parameters θ comprise scene parameters (a) and camera parameters (b). Assuming the image data D is statistically generated by the projection of the objects in the scene, during inference we search for the model parameters maximizing the Bayesian posterior distribution $p(\theta|D)$, which according to Bayes' rule is proportional to the product of prior and likelihood. In our case, the likelihood function $p(D|\theta)$ measures the distance between the features generated by the model (c), which are edges in this example, and the features D detected on the image plane (d). The prior distribution $p(\theta)$ encourages realistic scene and camera configurations. While the likelihood is evaluated against the 2D detected features, $p(\theta)$ operates in 3D, by evaluating the hypothesized 3D objects against prior information, such as typical size and position of objects like beds, tables, and doors. This information can also be effectively used to inform the inference process, by hypothesizing realistic objects in a top-down fashion. In this process, the camera, the scene and the objects in it are fit jointly, to provide a parsing of the scene that is globally consistent and plausible in 3D. This helps addressing the ambiguities arising from the imaging process, such as objects occluding each other.

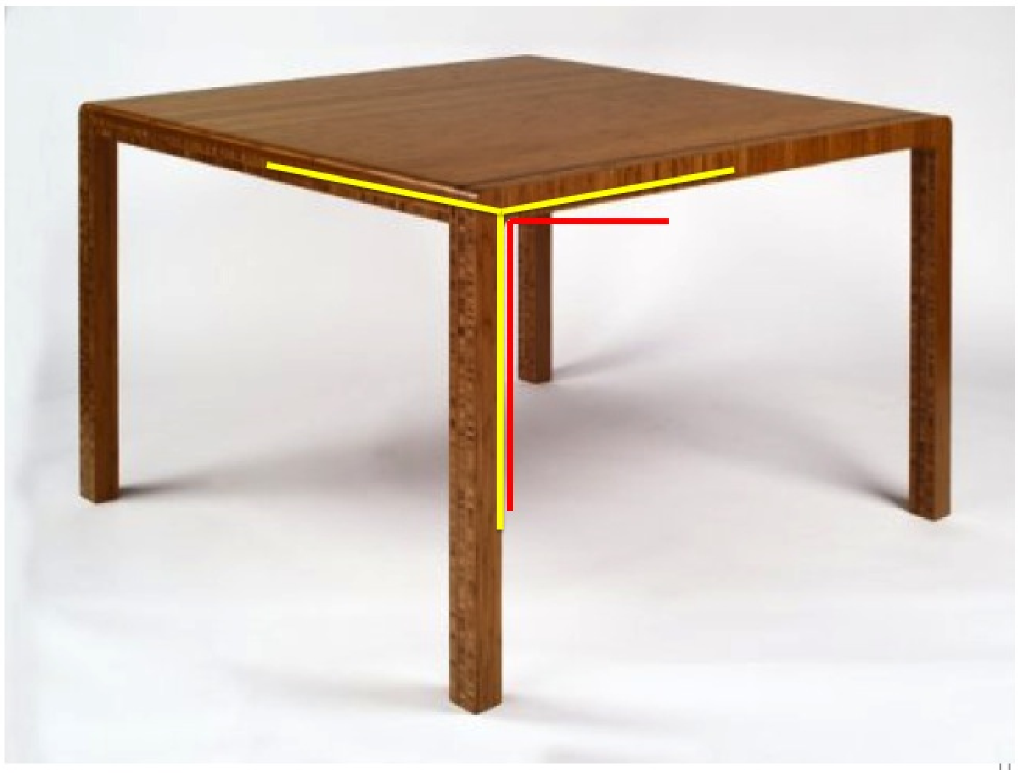

Figure 1.2: A perspective camera does not preserve angles. Here, the projected corner of a table (in yellow), which is orthogonal in 3D, is not a right angle on the image plane (compare it to the right angle denoted by the red lines). Notice how the four 2D angles corresponding to the table corners are all different. For this and other similar reasons, we model objects in the world using a three-dimensional representation, where shape and size have less variance than their projection in the image.

relies on a set of basic geometric primitives, such as 3D blocks and cylinders. We designed our system so that these parts are shared among different object categories, to exploit the similarities among different indoor object types. For instance, both couches and chairs have a backrest, and both tables and chairs are supported by legs. In general, sharing parts among objects is useful for both modeling and inference, and, as illustrated by these two examples, it also introduces a more explicit link between geometry and function. For example, all objects whose function is “sitting” naturally share a structure with seat and backrest.

A third main aspect of this work is the use of a strong semantic structure, imposed by introducing the notion of an object type (e.g. bed, table), which constrains the 3D position and size of an indoor object within the scene. For example, a bed always lies on the floor, and is usually found against a wall. Further, the semantic type imposes constraints on the object’s 3D shape, which, as just mentioned, is also related to an object’s function. For example, most chairs have legs, a seat and a backrest. The Bayesian framework allows encoding such constraints statistically, in the form of prior distributions that condition an object’s geometry, size and position on its type and on the scene layout.

Lastly, we further exploit the relationships between indoor scene elements to inform the inference process. Our core inference method relies on Reversible Jump MCMC (RJMCMC) sampling to handle the large and unknown number of parameters. In order to explore this vast output space more efficiently, we designed strategies to exploit the structure of indoor environments in a top-down fashion. For example, we predict objects at 3D locations that are likely given the current scene hypothesis, such as chairs around a table, or beds against a wall. Additionally, we combine these top-down proposals with bottom-up structures proposed from detected image features, such as line segments and image corners.

In the following sections of the introduction, we provide background on the field of scene understanding and give an overview of the related work. Then, in Chapter 2, we discuss the basic version of our approach for jointly inferring camera, scene layout and objects from a single image, where all objects are approximated by simple geometry. In Chapter 3 we introduce a semantic structure, by conditioning an object's size and position on its semantic type. Last, in Chapter 4 we introduce a more realistic 3D representation for objects, which uses more refined geometric models, and discuss its advantages.

1.1 Related work

In this section we discuss related work, organized according to how it relates to the main points of our work. We start with an overview of the literature on indoor scene understanding (1.1.1), and then consider object recognition and representation (1.1.2). Last, we discuss relevant work on statistical inference (1.1.3).

1.1.1 Indoor scene understanding

The area of indoor scene understanding received much interest over the past year. Most of the recent work in this field focused on **3D reconstruction of the indoor environment** from a monocular image, with the goal of recovering the position and orientation of the main surfaces defining the scene, such as walls, ceiling and floor. This information provides in fact crucial geometric context for higher level applications, such as object recognition and prediction of human activities. For example, Gupta et al. (2011) used extracted 3D information to identify locations where people can sit or lie, while Hedau et al. (2010) showed how knowledge of the 3D environment helps detecting pieces of furniture, such as beds. We now provide an overview of recent methods for indoor scene reconstruction, and of the most relevant applications.

Automatic 3D reconstruction from a monocular image is in general very challenging, because it requires estimating the camera pose and the focal length. Most indoor scenes, however, satisfy the Manhattan World assumption that most planes lie in one of three mutually orthogonal orientations. Edges from the Manhattan World generate three vanishing points on the image plane, and provide strong cues on the camera.

Early work by Coughlan and Yuille (1999, 2003) and Schindler and Dellaert (2004) used the Manhattan world assumption to recover the internal parameters of the camera and its relative rotation with respect to the scene from a single image. Both methods used image gradient information directly, while Rother (2002) and Kosecka and Zhang (2002) recovered a triplet of vanishing points corresponding to the three Manhattan directions by using straight line segments found in Manhattan structures. The camera parameters can then be estimated analytically from the triplet of vanishing points (see Sec. 2.3.2).

This set of methods based on line segments has been widely used in the domain of indoor images. For example, Kosecka and Zhang (2005) used the detected vanishing points to find rectilinear structures aligned with the Manhattan directions. Other relevant approaches for detecting rectangles in Manhattan images were developed by Micusk et al. (2008) and by Han and Zhu (2005), who arranged the hypothesized rectangles in regular patterns, such as grids and blocks, via a top-down grammar, which relates to other work on image parsing and grammar-based representations of images (Tu et al., 2005; Zhu and Mumford, 2006; Zhu et al., 2006). It is important to notice, however, that these approaches focus on image features directly, not on 3D world or 3D object characteristics.

Another related class of methods estimates the absolute 3D depth directly from image features, such as texture and color. Saxena et al. (2005) predicted depth maps for monocular images by using a Markov Random Field, trained on a set of images with associated ground truth depth maps. They then connected regions of the depth maps under weak assumptions of smoothness, connectivity and continuity (Saxena et al., 2008). Notice that the work of Saxena et al. (2008) is potentially more generic than the other work discussed so far, since it is not restricted to Manhattan world scenes. However, all these methods operate in the 2D image space, and there is no explicit attempt to recover the 3D structure of the scene.

The work of Yu et al. (2008) pushes the idea of reasoning about the 3D layout of the environment further. Specifically, they showed how to extract rectangular surfaces from a Manhattan World image, and then estimate the depth ordering of such planes using cues like coplanarity and convergence to common vanishing points. However, this method only provides depth cues of partial rectangular regions, which are not organized in a coherent 3D model of the scene.

The work of Delage et al. (2006) is instead one of the early attempts at recovering a full 3D model for an indoor scene. Their method uses appearance cues, such as color and edge orientation, to find the most likely floor-wall boundary in image space, which is then combined with the estimated camera pose to recover the 3D position and orientation of floor and walls. This work is closely relate to the approach by Hoiem et al. (2005b) for creating 3D "pop-up" models from monocular images of outdoor scenes. Lee et al. (2009) then showed how to find plausible 3D models of indoor scenes from detected line segments. Geometric constraints between groups of segments, such as convexity, and projective geometry are used to generate three dimensional hypotheses that are physically plausible. The hypothesis that best matches the entire collection of line segments is then used to create a full 3D model of the scene. However, both these method require the boundaries between floor, walls and ceiling to be visible in the image, and are prone to errors caused by occlusions and clutter, as they tend to fit walls to surfaces of objects, such as tables and beds.

A more recent class of methods uses a more compact representation, by assuming that most indoor scenes can be approximated with a parametric 3D box, which coarsely models the space of a room as if it were empty. This is often referred to as **room box**. Hedau et al. (2009) first proposed a method for estimating the room box from a single image that is robust to occlusions. They first estimate the camera from the vanishing points, and then iteratively localize clutter and refit the box. Clutter is identified using a probabilistic classifier based on image appearance, which is trained on images where clutter is labeled manually. Wang et al. (2010) extended this method by introducing latent variables to account for clutter, and their method does not require hand-labeling of the clutter in the training set.

In both these methods the interactions between clutter and room box are modeled in the image plane. Lee et al. (2010) showed how to model the interactions between occluding objects and the room box in 3D. An object in the scene is approximated as a parametric 3D box, which provides a reasonable bounding approximation for most indoor objects, such as tables and beds. Objects cannot intersect each other, they have to be contained in the free space defined by the room box. Lee et al. (2010) found that joint inference of the room box and the objects in it improves on estimating the former, since it allows accounting for occlusions.

The methods for estimating the room box discussed so far (Hedau et al., 2009; Wang et al., 2010; Lee et al., 2010) use structured prediction for both learning and inference, and discretize the output space for tractability. More recently, Schwing et al. (2012); Schwing and Urtasun (2012) showed how to perform exact (and more efficient) inference of the room box, which allowed to obtain better 3D reconstructions.

One problem with the framework of structured prediction is that it makes it harder to adapt these methods to handle complex interdependencies, such as conditioning the position and size of objects on their identity. Similarly, all recent works relied on approximating objects with simple 3D blocks, as more detailed geometric models would likely make the inference intractable. One exception is the work of Hedau et al. (2012), where objects with backrests are modeled by allowing a plane on top of the standard 3D block. However, if we exclude the backrest, this work still relies on the block representation. Also Satkin et al. (2012) used more complex 3D geometry. In their work they matched full, detailed models of bedrooms and living rooms available from Google Warehouse to images, but they do not allow any variability in the size and the arrangement of the objects in the mode, and their method does not extend to unseen configurations.

To overcome this problem, we rely on Bayesian inference to model the complex geometry and interdependencies of indoor scenes. The Bayesian framework enables separating the modeling and the inference, and makes it easier to extend our method to account for more refined geometric models and additional contextual relationships among objects and the room box (Chapter 4).

Another limitation of previous work is the lack of a strong, semantic structure, which is needed to fully exploit the rich structure of indoor environments. In fact, the methods discussed so far (Hedau et al., 2009; Wang et al., 2010; Lee et al., 2010; Schwing et al., 2012) only use simple geometrical constraints, such as the Manhattan orthogonality to ensure that the predicted scene interpretation is physically plausible. Our intuition is that, on top of geometry and physics, we need semantics to obtain a globally consistent interpretation of an indoor scene.

For example, in the work of Lee et al. (2010) and Hedau et al. (2012), blocks model regions of the 3D space that are occupied by a generic object, but there is no attempt at identifying which one (e.g. a couch or a bed). Here, we want to recognize the objects as well, while simultaneously inferring their geometry. This achieves a fuller understanding of the scene, and allows object knowledge to help fit the overall geometry. For example, knowing that a specific block is approximating a bed rather than, say, a wardrobe, adds constraints on the box's 3D size and position, as a bed is typically much lower. Conversely, the size and position of a 3D block provide strong cues on the identity of the object

In this work, we impose semantics by labeling each object in the scene with its semantic type, such as "bed" or "table". We then condition the 3D shape, position and size on the semantic type and on the room box (Chapter 3). These semantic labels also allow us to consider relationships among objects, for example chairs are usually arranged around a table (Chapter 4).

Indoor scene reconstruction: applications

The advancements in 3D reconstruction, such as the room box estimation methods described above, have enabled several recent interesting applications, which have further increased the interest on indoor scene understanding. A basic understanding of the 3D environment provides in fact geometric context for important applications, such as object recognition, prediction of human activities, and robotics.

For example, Hedau et al. (2010) used their estimate of the room box to build a 3D detector for beds. Specifically, given the recovered room geometry, they slide a 3D cuboid approximating a bed on the recovered floor, avoiding intersections with the walls. They then evaluate the projection of the hypothesized object against image evidence using Histograms of Oriented Gradient (HOGs) filters. Interestingly, this method overcomes some of the major problems of traditional detectors, which rely on a 2D sliding window approach since the size and position of the object is not known. The 3D room box instead informs on the likely size and the position of the bed, allowing to reduce the number of false alarms. Further, they rectify the faces of the hypothesized beds using the estimated camera, which allows comparing the HOGs in a space that is viewpoint invariant.

Gupta et al. (2011) used the estimated scene geometry to predict human activities and poses, for example to identify locations where people can sit or lie. Their method jointly models the space of human poses and 3D geometry, which is used to predict likely human poses and locations from a single image. Karsch et al. (2011) showed instead how to insert realistic Computer Graphics objects into legacy photographs with a semi-automatic method that does not require access to any scene measurement. In their work, the estimated camera is used to render the object in the estimated 3D scene, by taking occlusions into account. Interestingly, they also exploit the recovered 3D environment in order to estimate the scene illumination with the help of limited manual annotations, as correct lighting is needed for a realistic rendering.

Last, recovering 3D geometry is potentially very useful for robotics applications. In this context, the input is typically a sequence of images captured by a moving camera, and optionally data from laser range finders, which is much more informative than the case of a single image considered here. In this context, most traditional methods model a scene in the form of a 3D point cloud, which is inferred from the input video using techniques such as Structure-from-Motion (Hartley and Zisserman, 2004; Cornelis et al., 2006; Pollefeys et al., 2008) or Visual SLAM (Davison, 2003; Newman et al., 2006; Flint et al., 2010). More recently, methods in this field started to use the models developed for indoor scene understanding described in this section. These representations have the advantage of being more compact than point clouds, and capture meaningful properties of the environments, such as most surfaces being planar and orthogonal to each other. For example, both Furukawa et al. (2009) and Flint et al. (2011) modeled indoor scene as a collection of orthogonal surfaces, which is very similar to the model proposed by Lee et al. (2009), and used this representation to reconstruct a 3D map of the environment from a video. Tsai et al. (2011); Tsai and Kuipers (2012) used instead the "floor-wall" model by Delage et al. (2006) to generate scene hypotheses from the first frame of a video captured by a robot. They then used the remainder of the video stream to identify and update a Bayesian posterior probability distribution over the set of scene hypotheses, and select the scene explanation maximizing such distribution.

Recently, there has also been much interest in using inexpensive RGB-D sensors like Microsoft Kinect. For example, Silberman et al. (2012) used RGB-D data to detect the major surfaces in an indoor scene, while Koppula et al. (2011) combined RGB-data, geometric cues and contextual relationships to detect indoor objects. Ren et al. (2012) later provided a survey on recent methods for indoor scene labeling from RGB-D data. Last, Kim et al. (2012) exploited the rich structure and recurring patterns of indoor environments to identify objects given the output of a fast range scanner.

1.1.2 Object representation for recognition

Our method models objects in 3D and identifies them in the context of the overall scene, and thus relates to much work on object recognition and representation. Recognizing objects in images has in fact been a very active area of research, and a challenging one, due to the large variations in object appearance, shape, and pose, and the ambiguities created from projecting 3D objects into a 2D image (Fig. 1.2).

As a first general difference, our goal is to provide a global interpretation of the scene, which requires identifying the objects in it and understanding their interactions, while traditional recognition systems are designed to detect a specific object, or an instance of a specific object category. In this sense, our method is closely related to the work of Hoiem et al. (2006), who showed that individual elements in the scene can be detected more accurately if we consider their interplay. However, our work also builds on the great amount of literature on designing 3D and 2D representations of objects that are useful for recognition, which is the topic of this section.

Researchers used three-dimensional representations since the early days of computer vision. 3D models can in fact predict the appearance of an object under different viewpoints and conditions of illumination. A central challenge in this case is how to match 3D models to images. Over the last decade, the focus has shifted to view-dependent 2D representations, and recent machine learning methods for recognition rely on statistical models over the appearance of image patches and edge boundaries, which are view-dependent.

We now provide an overview of these two different approaches, starting with a discussion of three-dimensional representations, like ours, and their advantages over two-dimensional view-based models, which we consider next. Last, we discuss how our modeling of objects as sets of contiguous 3D parts relates to other work using part-based representations.

3D representations

Our work relies on a strong 3D representation, where the camera and the 3D objects are modeled separately. This allows us to address the ambiguities created by the imaging process. For example, while objects overlap on the image plane, they do not intersect in 3D, and reasoning about their position with respect to the camera helps explain occlusions.

A 3D representation models an object in the world, where its structure and size have much less variance than if we were to approximate them on the image plane. For example, most furniture is characterized by right corners, but we previously showed that the projection of such corners can result in just any angle due to the perspective transformation (Fig. 1.2). Additionally, in 3D we can consider the real size of an object in the world, whereas this information gets lost in 2D, since objects farther away from the camera are smaller. Similarly, relationships among objects, such as chairs around a table, or relative information about the internal structure of an object, such as the angle between a chair's seat and the backrest, are better modeled in 3D. Another advantage is that we can use prior knowledge available from sources other than images. For example, in this work we learn priors on furniture's 3D size from the text available in furniture catalogs. Learning these quantities from images would be challenging, since, again, this information gets mostly lost due to the imaging process. Unfortunately, matching 3D models to images is a challenging, under-constrained problem, which requires estimating an object's structure, its pose, and the camera from image data.

Vision researchers have represented object with three-dimensional models for a long time. One of the first recognition systems using 3D models is the block-world framework by Roberts (1965). In this method, objects are restricted to polyhedra, which are found in images by establishing a mapping between the 3D edges of the model and the edges detected in the image. This work provided important insights on how to determine the pose of an object in an image given its 3D representation, despite being restricted to very simple objects on uniform background. Later, Agin and Binford (1976) considered curved objects, which they approximated by sweeping a variable cross section along a curved axis.

An alternative representation that emerged in the same period, called aspect-graph, is obtained by organizing a series of distinct 2D views of a known object into a network or graph (Underwood and Coates, 1975; Koenderink, 1976; Chakravarty and Freeman, 1982). Recognition is then preformed by matching an image to the stored views of the objects, and thus reduces to a series of two-dimensional matching problems. Notice that this paradigm achieves pose-invariance by considering multiple 2D views of the object instead of using a completely 3D representation. For more details on this and other approaches in the early days of vision, we refer to the comprehensive survey by Mundy (2006).

In more recent work, Lowe (1987, 1991), managed to fit more complex, parametrized 3D models to images. Their method uses a gradient decent algorithm to estimate the 3D parameters of the model, based on an objective function that evaluates the match between the contours of the projected model and detected image edges, which is similar in spirit to our generative approach (Fig. 1.1). Huttenlocher and Ullman (1990) then showed how to compute a transformation from a 3D model coordinate frame to the 2D image coordinate frame in closed form, given a few known correspondence points between the 3D model and its projected image.

One problem with the methods discussed so far is that they rely on using a given, fixed 3D model, which does not allow significant variations in size, shape and appearance. This makes it challenging to account for variations within an object class (e.g. couch), because we potentially need a model for each instance of that category. This challenge has been tackled by modeling the variations of a categorical model statistically. For example, Pope and Lowe (1996) learn a probability distribution describing the range of variations in an object's appearance from a set of training images. However, their model is more related to 2D-based representations, as it does not rely on a completely 3D model. In fact, variations in an object's pose are modeled by first partitioning the training images in different cluster corresponding to different views, and then learning the statistics over appearance within each cluster.

Our approach is instead more related to methods that aim at detecting objects by also detecting their 3D pose and enforcing 3D consistency (Savarese and Fei-Fei, 2007, 2008). A major difference with respect to these methods is that we rely on an even stronger model, as our goal is to identify objects by using specific geometric topologies, whose structure is encoded in 3D. For example, we use a chair model with a fixed 3D structure (i.e., a backrest on top of a seat on top of a set of legs), as opposed to linking parts of the objects as seen from different viewing points (Savarese and Fei-Fei, 2007).

Similarly, we advocate a stronger 3D representation than other recent recognition systems based on modeling the geometry of object categories (Liebelt and Schmid, 2010; Xiang and Savarese, 2012). In fact,we encode geometric variations within an object category in 3D, for example using priors on 3D size, instead of learning orientations and distances among the parts of an object in 2D (Xiang and Savarese, 2012). Our 3D representation is also independent of the camera, hence we do not need to discretize the viewpoint and learn a different model per viewpoint (Liebelt and Schmid, 2010; Xiang and Savarese, 2012).

Another difference with respect to these methods, and more in general with respect to traditional object recognition systems, is that we do not train specific appearance models for each object category. Instead, we identify objects in the context of the overall scene, and our likelihood function globally evaluates the fit of the entire scene hypothesis, and not of each individual object. Our representation is instead closely related to to the work of Schlecht et al. (2007); Schlecht and Barnard (2009a), where the 3D structure and the imaging system are modeled separately, and **the statistics over variations in shape and size are modeled in 3D**. The benefits of this representation were demonstrated in two settings with a very different imaging process. In the first case (Schlecht et al., 2007), the biological structure of fungi is inferred from a stack of microscopic images captured at incremental focal lengths. In the second case (Schlecht and Barnard, 2009a), topological 3D models are learned from images of furniture captured by a standard perspective cameras. Both experiments confirmed that a major advantage of 3-D models is their capacity for quantification.

Last, we relate to recent approaches for detecting 3D oriented cuboids from single images (Hedau et al., 2010; Xiao et al., 2012; Fidler et al., 2012), which provide bounding boxes for generic objects. Some of these methods (Hedau et al., 2010; Fidler et al., 2012) rescore detected cuboids with contextual information, for example by penalizing cuboids that intersect each other in 3D, or that are not compatible with an initial estimate of the scene geometry (e.g. the room box in the case of indoor scenes). While this is on the lines of our approach, we instead want to identify real-world objects rather than generic cuboids, and we use the fact that the geometry varies among object classes to better recognize them. Further, while we also understand objects in the context of the 3D scene, we do so by estimating the scene and the objects jointly.

2D representations: View-based vision

Over the last decade, there has been a shift in interests towards methods based on 2D spatial representations. Thanks to major advancements in machine learning and pattern recognition, researchers have developed several promising recognition systems using discriminative classifiers and statistical models over the 2D appearance of objects. A key idea is that simple detection and recognition problems can be solved directly on the image plane using statistics over simple features such edges, color and texture, without having to face the challenges of matching 3D models to images. As another motivation, having to build detailed 3D models for each object category we want to recognize does not scale well with the number of categories. In this section we provide an overview of these methods, and conclude with a discussion of their main limitations.

Early work represented objects with global descriptors (Pontil and Verri, 1998; Schiele and Crowley, 2000), such as color and texture histograms. Unfortunately, these methods are not robust to changes in viewpoint, illuminations and clutter, because global descriptors discard information on the spatial layout of objects, and they are orderless, since they do not preserve the frequencies of individual features. To face these challenges, newer method introduced: 1) local features, designed to be invariant to illumination and affine transformations such as scaling and translation; and 2) part-based representations, to capture the spatial structure of the object.

The SIFT descriptor (Lowe, 2004), local Histogram of Oriedent Gradients (HOGs) (Dalal and Triggs, 2005), and the affine descriptor designed by Lazebnik et al. (2003) and Mikolajczyk and Schmid (2004) are notable examples of features invariant to affine transformations. These and other similar descriptors have initially been used to build probabilistic models for recognition (Fergus et al., 2003; Fei-Fei et al., 2003, 2004). Then, following the promising classification results of SVM classifiers and new methods for discriminative training (Scholkopf and Smola, 2001), these approaches were supplanted by discriminative methods using kernels suitable for local features (Dalal and Triggs, 2005; Grauman and Darrel, 2005;

Felzenszwalb et al., 2009; Zhang et al., 2007).

Local features have often been used in combination with part-based models, which capture the spatial structure of the objects. For example, Fergus et al. (2003); Fei-Fei et al. (2003, 2004) used "constellations" of parts, where they explicitly model the spatial relationships among image patches, which represent object parts. Instead, the model by Felzenszwalb and Huttenlocher (2005) arranged parts in deformable configurations, inspired by early work on pictorial structure (Fischler and Elschlager, 1973). Other work organized invariant features in 2D rigid templates, for example Dalal and Triggs (2005) used grids of HOGs, or deformable templates, such as the method by Yuille et al. (1992) for face detection, or the work of Felzenszwalb et al. (2009), which we discuss in more detail below. Kushal et al. (2007) and Leordeanu et al. (2007) considered instead geometric constraints over the spatial arrangement of parts. Additionally, other methods used features capturing the invariant properties of the projected object contours (Shotton et al., 2005; Ferrari et al., 2008) and shapes (Jurie and Schmid, 2004; Berg et al., 2005).

A very different class of methods (Fei-Fei and Perona, 2005; Sivic et al., 2005) detects the presence of an object by counting the occurrences of representative local image features, without any attempt to estimate the location or the pose of the object in the image. This strategy is called "bag-of-features" in analogy to "bag-of-words" models for text, where a document is classified based on the orderless frequencies of the words in it (Blei et al., 2003). Bag-of-features methods are computationally efficient, and, despite being orderless, they can capture significant variations in appearance, and often outperformed more structured 2D models (Bernstein and Amit, 2005; Fergus et al., 2003; Fei-Fei et al., 2004).

Felzenszwalb et al. (2008, 2009) then introduced a mixture of deformable part-based models. The mixture model deals with major changes in appearance, such changes in viewpoint, while the deformable parts allow for intra-category variations in the spatial layout. This method proved very effective in detecting some object classes, especially their "person" detector is now considered as state-of-the-art. Additionally, very recent methods pushed their idea of using 2D models with a structured layout even further. For example, Girshick et al. (2011) used grammar models, while Zhu et al. (2010) used hierarchic deformable templates.

A common challenge that these 2D based methods have to face is achieving invariance to different views. Some methods, like the detector by Felzenszwalb et al. (2009) or the work by Schneiderman and Kanade (2000), use mixture models to account for different views. Alternatively, other methods that rely on multiple views discretize the viewpoint space (Liebelt and Schmid, 2010; Xiang and Savarese, 2012). A third approach to this problem is to rely on features designed to be as invariant as possible, and use orderless representations (Fei-Fei and Perona, 2005; Sivic et al., 2005). However, in all these cases the variations introduced by the camera are still a major confusion. On the other hand, a fully 3D representation does not suffer from this limitation, as discussed in the previous section.

A second common problem is that most these methods do not separate background from foreground features. This is obvious for bag-of-features models (Fei-Fei and Perona, 2005; Sivic et al., 2005), as they do not localize the object, but consider features from the entire image. Also template-methods have this problem. For example, both Dalal and Triggs (2005) and Felzenszwalb et al. (2009) use a sliding window approach to allow for different scales and position. While Felzenszwalb et al. (2009) consider rectangular windows with different aspect ratios, to model different views as tightly as possible, these rectangles inevitably contain much background, which is confusing evidence for both inference and training. Instead, modeling the camera and the 3D object separately allows us to predict exactly which pixels correspond to the projection of an object.

Third, even the most structured 2D models based on templates cannot provide the rich and compact representation of a full 3D model. In fact, 2D variations in size, position and in the spatial layout of the parts are impacted by the camera, which is not the case in 3D. Importantly, learning this information from images can be very challenging, and potentially requires large quantities of manually labeled training data (for example, the work of Xiang and Savarese (2012)), which is time consuming to prepare. To address this problem, recent methods by Yu and Joachims (2009); Felzenszwalb et al. (2009) used models with latent variables to learn from partially labeled image data in a weakly supervised manner. In this context, an advantage of 3D representations is that they allow integrating quantitative information from non image data, such as the text descriptions in furniture catalogs, which we use to learn the typical size of an object. For other limitations of 2D object recognition systems, we refer to the recent survey by Hoiem et al. (2012).

Last, most of these 2D methods cannot be used outside detection. In fact, high level applications such as robotics require information on the 3D geometric layout of the scene, such as the size of the objects and their distance, which 2D methods do not provide.

Part-based representation

In this work, we propose modeling an object as an assemblage of contiguous 3D parts, which provides a more realistic representation than the traditional 3D boxes. For example, instead of using a single block to approximate a table, we suggest using a model with four symmetric legs and top. Parts are shared among similar object categories, for example the four symmetric legs component is used both for chairs and tables. The statistics of a 3D part's size are conditioned on the type of the object model. For example, a backrest is used to model both chairs and couches, but its width is much larger in the case of couches.

The idea of models as sets of primitive parts goes back to at least the work of Biederman (1987), and we also saw in the two previous sections that several object recognition systems rely on part-based representations. In the case of 2D methods, using parts allows to capture the local variations of the objects and their spatial layout (Kushal et al., 2007; Leordeanu et al., 2007; Fergus et al., 2003; Fei-Fei et al., 2004; Felzenszwalb et al., 2009; Crandall and Huttenlocher, 2006). This typically guarantees more robustness to changes in the viewpoint, intra-class variations, and, potentially, occlusions. Further, sharing parts among object classes allows to exploit local similarities between different categories. The method proposed by

Torralba et al. (2004), and more recent work by Ott and Everingham (2011) and by Song et al. (2012), showed that this is a desirable property in multi-class problems with a high number of classes, since sharing structure and features allows more scalability. Notice that, in the case of 2D representations, parts and their relationships do not necessarily correspond to actual parts of an object that a human might identify. For example, Felzenszwalb et al. (2009) learn the parts from training instances in a semi supervised way, i.e. the parts of the objects have not been marked by hand, and the parts learned by their system do not necessarily have a semantic meaning. This is the case also for work that is more “3D-based” (Savarese and Fei-Fei, 2007, 2008), but where parts are a collection of 3D oriented surfaces.

Using parts provides similar advantages in 3D, which are well illustrated by the work of Schlecht and Barnard (2009a), which used a set of 3D geometric primitives to learn 3D models of furniture. Their part-based model inspired our representation, with the difference that in our case the 3D topological structure of our models is given, and not learned from images. Further, we do not learn the 3D variations in size and shape of objects from images, but from different sources such as measurements available in furniture catalogs. Despite these differences, we use this 3D part-based representation driven by the very same reasons that Schlecht and Barnard (2009a) present in their work, which we now summarize.

First, the structural variability can be modeled locally, for example the variations of a table top (e.g. round or square) are independent of the variations of the legs (e.g. three or four legs). Second, sharing parts allows sharing structure among different object classes, which allows scalability. Here, we let parts share functionalities also during inference, as we designed dedicated bottom-up strategies for each part in our palette, which are naturally shared among all objects containing that part. For example, the mechanism we use to detect legs is shared by all furniture classes supported by legs, such as chairs and tables.

Third, the 3D structure of parts is often related to function, for example most furniture whose function is “sitting”, such as couches and chairs, typically share a seat and a backrest. Parts with a semantic meaning are more useful for robotics applications, as they allow more interactions with the environment. For example, an agent could infer which parts of of an object can be grabbed or manipulated independently (e.g. the handle of a door). Notice however that not every 3D part-based model uses this semantic approach. For example, Liebelt and Schmid (2010) find 3D parts from CAD models by clustering the 3D points based on their appearance in a set of registered images. Hence, the retrieved clusters do not necessarily have a semantic meaning.

In this context, an interesting observation is that certain object categories with large intra-class variations in appearance, such as chairs, are instead more univocally defined by their function (e.g. sitting). Defining objects in terms of the actions that they enable, or “affordances”, goes back at least to the work of psychologist J. J. Gibson (Gibson, 1977), and has been used by recent work in indoor scenes. For example, Grabner et al. (2011) modeled objects both in terms of 2D appearance and function, by considering how well the predicted 3D object would support an action typical for the target category, which they simulate using a 3D statistical model of a human. Other recent work showed that human poses estimated from a 2D person detector can be used to refine the 3D reconstruction of an indoor scene (Fouhey et al., 2012), or improve object classification (Delaitre et al., 2012).

1.1.3 Statistical inference

Given the dimensionality of our output space, and the complex dependencies among the output variables, performing inference analytically would be very challenging. We thus rely on Markov chain Monte Carlo (MCMC) methods (Neal, 1993; Andrieu et al., 2003) to sample from the Bayesian posterior distribution over the model parameters.

MCMC methods are often used to solve optimization and integration in spaces with high dimensionality, and have also been used for simulating physical systems. These techniques are used to generate samples from a probability distribution that cannot be sampled directly, but that can be evaluated up to a normalization factor. More formally, MCMC methods generate samples from a target distribution by constructing a Markov chain that has the target distribution as its equilibrium distribution. For a general overview of MCMC techniques for probabilistic inference, we refer to the work of Neal (1993), and to the work of Tierney (1994), who discussed how to use MCMC to sample from a posterior distribution Further, Forsyth et al. (2001) and Andrieu et al. (2003) reviewed MCMC approaches for Bayesian posterior inference in the domain of computer vision.

In this work, we combine a few sampling techniques with different strengths to explore the vast output space more effectively. We use Hamiltonian dynamics (Neal, 1993) to sample over the continuous parameters of our model, which include the size, position and shape of each object and of the room box, and the camera parameters. This sampling technique tries to avoid random walk by using gradient information, and has been used successfully to sample a model with a similar parametrization (Schlecht and Barnard, 2009a). To change the discrete structure of the model, which includes the unknown number of objects, and the type of each of them, we use reversible jump MCMC (Green, 1995, 2003), which allows switching between parameter spaces of different dimensionality in an unbiased way.

To speed up the inference, we use a data-driven mechanism (Tu and Zhu, 2002; Zhu et al., 2000) to condition the sampling on the data, by proposing samples from image evidence in a bottom-up fashion. For example, we use detected image corners to propose objects in likely 3D positions. Further, we use top-down cues to predict structure that is likely given the current scene hypothesis, for example chairs around a table, or beds against a wall. Similarly, we generate plausible objects by sampling from prior distributions on object's size learned from 3D training data. In this sense, our method is related to work that integrates top-down and bottom-up information for object recognition (Zhu et al., 2000) and image parsing (Han and Zhu, 2005).

CHAPTER 2

Joint inference of camera, room box and indoor objects

2.1 Introduction

In this chapter we discuss the core of our approach for jointly inferring the camera parameters, the room box (i.e 3D position of walls, floor and ceiling defining the room), and the objects in it from a monocular image of an indoor scene[1]. Our goal is to infer the 3D size and position of the objects, rather than their position on the image plane.

One of the key ideas of our method is to provide a global scene interpretation, by inferring all the elements in the scene jointly, while also capturing their informative interdependencies. We thus rely on the Bayesian framework, since it allows to: 1) separate the modeling and the inference, to model the rich structure of an indoor scene in a principled way; and 2) infer all the model parameters jointly without committing to partial solution, in order to understand an image as a whole, rather than infer its components independently.

We start by introducing our generative Bayesian model for indoor scenes (Sec. 2.2), where we separately model the 3D geometry of the scene, the imaging process, and an image likelihood. This is another important point of this work, as this separation allows modeling objects in 3D, which has less variance than image-based representations, since changes in 2D appearance are due both to variations in the view and in the 3D objects. The proposed generative model also allows tackling the ambiguities of the imaging process in a top-down fashion, for example the occlusions in the image plane are explained by considering the positions of the 3D objects with respect to the camera taking the picture, which we model using a

[1]Most of the content of this chapter is the subject of our 2011 CVPR paper (Del Pero et al., 2011)

standard perspective camera(Sec. 2.2.1).

In this first version of our method, each object in the scene is modeled with a single 3D block (Sec. 2.2.2), and so is the room box, which approximates the empty space of the room as if it were empty. This is a reasonable assumption for most indoor objects, since a 3D block provides a bounding-box that approximates well furniture like beds and couches. Additionally, we use thin blocks anchored to a wall to model objects like doors, picture frames, and windows, which we call frames.

We use an image likelihood (Sec. 2.2.3) based on edges to fit the 3D model to image data. This compares the edges generated by projecting the contours of the hypothesized 3D with the hypothesized camera, and the edges detected on the image plane. We integrate this into a Bayesian posterior over model parameters, which also includes prior knowledge of the world. In this work, prior information is limited to simple physical constraints on the 3D scene structure, such as objects cannot intersect in 3D, and they have to lie on the floor.

During inference (Sec. 2.3), we use MCMC sampling to find the instance of model parameters maximizing the Bayesian posterior. The inference engine introduced here is very powerful, as it will also support (with small modifications) the more sophisticated versions of our model discussed in Chapter 3 and 4. We detail several sampling moves (Sec. 2.3.1 and 2.3.2) to search over the continuous and discrete model parameters, while at the same time ensuring that the physical constraints of the model are not violated.

Further, we discuss how to make the inference efficient by using bottom-up cues in a data-driven fashion (Sec. 2.3.2). First, we initialize the camera parameters from the vanishing points detected on the image plane, which are strongly constrained in indoor scenes, where most surfaces are aligned with three orthogonal directions. Second, we use corners detected on the image plane to generate 3D object proposals that are accepted with high probability.

Last, we provide a comprehensive evaluation on two datasets of indoor images, and discuss the standard evaluation criteria used in this domain (Sec. 2.4). These experiments suggest that there is indeed merit in performing joint inference over all the elements in the scene, as opposed to estimating them individually.

2.1.1 Related work

For a comprehensive review of the relevant literature, we refer to Chapter 1. Here, we only focus on the differences with work closely related to the method discussed in this chapter.

Using a 3D box to approximate an indoor scene was introduced by Hedau et al. (2009), and has been widely used since (Wang et al., 2010; Lee et al., 2010; Schwing et al., 2012; Hedau et al., 2012). This replaced more generic but also less compact representations, were indoor layouts are modeled as a set of vertical surfaces orthogonal to the floor (Delage et al., 2006; Lee et al., 2009).

The idea of reasoning about the clutter in the room to better estimate the room box, which is usually hidden by occluding objects, was also introduced by Hedau et al. (2009). In this case, the interactions between clutter and room box are modeled in 2D, which is the case also for the work of Wang et al. (2010); Schwing et al. (2012).

Lee et al. (2010) suggested instead using a full 3D model, consisting of a 3D room box, and objects approximated as 3D blocks. The 3D geometry of this model is very similar to the one we use in this chapter, if we exclude the addition of frames. An important difference with respect to this work is the Bayesian approach to the problem suggested here. In the method by Lee et al. (2010), the model and the inference method (Structured SVM) are tightly coupled, while Bayesian in inference the modeling and the inference are separate. The main advantage of this is that the model proposed here can be extend relatively easily to include more sophisticated scene and image models. More in general, this also allows our approach to be even more top-down, and with a more unified representation.

A second difference with respect to the work of Lee et al. (2010), and this applies also to the other methods mentioned above, is that they commit to an initial estimate of the camera, while we allow this initial estimate to change, and potentially improve, during inference. All these works, including ours, use standard techniques for camera estimation from Manhattan world images, as discussed in detail in Sect. 2.3.2.

The inference method used here was inspired by the work of Schlecht and Barnard (2009a) on inferring 3D models of furniture from images. We use some of the techniques introduced in their work to sample over both the continuous and discrete parameters of the model, and introduce new ones, such as a data-driven method to propose 3D objects from image corners. This concept of using image evidence to propose samples was originally developed by Tu and Zhu (2002) and Zhu et al. (2000).

The standard criterion we use to evaluate the estimation of the room box was introduced by Hedau et al. (2009), who also annotated of the datasets used for evaluation. Both this dataset and the evaluation procedure have been used by most recent methods in this domain (Wang et al., 2010; Lee et al., 2010; Schwing et al., 2012; Schwing and Urtasun, 2012). We also evaluate on a second dataset of indoor images made available by Yu et al. (2008), which we manually annotated for evaluating on room box estimation.

2.2 A Bayesian generative model for indoor scenes

In the proposed generative approach, we separately model the 3D scene and the camera capturing it, and assume that an image is formed by projecting the hypothesized scene using the hypothesized camera (Fig. 1.1). In order to fit the model to the available image data, we reverse this forward process in a Bayesian fashion, and infer the most likely 3D scene and camera from the observed image.

More specifically, we partition model parameters, θ, into scene parameters, s, encoding the 3D geometry of the scene, and camera parameters, c, modeling the projective transformation

$$\theta = (s, c) \quad . \tag{2.1}$$

Given a set of features D detected on the image plane, such as edges, our goal is to find an instance of model parameters maximizing the Bayesian posterior distribution

$$p(\theta|D) \propto p(D|\theta)p(\theta) \quad . \tag{2.2}$$

The likelihood $p(D|\theta)$ measures how well image features are explained by the features predicted by a model hypothesis θ, while $p(\theta)$ is the prior distribution over model parameters. In what follows, we discuss all these components in detail.

2.2.1 The camera model

The imaging process is modeled with a standard pinhole camera model, which specifies how a point in 3D space gets mapped onto a point on the 2D image plane. This includes the parameters characterizing the optical and digital characteristics of the camera, often referred to as **intrinsic** parameters, and the position and pose of the camera in the 3D world reference frame, denoted as **extrinsic** parameters.

Traditionally, the intrinsic parameters of the camera comprise the focal length f, which models the amount of perspective distortion, the position of the principal point, specifying the intersection of the optical axis and the image plane, and skew and pixel aspect ratio, which approximate imperfections introduced by digital cameras. In this work we estimate the focal length from the image, and keep the principal point fixed at the center of the image, which is a safe assumption for modern digital cameras. We further assume that pixels are square (pixel aspect ratio$=1$), and that the image axes are exactly perpendicular (skew angle$=0°$).

The camera extrinsic parameters encode the 3D position and orientation of the camera relatively to the world coordinate frame. This is determined by the 3D position of the camera center, and the rotation matrix that aligns the camera's axes with the world axes. However, absolute positions cannot be determined when reconstructing from a single image. We thus conveniently position the camera center at the origin of the world reference frame, and determine the position of the 3D scene relatively to the camera location.

We parametrize the orientation of the camera reference frame in terms of three Euler angles, which is a more efficient parametrization for inference than a rotation matrix. Borrowing aviation terminology, we call these three angles pitch (ϕ), roll (ψ), and yaw (γ), as illustrated in Fig. 2.1. We now univocally show how to recover the camera rotation matrix from these three angles, as different orders of rotation produce different outcomes.

The camera axes form an orthogonal right-handed reference frame, which, when no rotation is applied, is centered at the origin, with the z-axis coinciding with the world's negative z-axis. This is equivalent to saying that the camera is pointing down the negative z-axis, which is a convention from computer graphics. The pitch specifies the rotation about the camera's x-axis, followed by a rotation around the camera's z-axis (roll). We follow the intrinsic rotation convention, meaning that the reference coordinate system changes after each rotation. Further, pitch and roll are constrained within ranges of plausible values for indoor scenes ($\phi \in [-60°, 60°]$), $\psi \in [-10°, 10°]$)

The third rotation angle (yaw) determines the amount of rotation around the y-axis. We found it more convenient to separate these three rotations, to avoid discontinuities during inference that would occur when sampling over three Euler angles, caused by a phenomenon called Gimbal lock. This happens when two of the three rotation axes align, and one rotation has no effect. However, given our assumptions over pitch and roll, the output space defined over these two angles is well-behaved, while adding the yaw, which is not constrained, could potentially cause Gimbal lock.

However, we notice that since we want to recover the relative orientation between camera and scene, rotating the 3D model of the scene around its y-axis is equivalent to doing so with the camera. The yaw angle thus becomes a parameter of the 3D scene, which we introduce in the next section. With this parametrization, we can recover the three angles defining the camera orientation angles without worrying about discontinuities. Further, this choice proves particularly efficient in Manhattan world where all objects are aligned, and the yaw of the scene is thus shared by all the objects in it.

To summarize, the camera is positioned at the world origin, and its free parameters include the focal length f, the pitch ϕ and the roll angle ψ

$$c = (f, \phi, \psi) \quad , \tag{2.3}$$

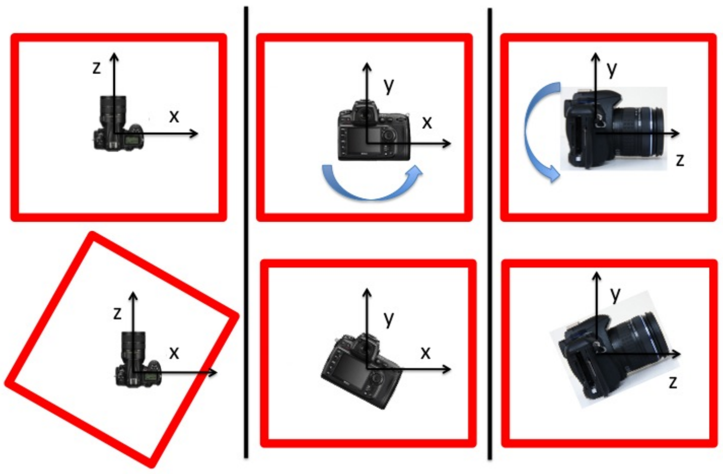

Figure 2.1: Orientation of the camera with respect to the scene. From a monocular image we cannot recover absolute positions, and we can only infer the relative rotation of the camera with respect to the scene, determined by three angles. In absence of rotation, the camera is positioned at the origin of the world coordinate system, pointing down the negative z-axis. The room box approximating the 3D scene (in red), can rotate around its y-axis, thus determining the yaw of the camera (left column). Two more angles, a rotation around the camera z-axis (roll, mid column) and a rotation about the x-axis (pitch, right column) complete the camera orientation specification. All the axes shown above are those of the world reference frame.

while the yaw of the camera is modeled by letting the 3D scene model rotate around its y-axis. As an additional constraint, the focal length has to be positive.

2.2.2 The scene model

The 3D scene model comprises the room box and the objects in it

$$s = (r, o_1, ..., o_n) \quad , \tag{2.4}$$

where the number of objects n is not known a priori. The room box models the space of a room as if it were empty. This is approximated with a single right-angled parallelepiped, defined in terms of the 3D position of its center (x_r, y_r, z_r), and its width, height and length

$$r = (x_r, y_r, z_r, w_r, h_r, l_r, \gamma_r) \quad . \tag{2.5}$$

The room yaw γ_r is the amount of rotation of the room box around its y-axis. Notice that the floor of the room box is aligned with the $x - z$ plane of the world reference frame, and its y-axis is thus a line parallel to the world's y-axis and through the room center.

As discussed in the previous section, from a monocular image we can only recover relative positions and orientations. Here, the yaw of the room, together with the pitch and roll of the camera, fully determine the relative orientation between scene and camera. The center of the room box, stored in world coordinates, determines instead the position of the scene relative to the camera, which is always centered at the world origin. Similarly, absolute sizes cannot be inferred from a single image, since it is not possible to distinguish the actual size of an object from its distance to the camera. Hence, we can only reconstruct the 3D environment up to an overall scale factor. This could be solved at any point by choosing a reasonable value for this factor, for example by setting the camera height from the floor to a plausible value, say 5 ft., which would in turn determine the absolute size of all the elements in the 3D model.

Each object in the room is also approximated with a single 3D right-angled parallelepiped, which we call **block**. A block on the floor could approximate a convex object such as a bed, or provide a bounding box for a more complex object, such as a table. Windows, doors and pictures are instead approximated with thin blocks attached to a wall. We refer to these two different object categories as **furniture** and **frames**.

All objects share the same orientation γ_r of the room block, following the Manhattan world assumption. Each object is "anchored" to a room surface, whose index is stored in a discrete variable (s_i). Furniture objects lie on the floor ($s_i = 4$), while frames are anchored to one of the walls, which are indexed from 0 to 3. To summarize the object parameters,

$$o_i = (s_i, x_i, y_i, z_i, w_i, h_i, l_i) \quad . \tag{2.6}$$

where (x_i, y_i, z_i) is the object center and (w_i, h_i, l_i) are its width, height and length.

While the 3D coordinates of the room box center (x_r, y_r, z_r) are relative to the world coordinate frame, the center of an object is relative to the coordinate frame defined by the room center, and whose axes are aligned with the room walls, which we call “room coordinates”. The advantage of this representation is that it allows for efficient computation of intersections among furniture objects (Fig. 2.2), between an object and the room box, and among frames on the same wall (Fig. 2.3).

We now define a coordinate transformation function between the room and the world reference system for later use, as we will often need to transform points from one reference system to the other

$$r_{coord}(x, y, z) = (x^r, y^r, z^r) \quad . \tag{2.7}$$

(x, y, z) is a point in world coordinates, and r_{coord} transforms it into room coordinates. Since the world $x - z$ plane is parallel to the room floor, r_{coord} simply applies a translation and a rotation around the y-axis defined by the room yaw γ_r. The inverse of this function $r^{-1}_{coord}()$ transforms points from the room to the world coordinate system.

It is important to notice that, conditioned on the attachment variable s_i, not all object parameters are free. In the case of furniture objects ($s_i = 4$), given the object height h_i, the y coordinate of the object center is not a free parameter. Specifically, $y_i = -\frac{y_r}{2} + \frac{h_i}{2}$, where $-(y_r/2)$ is the position of the floor in room coordinates (Fig. 2.2). The attachment variable analogously constrains a frame’s parameters, as illustrated in Fig. 2.3.

In addition, we enforce some simple geometric constraints to ensure that the model is physically plausible. Specifically, the camera and all the objects must be fully inside the room box, and objects cannot intersect each other in 3D. The semantics encoded by this initial version of the model are limited to these simple

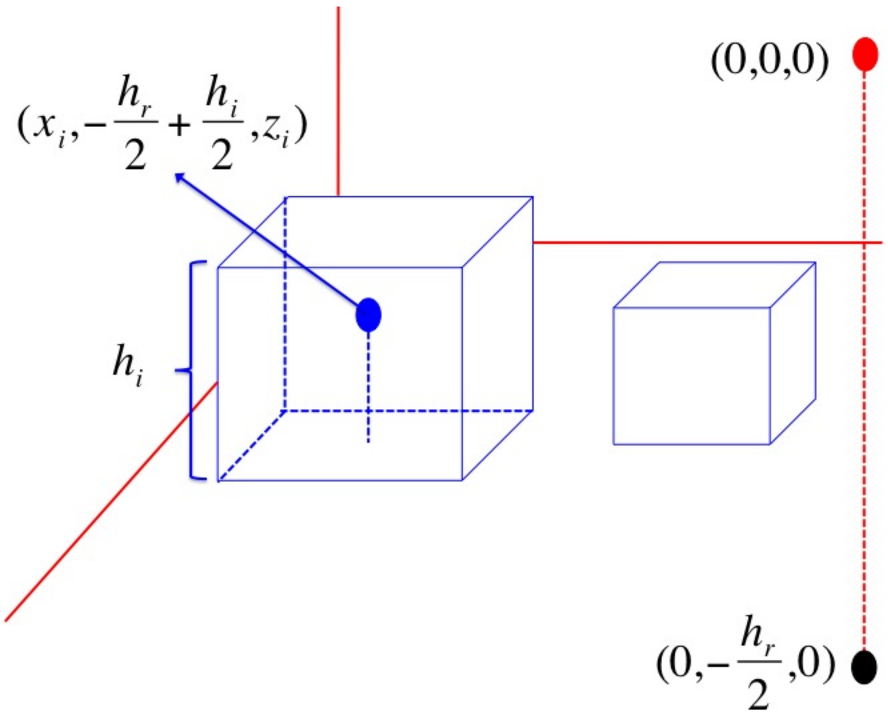

Figure 2.2: We model a piece of furniture with a single 3D block. This approximates most indoor objects (e.g. beds, tables, etc.). Each furniture block lies on the floor of the room box (in red), and is aligned with the room surfaces. Its parameters include the block center and size (Eq. 2.6). The center coordinates are relative to the room box coordinate system, where the origin is the room center (the red dot) and the axes are aligned with the room surfaces. In this reference frame, a point on the floor has $y_{floor} = -\frac{h_r}{2}$, where h_r is the height of the room. Since a furniture object is "anchored" to the floor, the y coordinate of its center (the blue dot) is not a free parameter ($y_i = y_{floor} + \frac{h_i}{2}$). Notice how in the room reference frame we can efficiently test whether two objects (the blue boxes above) intersect, by checking if there is overlap between their bases, which are 2D axis-aligned rectangles. We can similarly check whether an object is completely inside the room box.

constraints, and to the distinction between furniture and frames. This weak semantic representation, where block models a generic piece of furniture, and there is no attempt to identify its type (e.g. bed, couch, etc.), also prevents from considering more meaningful contextual relationships than just avoiding intersection. Notice that these constraints can also be seen as a strong prior preventing invalid configurations, which is also the only prior information we use in this version of the model. A stronger semantic representation will be the object of the next chapter.

2.2.3 The image model

We model an image as a collection of detected features $D = (f_1, ..., f_n)$, which we assume to be statistically generated by the hypothesized camera and scene parameters. For each feature (e.g. edges), we have a **detector** to compute the response for that feature from the image data, and a **generator** that stochastically predicts the response of the detector given an instance of the model parameters θ. We then evaluate how well that instance of the model fits the image data with respect to f_i by using a likelihood function $p(f_i|\theta)$, which evaluates the response found by the detector for f_i against the response predicted from the model. This is very similar to the image model proposed by Schlecht and Barnard (2009a).

Given the entire set of features, the likelihood function takes the form

$$p(D|\theta) = \prod_{i=1}^{F} p(f_i|\theta) \quad , \tag{2.8}$$

where we assumed independence among the features in D. In this chapter we discuss a simple likelihood function using just one feature, edges, which we use to demonstrate the benefits of inferring the scene, the camera and the objects jointly. In the next chapters we will extend our feature set, in order to make the method more robust.

The edge likelihood

We assume image edges to be generated by the projected contours of the room box and the other objects in the scene. In this process, 3D objects are considered as

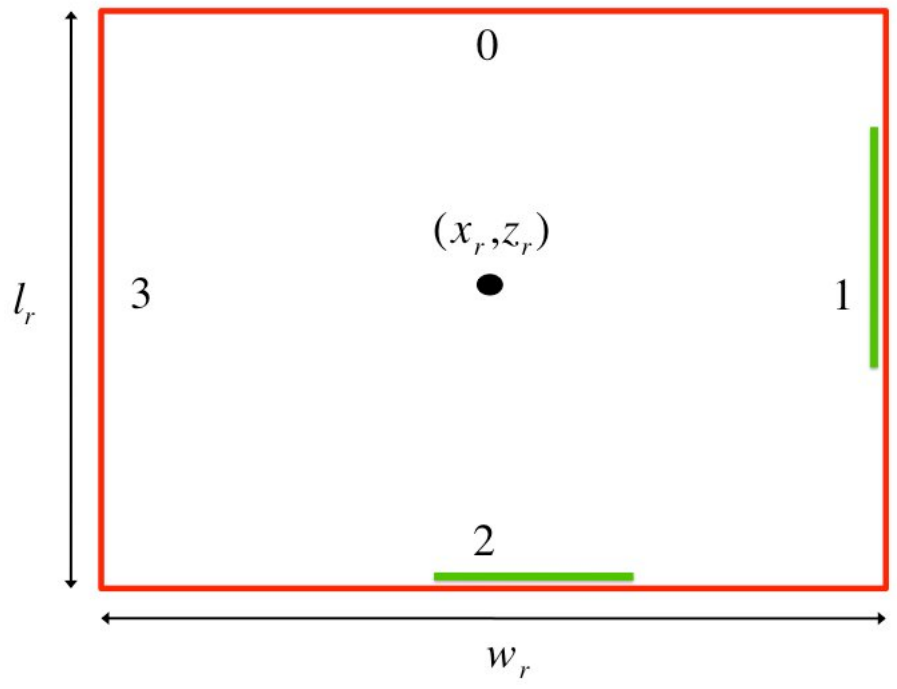

Figure 2.3: Frame parameters. Frames are thin blocks attached to a wall, approximating objects like doors and windows. Here, two frames (in green) are attached to a wall of the room box (in red, seen from above). Like furniture (Fig. 2.2), a frame is aligned with the room walls, and its center coordinates are relative to the center of the room box (x_r, y_r, z_r). The index of the wall the frame is "anchored" to is stored in discrete variable s_i (Eq. 2.6), with walls indexed from 0 to 3. Notice how this attachment imposes constraints on the other parameters in the equation: the frame on the right has $s_i = 1$, $x_i = (w_r/2)$, and is very thin along its width ($w_i = \epsilon$, with $\epsilon = 0.01$ units). The frame at the bottom has $s_i = 2$, $z_i = (l_r/2)$, and is thin along its length ($l_i = \epsilon$). Like furniture, we test whether two frames "anchored" to the same wall overlap by checking if they intersect on that wall.

opaque, in order to explain occlusions. Following the work of Schlecht and Barnard (2009a), our edge likelihood measures how well an instance of the model parameters θ fits the edge data E_d detected on the image plane, by comparing the set of edges E_θ, generated by projecting the scene model, to E_d.

The edge model by Schlecht and Barnard (2009a) relies on an edge detector to determine whether each pixel is part of an edge or not, and, if it is, the detector also provides the orientation of the edge using gradient information. Edge detection is performed once at the beginning of the inference to compute E_d, which is then used throughout the entire process. Schlecht and Barnard (2009a) used the standard Canny edge detector (Canny, 1986), but any similar detector can be used (we discuss the one used in this work at the end of the section).

The edge set E_θ predicted by the model is the collection of pixels that lie on the projected contour of one of the scene objects. For each of these pixels, it is possible to predict the orientation of the corresponding image edge from the 3D model and the camera. Notice that E_θ is generated every time we need to evaluate a new instance of model parameters, and we do so efficiently by using modern rendering software and hardware, as discussed in Appendix A.

Given these two edge sets, the edge likelihood function $p(E_d|E_\theta)$ is specified using the following intuitions:

- Suppose that, for each data point $e_{dj} \in E_d$, we knew which point $e_{\theta k} \in E_\theta$ was responsible for generating it. Ideally, the predicted position and orientation of $e_{\theta k}$ should coincide with those of the detected edge point e_{dj} it generated. Conditioned on knowing this correspondence between $e_{\theta k}$ and e_{dj}, we could softly penalize this match according to the distance between the points d_{jk}, and the difference in orientation ϕ_{jk} between the edges. Specifically, we would use $p(e_{dj}|e_{\theta k}) = \mathcal{N}(d_{jk}, 0, \sigma_d)\mathcal{N}(\phi_{jk}, 0, \sigma_\phi)$. Notice that the probability is maximized when the position and orientation of the matched edge points coincide, and it decreases as either their distance on the image plane or their difference in orientation increase.

- Each data point e_{dj} should be generated by a model point. If this not the case, we consider e_{dj} as noise, and define p_n as the probability of such an event occurring.

- Each predicted $e_{\theta k}$ should generate exactly one data edge point. However, we explain the case where no data point is available in the neighborhood of the projected model point as a miss by the edge detector. As for noise, we introduce the probability of such an event occurring, but we differentiate between two cases by using probabilities p_{hmiss} and p_{smiss}. The former is used for predicted "hard" edges arising from occlusion boundaries, such as the edges that belong to the silhouette of an object. The latter is used for "soft" edges, such as the room edges and non-silhouette edges from objects. Differentiating between these two cases is useful, since the detector is less likely to miss a "hard" edge than a "soft" edge. Notice also that an advantage of using a full 3D model is that we can determine whether hypothesized edge points in E_θ are hard or soft (see Appendix A).

Unfortunately, the correspondences between predicted and detected edge points, which needs to be re-established any time that the model parameters change, is not known. Searching for the most likely correspondence is not a viable option, as there are exponentially many possibilities. Hence, we approximate the correspondence by matching edge points locally in a greedy fashion. We assume that an edge point $e_{dj} \in E_D$ with no correspondence information was generated by a set of several candidate points from E_θ, and that each point in E_θ can be matched to at most a detected edge point (Schlecht and Barnard, 2009b).

This is accomplished as follows. First, For each point e_{dj} in E_D, we compute the distance between e_{dj} and any point $e_{\theta k} \in E_\theta$ that lies on the direction defined by the gradient computed at e_{dj}. Then, we establish a link between each point in E_θ and the closest point in E_D based on the distance. A link is not created if the distance to the closest detected edge point is more than 40 pixels, or if the difference in orientation between the two edges is more than 0.7 radian. This procedure guarantees that each $e_{\theta k} \in E_\theta$ will be matched to at most one point in E_d, since we assumed that a model point can generate at most one data point. In case $e_{\theta k}$ does not lie along the gradient of any detected point, or either its distance or difference in orientation with respect to the closest point are larger than the allowed thresholds, it will not be linked to any data point, and considered as a missed detection. It will be labeled as either "hard" or "soft" depending on the type of the model edge that generated it.

After this step, each e_{dj} in E_D will be matched to a disjoint set of predicted edges M_j, which are the candidates for generating e_{dj}. We then introduce

$$p(e_{dj}|M_j) = \frac{1}{|M_j|} \sum_{e_{\theta k} \in M_j} p(e_{dj}|e_{\theta k}) \quad , \tag{2.9}$$

which is the probability of e_{dj} being generated by one of the linked model edges, where we average the response of each point in M_j, since we do not know which one actually generated e_{dj}. Notice also that set M_j can be empty for some e_{dj}, due to no predicted edges along the data edge gradient, or because all predicted edges are closer to a different data edge. In this case, we label e_{dj} as noise.

At the end of this procedure, we have a set of data edges matched to at least a model point, a set of noise points, and two sets of missed points ("hard" and "soft"). The full edge likelihood function is then computed as

$$\tilde{p}(E_d|E_\theta) = p_n^{N_n} p_{smiss}^{N_{smiss}} p_{hmiss}^{N_{hmiss}} \prod_{\{e_{dj}:|M_j|\geq 1\}} p(e_{dj}|M_j) \quad , \tag{2.10}$$

where N_n is the number of noise points, and N_{smiss} and N_{hmiss} are, respectively, the number of "soft" missed and "hard" missed points. Notice that, conditioned on the model hypothesis, the contribution $p(e_{dj}|e_{\theta k})$ of a match is independent of all the other matches (Schlecht and Barnard, 2009a). We further adjust this likelihood function to make it less sensitive to the number of edge points, which we found makes it more stable over a larger variety of input data. Specifically, we use

$$p(E_d|E_\theta) \approx \tilde{p}(E_d|E_\theta)^{(N_{hmiss}+N_{smiss}+N_n+N_{matches})^{-1}} \quad . \tag{2.11}$$

In this work, we set the parameters $(\sigma_\phi, \sigma_d, p_{smiss}, p_{hmiss})$ to the values suggested by Schlecht and Barnard (2009b).

Unfortunately, evaluating the likelihood function as just discussed is computationally expensive, the procedure for finding correspondences being the bottleneck. However, we use caching techniques to make this process more efficient. For the details on this, we refer to the original work by Schlecht and Barnard (2009b).

Edge detection

We use the edge detection software by Hoiem et al. (2005a), which implements the algorithm for detecting straight line segments introduced by Kosecka and Zhang (2002). This works well for our purposes, since we are after edges generated by the 3D straight contours of the blocks. We then construct E_d by considering all the image pixels that lie on one of the line segments found by this detector. This step is required because in the edge likelihood discussed above, the correspondence is established between individual edge points independently, and not between line segments.

As suggested by Schlecht and Barnard (2009a), a likelihood using edge points as opposed to line segments is more robust to mistakes in the process of grouping needed to find line segments, because edge points from different edges can be erroneously linked, or, vice versa, a continuous edge might be broken into multiple segments. Note that both Canny's and Hoiem's detector start by estimating a gradient map and then link edge points along dominant gradient directions, but the latter also checks for straightness when constructing edges. We found that this is more robust to noise in indoor scenes, where most edges are straight, and it also results in a more accurate estimate of the edge orientation.

2.3 Inference

Given the observed image data, our goal is to find the most likely model under the Bayesian posterior distribution (Eq. 2.2). This is an optimization problem over the output space Θ, which is defined by the camera and room box parameters, the unknown number of objects, and the parameters of each object. As already mentioned, a key idea of this work is to infer all these entities jointly, to provide a global interpretation of an image that takes into account the interplay among the scene elements.

In this scope, the main challenges are the large and unknown dimensionality of Θ, and the several complex dependencies among the output variables, such as objects not intersecting each other. Further, the Bayesian posterior is a very complex distribution, which cannot be evaluated analytically. This is in fact the case even in the basic version of the model discussed in this chapter, where the posterior reduces to evaluating the edge likelihood

$$p(\theta|D) \propto p(D|\theta)p(\theta) = p(E_d|E_\theta) \quad , \tag{2.12}$$

since edges are the only feature used, and we have no priors other than the geometric constraints in 3D. However, the edge likelihood is a very convoluted function with deep local minima defined over a high dimensional space, and it is virtually impossible to evaluate it analytically.

We address the challenges in this optimization problem by using Markov chain Monte Carlo (MCMC) sampling to explore the output space in search of a maximum under the posterior. This technique is in fact suitable for large dimensional spaces (Andrieu et al., 2003), and for situations where distributions cannot be characterized analytically (Neal, 1993).

MCMC methods generate random, unbiased samples from the output space θ by following a Markov chain, which at each step t creates a new sample θ^t by proposing changes to the parameters of the sample at the previous step θ^{t-1}. This process of proposing a sample from the previous one is often referred to as “move”. In our scenario, moves fall in two different categories, depending on whether they change the continuous parameters of the model, which include the room box, camera, and object parameters, or the discrete parameters, in this case the number of objects in the scene. The proposals from these two different strategies are often referred to as “jump” and “diffusion” moves Tu and Zhu (2002).

During inference, we randomly alternate both kind of moves. For diffusion sampling, we rely on Hamiltonian Dynamics (Neal, 1993), an efficient sampling strategy that avoids the random walk behavior typical of simpler forms of MCMC. For jump moves, we use the Reversible Jump modification of MCMC (RJMCMC) (Green, 1995, 2003), which allows switching between parameter spaces of different dimensionality in an unbiased way. Jump moves pose in fact the challenge of comparing models that live in subspaces with different dimensionality, in our case scenes with a different number of objects. Another challenge introduced by jump moves in such a large output space is that naive proposals, such as samples from a prior distribution, would lead to unacceptably long running times. We address this problem with a data-driven strategy (Tu and Zhu, 2002; Zhu et al., 2000) to condition the sampling on the data, by proposing samples from image evidence in a bottom-up fashion.

In what follows, we first detail with the Diffusion moves in Sect. 2.3.1, where we develop concepts needed to better discuss the Jump moves (Sect. 2.3.2). We conclude with a summary of the entire inference process.

2.3.1 Diffusion moves

We use Hamiltonian Dynamics (Neal, 1993) to sample over the continuous parameters of our model. This technique avoids random walk behavior and it is thus more efficient than simpler forms of MCMC, such as using Gaussian proposal distributions centered at the current sample. Further, Schlecht and Barnard (2009a) showed that this sampling method is very powerful for addressing the complexities of their edge likelihood, which we also use here.

We follow Neal's formulation of Hamiltonian dynamics, and we refer to his work for the implementation details (Neal, 1993). In our case, the potential energy function is defined in terms of the joint distribution of the parameters and the image data $p(\theta, D)$

$$E(\theta) = -log(p(D|\theta)) - log(p(\theta)) \quad , \tag{2.13}$$

which reduces to

$$E(\theta) = -log(p(E_d|E_\theta)) \quad , \tag{2.14}$$

since only edges are used in this chapter. We follow the dynamics using leapfrog discretization (Neal, 1993), and compute the derivative of the potential energy with numerical approximation. This is the current bottleneck, as in this gradient based method we have to compute the derivative with respect to the energy for each parameter, which in turn requires evaluating the computationally expensive edge likelihood.

Hamiltonian dynamics allow to jointly sample over a set of continuous parameters, and it would then be tempting to use them to sample over all the continuous parameters of our model. However, the dependencies among these variables, introduced by the complex structure of the model (e.g. objects volumes cannot overlap), would introduce several ambiguities if we were to sample over all the parameters at the same time. Consider for example objects A and B in Fig. 2.4, and imagine sampling over the parameters of object A only. If this object tried to expand towards its right, object B would need to shrink, in order to preserve the no-overlap constraint. Now, if we were to sample over both objects jointly, this could result in a conflict where object B tries to expand towards the left, and A towards the right, and it would not be clear how to prevent A and B from overlapping.

To avoid this and other similar ambiguities, we define three different types of diffusion moves over subsets of the scene parameters, and define rules for adapting the model so that none of the geometric constraint is violated. Remember that our two main constraints are that objects cannot intersect in 3D, and that each object must be fully inside the room box. In the first proposed diffusion move, which we call Diffusion 1, we sample over the parameters of a single object, and we enforce constraints by shrinking other objects in case of overlap, and expanding the room box in case it is not big enough to contain the object (Fig. 2.4, a-b). The second move (Diffusion 2) samples over the parameters of the room box only, and adapts the objects to respect the constraints. Specifically, as the room box becomes smaller, we shrink objects in case they end outside the room (Fig. 2.4, c). Third, we jointly sample over camera and room box parameters (Diffusion 3), by enforcing the same constraints from the previous move. Further, for all the three moves just discussed,

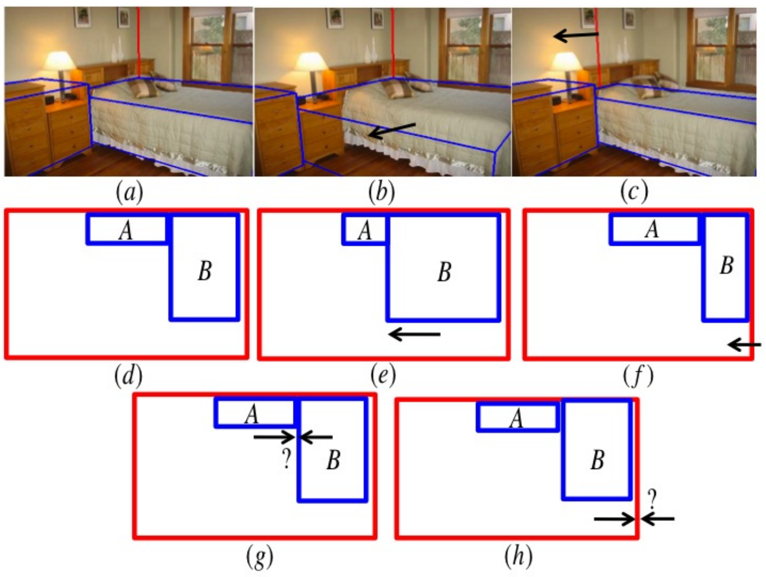

Figure 2.4: Sampling over subsets of parameters handles containment constraints (top two rows) and avoids ambiguities (bottom row). Consider sampling over the parameters of a single object at a time. As the object expands, other objects have to shrink to avoid overlap. For example, **A** has to shrink to allow **B** to grow (**a-b**), as shown also in the corresponding birdviews (**d-e**). Similarly, when sampling over the room box only, objects have to shrink so that they are entirely in the room, like object **B** in (**c-f**). Instead, sampling over two objects at the same time could result in conflicts such as in (**g**), where both **A** and **B** are contending the same 3D space. Sampling over the room box and an object jointly creates similar conflicts (**h**).

we enforce that the camera is inside the room box, and that it is not within the volume occupied by an object. These are corner cases that occur less often, and we simply reject samples violating these constraints. We now discuss these three moves in more detail.

Sampling over the continuous parameters of an object (Diffusion 1)

When sampling over the parameters of object o_i, we adapt the room box and the other objects so that no constraints are violated. When we detect an overlap with another object, we shrink the latter, and delete it if it is completely contained in o_i. We also check whether o_i is partly outside the room. If this is the case, we expand the room box so that o_i is entirely inside.

To reduce the sampling time, we designed efficient ways for sampling object parameters. For example, consider the window in Fig. 2.5 (a). To find the correct fit, the window height must be stretched, and its center must be shifted downwards. To do this efficiently, we vary the height of the window by keeping the upper edge fixed. In this example, we would run Hamiltonian dynamics on the object height h_i. At each leapfrog step t, a proposed change in the height $h_i^t = h_i^{t-1} + \delta$ is followed by changing the y position of the object center as $y_i^t = y_i^{t-1} - \frac{\delta}{2}$, achieving the result in Fig. 2.5 (b). This technique is effective in our framework, since typically one edge of the object is correctly "latched" to an image edge, given our data-driven proposals from image corners discussed in Sect. 2.3.2, and we want to find the correct size of the object without displacing that edge. There are four directions to sample a frame using this strategy, and six for furniture objects, as illustrated in Fig. 2.5 (c), (d) and (e). When executing move Diffusion 1 for object o_i, we iterate over all four possible directions if o_i is a frame, six if it is a furniture object. For each direction, we use 20 leapfrog steps, and at each step we enforce the containment constraints.

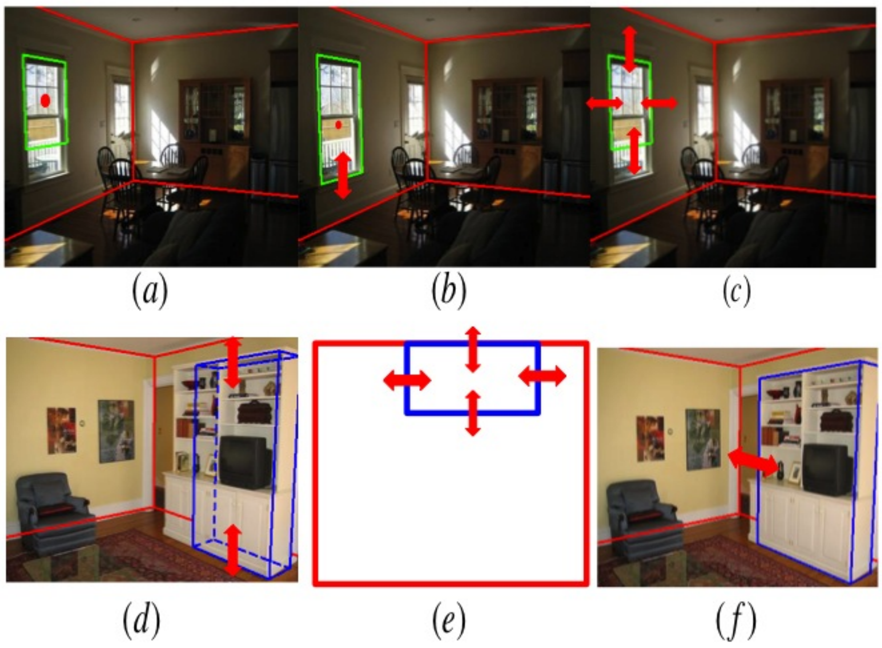

Figure 2.5: Efficient strategies for sampling over an object's continuous parameters. The upper edge of the window in (**a**) is positioned correctly, and the correct fit for the window can be obtained efficiently by dragging down the lower edge while keeping the upper edge fixed (**b**). We designed moves that achieve this by sampling the continuous parameters of the window in 3D (see text). For a frame, this principle can be applied to any of the edges, and this creates four possible sampling directions, denoted by the arrows in (**c**). For furniture we have six alternatives, as illustrated in (**d**) and (**e**), which is a birdview of the model. (**f**) is an example of the effects of this move when sampling over the direction denoted by the arrow.

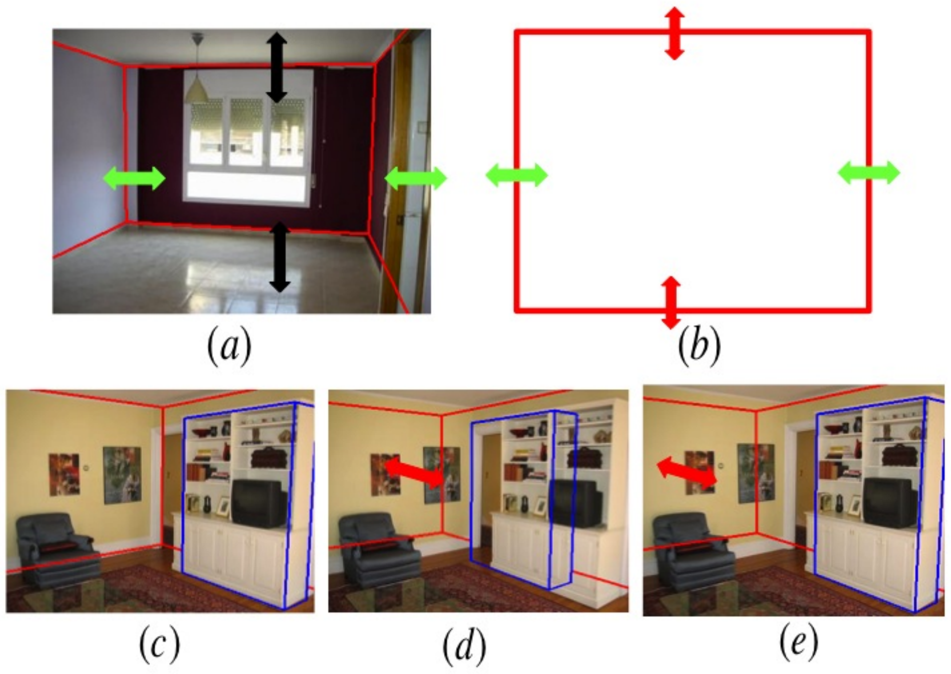

Figure 2.6: Diffusion sampling over the room box parameters. Top row: as for objects (Fig. 2.5), it is more efficient to sample along one direction of the room by keeping one edge "latched". For the room box, six sampling directions are available. Four of those are shown in (**a**), denoted by the arrows, and the remaining two in (**b**) (the red arrows). Note that (**b**) is the birdview corresponding to (**a**), and two of the sampling directions are shown in both views (the green arrows). Bottom row: since object coordinates are relative to the room box center, changing in the latter does not preserve the projection of objects already in the room. We only changed the 3D center of the room box from (**c**) to (**d**), but this "shifted" the projection of the blue block as well. By introducing a transformation that preserves the projection of the objects in the room (see text), we obtain the result in (**e**)

Sampling over the continuous parameters of the room box (Diffusion 2)

When sampling over the parameters of the room box r, we adapt all the objects so that no constraints are violated. As for the object parameters, we found that it is more efficient to sample along one direction of the room by keeping one of its edges "latched" (Fig. 2.6, top row). In the case of the room box, six sampling directions are available, and we use 20 leapfrog steps per direction. At each leapfrog step we enforce the containment constraints, by shrinking an object when we detect that it is partly outside the box, or by removing it if it is completely outside.

An undesirable effect of changing the room center is that it changes the projection of objects already in the room. In fact, an object's parameters are relative to the room box coordinate system, and changing the 3D position of the room box would not preserve the projection of the room objects. An example is shown in Fig. 2.6 (c-d), where a small change in the room center causes the object object to shift as well.

We address this by introducing a transformation that preserves an object's projection after a change in the room center, which we apply to all objects after each leapfrog step. Let us respectively denote the 3D room center at steps t and $t+1$ as $r_c^t = (r_x^t, r_y^t, r_z^t)$ and $r_c^{t+1} = (r_x^t + \delta_x, r_y^t + \delta_y, r_z^t + \delta_z)$. Let us also define $o_{ic}^t = (x_i^t, y_i^t, z_i^t)$ as the center of object o_i at step (t). Remember that the room center is with respect to the world coordinate system, while the object center is with respect to the room reference frame. To preserve the projection of the object, we compute the new center of object as

$$o_{ic}^{t+1} = (x_i^t + \delta_x^r, y_i^t + \delta_y^r, z_i^t + \delta_z^r) \tag{2.15}$$

where $(\delta_x^r, \delta_y^r, \delta_z^r) = r_{coord}(\delta_x, \delta_y, \delta_z)$, and $r_{coord}()$ is defined in Eq. 2.7. The result of applying this transformation is shown in Fig. 2.6 (d), where the room box has changed, but the blue block kept its position on the image plane. For consistency, this transformation needs to be applied also in move Diffusion 1, in the event that the room box changes in order to contain the object currently being sampled.

Sampling over camera and room box parameters (Diffusion 3)

We sample over camera and room box parameters jointly, for a total of 10 parameters. We found that sampling over the camera parameters independently typically produces samples with a lower posterior, due to correlations with other scene parameters. For example, sampling over the focal length should drive the model to the current perspective distortion, which provides a better alignment of the projected model edges and the image edges. However, changing the focal length also modifies the size of the projection of the 3D scene, and these two forces are conflicting.

Hence, we jointly sample over camera and room box, as this allows to account for some of the correlations between camera and scene parameters. For this move, we use Hamiltonian dynamics for 20 leapfrog iterations. At each step, we enforce the same constraints that we use when we sample over the room box only (Diffusion 2).

Sensible dynamics with multiple objects

The constraints in the model give rise to one additional problem during diffusion sampling, illustrated in Fig. 2.7. Suppose sampling over the block projected over the bed (a), along the direction denoted by the arrow (b) for N leapfrog steps. For a few steps, the bed tries to expand, and the containment constraints force the block on the left to shrink (b). Then, the bed tries to shrink (c), but we see that the left block does not return to its initial position. The behavior in the second row is more desirable, where the left block "grows" back to its initial location as the bed shrinks (e), but does not grow beyond that point (f).

In the third row, we have a similar situation, where we show the desired behavior of the sampler when the room box is trying to expand. As the room shrinks, so does the bed (g), which completely disappears when it is fully outside the room (h). When the box starts to expand again, the bed grows back to its initial size (i), but no further (l).

We now discuss a procedure to implement this desired behavior for all the diffusion moves introduced so far. Each of this moves samples over a subset θ' of the continuous parameters θ, which could either be the parameters of an object (Diffusion 1), those of the room box (Diffusion 2), or those of both the room box and the camera (Diffusion 3). Given an initial sample θ_{in} and an output sample θ_{out} after N leapfrog steps of Hamiltonian Dynamics, this procedure works as follows.

Algorithm 2.1: Sensible dynamics

```
1. Set $\theta_0 = \theta_{in}$
2. For every step $i = 1, ..., N$
    (a) Compute $\theta_i$ from $\theta_{i-1}$ by following the Hamiltonian dynamics. This will change the parameters of $\theta'_i$ only.
    (b) Whenever the posterior needs to be evaluated, compute it on a copy of $\theta_{i-1}$, where we apply the containment constraints with respect to the subset $\theta'_{i-1}$ (e.g. if $\theta'_{i-1}$ is the room box, shrink the objects accordingly). Never apply the constraints on either $\theta_i$ or $\theta_{i-1}$
3. Set $\theta_{out} = \theta_N$
4. Apply the containment constraints on $\theta_{out}$ with respect to $\theta'_{out}$
```

The key idea is that within a diffusion move, the containment constraints are never applied to the samples $(\theta_0, ..., \theta_N)$ in the chain. This keeps track of the initial state of the model, which allows to revert to the configuration of the first sample for all the parameters that are not in the set θ' we are currently sampling over. However, any time the posterior needs to be evaluated, we need to enforce the containment constraints to preserve a correct behavior, but we do so on a copy of the current sample, as this could potentially modify the parameters not in θ'. Finally, the containment constraints are applied to the last sample, which will be the starting point for the next sampling move.

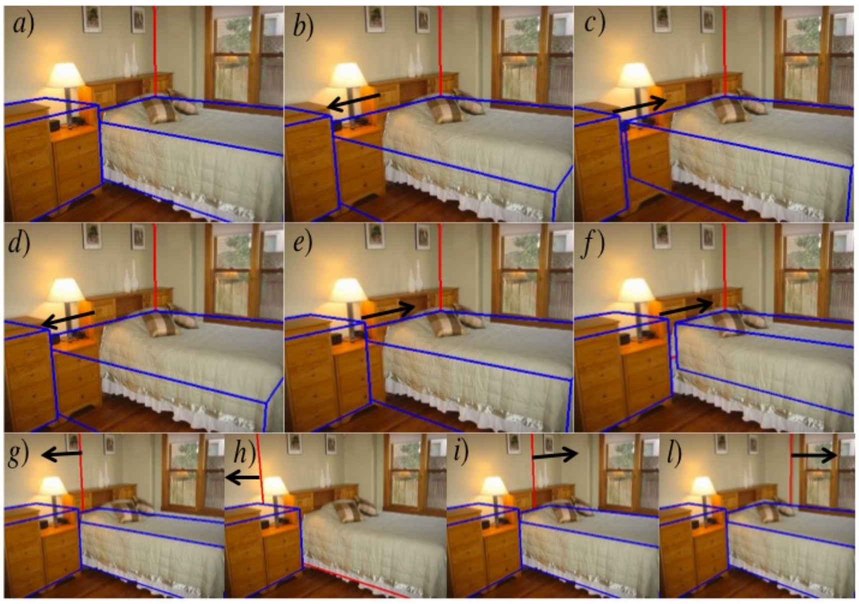

Figure 2.7: Sensible sampling with interacting objects. Suppose sampling over the block projected over the bed (**a**), along the direction denoted by the arrow (**b**). For a few leapfrog steps, the bed tries to expand, and the containment constraints force the block on the left to shrink (**b**). Then, the bed tries to shrink (**c**), but we see that the left block does not return to its initial position. The behavior in the second row is more desirable, where the left block "grows" back to its initial location as the bed shrinks (**e**), but does not grow beyond that point (**f**). In the third row, we show the desired behavior of the sampler when the room box is trying to expand along the direction of the arrow. As the room shrinks, so does the bed (**g**), which completely disappears when it is fully outside the room (**h**). When the box starts to expand again, the bed grows back to its initial size (**i**), but no further (**l**)

2.3.2 Jump moves

Jump moves are needed to change the discrete structure of the model, which in our case is the unknown number of objects in the scene. We use three jump moves that respectively add an object to the current scene hypothesis (Jump Move 1), remove an object from the scene (Jump Move 2), and replace the current scene hypothesis with a new, empty room box (Jump Move 3). In order to accept or reject samples when switching between subspaces of different dimensionality without any bias, we use the Reversible Jump modification of the Metropolis Hastings acceptance formula, which is discussed in detail in the work by Green (1995, 2003).

An important detail or our implementation is that the edge likelihood was designed to be not susceptible to changes in the dimensionality of the model. The set of edge points predicted from the model is in fact generated by projecting the entire scene, and our way of comparing this set of edges to the detected edges is insensitive to the number of objects in the scene. The additional components of the likelihood introduced in the following chapters will also have this property, and this is very useful for inference, as the likelihood of two samples with different dimensionality can be compared without being biased towards either one.

In the rest of this section, we mostly discuss how to efficiently propose adding objects to the scene, which was the main challenge we faced while designing our inference framework. In fact, the sampling space is very large, and naive jump proposals, such as samples from a uniform prior, are unlikely to be accepted, which leads to unacceptably long running times. We address this problem by introducing a data-driven strategy (Tu and Zhu, 2002; Zhu et al., 2000) to condition the sampling on the data, by proposing samples from image evidence in a bottom-up fashion. In this process, we exploit the Manhattan world assumption, as this provides strong additional constraints on the parameter space.

In Fig. 2.8, we summarize how our proposal mechanism combines the Manhattan world constraints and the advantages of data-driven inference. Here, we show that intersections of line segments on the image plane, which we call *image corners*,

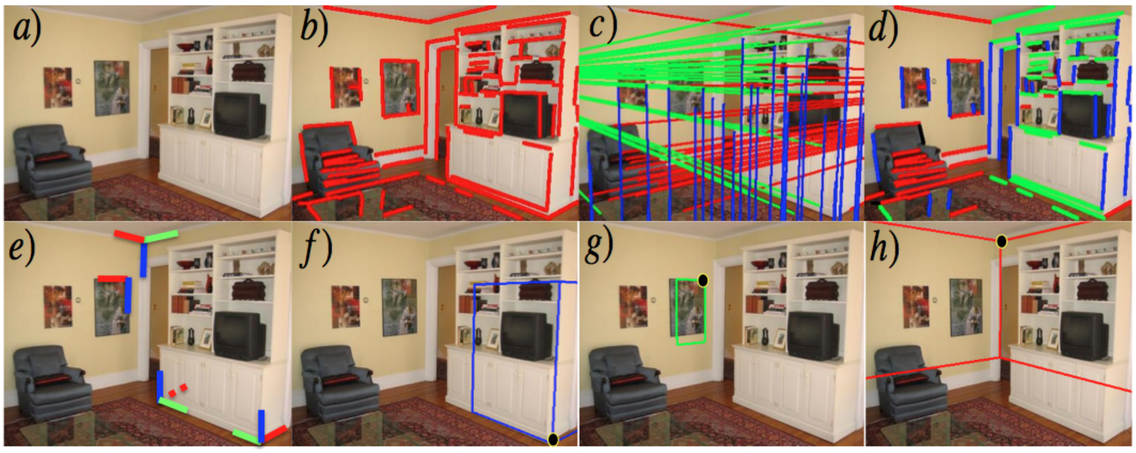

Figure 2.8: Detecting and using "Manhattan" corners for efficient object proposals. Most surfaces in indoor scenes are aligned with three principal orthogonal directions. This defines a triplet of orthogonal vanishing points in the image plane, which we find in a RANSAC fashion (**c**) from straight detected segments (**b**). Segments are assigned to one of three groups based on the vanishing point they converge to (**d**). Intersections of edges in different groups, which we call image corners, are typically generated by the projection of an orthogonal 3D corner (**e**). Given an estimate of the camera pose computed from the vanishing points, an image corner can be used to propose a furniture object (**f**), a frame (**g**), or the room box (**h**)

typically correspond to the projection of corners that are orthogonal in the 3D world. Using projective geometry and Manhattan orthogonality constraints, an image corner can be used to efficiently propose furniture objects (Fig. 2.8, f), frames (g) and room box candidates (h), which are accepted with high probability.

To achieve this goal, we first estimate the triplet of orthogonal vanishing points defining the Manhattan world directions. This provides a good estimate of the camera focal length and pose, which are needed to propose 3D blocks from image corners. In what follows, we describe the procedure for estimating the vanishing points, how to detect image corners, and how to use them to propose objects in the scene and room box candidates.

Vanishing point estimation

We first detect straight edges and fit line segments to them using the straight line segment detector by Hoiem et al. (2005a). Then, we detect vanishing points following the RANSAC procedure proposed by Lee et al. (2009). At each step, we randomly select three pairs of line segments and position a vanishing point at the intersection of each pair. We then check the orthogonality of the vanishing points and reject triplets that are not orthogonal (Lee et al., 2009). We then estimate the intrinsic camera matrix K from the Choleski decomposition of the absolute conic matrix Hartley and Zisserman (2004), which is fully determined by the position of the three vanishing points. We reject triplets that provide a non realistic focal length ($f \notin [50, 2000]$, measured in pixel), or a principal point whose distance to the image center is larger than 20 pixels.

For each valid triplet $V_t(v_1, v_2, v_3)$, we compute the objective function $f(V_t)$ as follows. First, we assign each segment s_i to the vanishing point v_k minimizing the angular distance $\alpha(s_i, v_k)$ (Fig. 2.9). A segment s_i is labeled as an outlier if $\min_{k=1,2,3} \alpha(s_i, v_k) > 0.06$. Denoting n as the total number of inliers with respect to triplet V_t, we define

$$f(V_t) = \left(\sum_{i=1}^{n} \frac{\min_{k=1,2,3} \alpha(s_i, v_k)}{l_i} \right) \Big/ n \quad , \tag{2.16}$$

where l_i is the length of segment s_i, and is used to give a larger weight to long segments.

We keep the valid triplet of vanishing points minimizing $f(V_t)$ and such that the ratio r_{out} between the number of outliers and the total number of segments is less than 0.1. If no triplet satisfying these constraints is found, we increase this threshold by 0.1 and repeat until a triplet is found. This allows for considering only triplets supported by a large number of segments first, and, if none is available, we allow a larger number of outliers. Notice that the focal length computed from the best triplet of vanishing point is also used to initialize the focal length f of our camera model.

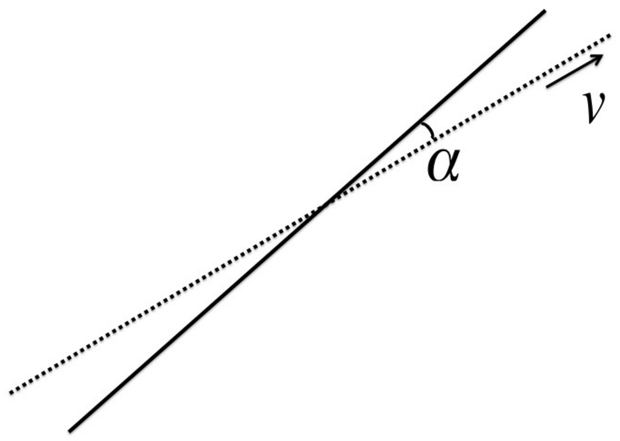

Figure 2.9: The angular distance α between a line segment and a vanishing point v. α measures the angle between the segment and the line through the mid point of the segment and v (the dashed line).

Image corners

Given the estimated triplet of vanishing points, each line segment is assigned to the vanishing point minimizing the angular distance α (Hedau et al., 2009), thus partitioning them in three groups. Segments with $\alpha > 0.12$ are not assigned to any group and considered as outliers. Notice that this threshold is less strict than the one used during RANSAC, as we want to have more candidates at this point. An example is shown in Fig. 2.8 (d), where the three different groups are shown, respectively, in green, red and blue, and the outliers in black.

We assume that two image segments in different groups were generated by segments that are orthogonal in 3D. The intersection of two or three such segments, which we call image corner, is thus likely to be generated by a 3D orthogonal corner, such as the corners of the cabinet or the inner corner of the room shown in Fig. 2.8 (e). Given an image corner and an estimate for the camera pose, we can propose 3D objects in likely positions.

We create an image corner as follows. We consider each pair of line segments converging to different vanishing points, and find their intersection on the image

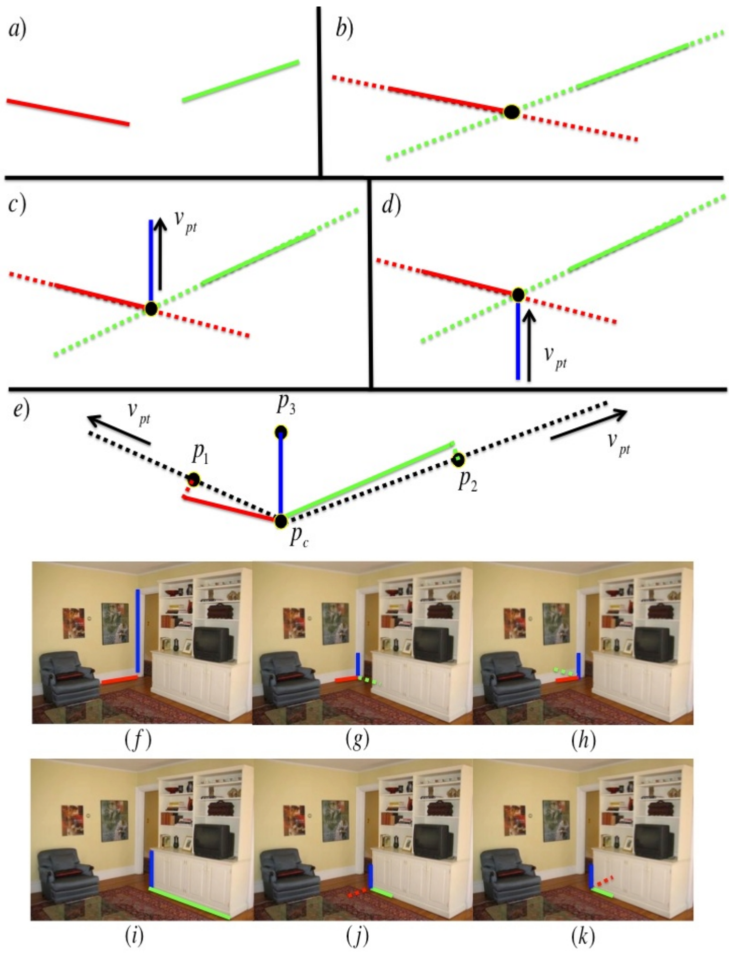

Figure 2.10: Creating orthogonal corners from line segments. Given a pair of segments converging to different vanishing points (**a**), we first find their intersection (**b**). We position the third segment of the corner on the line through the corner position and the third vanishing point (**c**). Two corner configurations are possible (**c**-**d**). Last, we "rectify" the corner (**e**) to enforce orthogonality (see text). The corner is specified by its position p_c and the three rectified segments p_1p_c, p_2p_c and p_3p_c. In the last two rows, we show examples of corners (middle-right) created from a pair of segments (left).

plane (Fig. 2.10 (**a**-**b**)), which we call corner center, denoted by p_c. We do not create corners if the distance between the intersection and each segment is larger than 30 pixels. This initial corner hypothesis might have been generated by the projection of two different 3D corners (Fig. 2.10 **c-d**). We consider both possibilities, and create two corners. For the first one, we hallucinate a third line segment through p_c and oriented towards the remaining vanishing point v_{pt} in the triplet (Fig. 2.10 **c**). For the second corner we use instead the segment through p_c and oriented away from v_{pt} (Fig. 2.10 **d**). Proposing corners from just two segments allows to detect many more candidates than if we were to consider only intersection of three line segments. This is because in most cases only two out of three segments of the corner are visible, such as the bottom left corner of the cabinet in Fig. 2.10 (**k**), where the hidden segment is denoted by the dashed red line.

Last, we "rectify" the corner (Fig. 2.10 (**e**)), to make sure it satisfies the orthogonality constraint imposed by the Manhattan world, which is needed to propose 3D objects from a 2D corner. In fact, corners are created from detected image segments, which do not necessarily satisfy these constraints due to errors in the edge detection. Specifically, instead of using the image segments, we consider their projections on the lines through the corner center and the vanishing points, which are guaranteed to be orthogonal in 3D. This rectification process also provides a more accurate estimate of the corner, as the rectified segments are found using the vanishing points, which are estimated globally by considering all edges in the image. Hence, the rectified segments are usually more robust to edge detection errors than the original segments used to initialize the corner, which were found using only local image evidence.

The corner is then fully specified by its position p_c, and the three rectified segments p_1p_c, p_2p_c and p_3p_c. Notice that the "hallucinated" segment (p_3p_c in Fig. 2.10) does not need to be rectified, as it already lies on the line between p_c and the corresponding vanishing point. The length of the hallucinated segment is arbitrarily set to 20 pixels. Notice in fact that we use corners only to propose the correct position and orientation of blocks in 3D, and not their size. Hence, we are only interested in the direction of the corner segments on the image plane, while their 2D length, which is hard to estimate reliably, is not relevant to our algorithm for proposing 3D blocks.

Proposing objects in the scene using image corners (Jump 1)

As discussed above, we create image corners by considering the intersection of each pair of segments converging to different vanishing points. A corner is then used to add an object to the scene in a bottom-up data-driven fashion, in order to increase the acceptance probability in the MH acceptance formula. Proposal generated by blindly adding objects to the scene at random locations would in fact be rejected most of the time.

We use image corners to propose both furniture objects and frames. First, the image corner is used to estimate the position of the object corner in 3D, which is conditioned on the current estimate of both camera pose and room box. For furniture objects, we consider two different algorithms for finding the 3D corner position, depending on whether the image corner is pointing up or down. These two cases are shown in Fig. 2.11 and Fig. 2.12, while in Fig. 2.13 we illustrate the similar process we use for frames. Notice that the move adding an object chooses whether to add a furniture object or a frame with equal probability.

Then, the size of the proposed object is chosen randomly, and to further increase the acceptance ratio of jump moves, we use a delayed acceptance mechanism, which makes up for the fact that, while corners help propose objects at the right position, we often propose the wrong size. Hence, we briefly sample over the continuous parameters of the newly added object before consulting the MH acceptance formula to decide whether to accept or reject the sample. During this process we also need to enforce that the 3D containment constraints of the scene are not violated.

The full proposal procedure to add a furniture object in the scene is as follows.

Algorithm 2.2: Proposing a furniture object from a corner

1. Randomly choose an image corner, and determine whether it is pointing up or down (Fig. 2.11)

2. If the corner is pointing up, find the position of the 3D corner on the floor (Fig. 2.11), otherwise find the position of the 3D corner as explained in Fig. 2.12.

3. Sample the proposed height of the object h_i from uniform distribution $\mathcal{U}(\frac{r_h}{10.0}, r_h)$, where r_h is the current height of the room box. This helps constraining the object height within a set of reasonable values.

4. Sample both the proposed object width w_i and length l_i from uniform distribution $\mathcal{U}(\frac{h_i}{10.0}, 10.0 * h_i)$, using h_i from the previous step. Again, this constrains the object's size to reasonable values.

5. Shrink any furniture object in the scene colliding with the proposed one

6. In case the object does not fit in the room, expand the room box. This might involve lowering the position of the floor, raising the ceiling, or increasing the room width and/or length. In case the room box changes here, we correct for the position of the objects using Eq. 2.15

7. Sample over the object continuous parameters using Diffusion move 1 (delayed acceptance step)

8. Accept or reject the proposed object by consulting the MH acceptance formula

Similarly, the procedure to add a frame in the scene is summarized below.

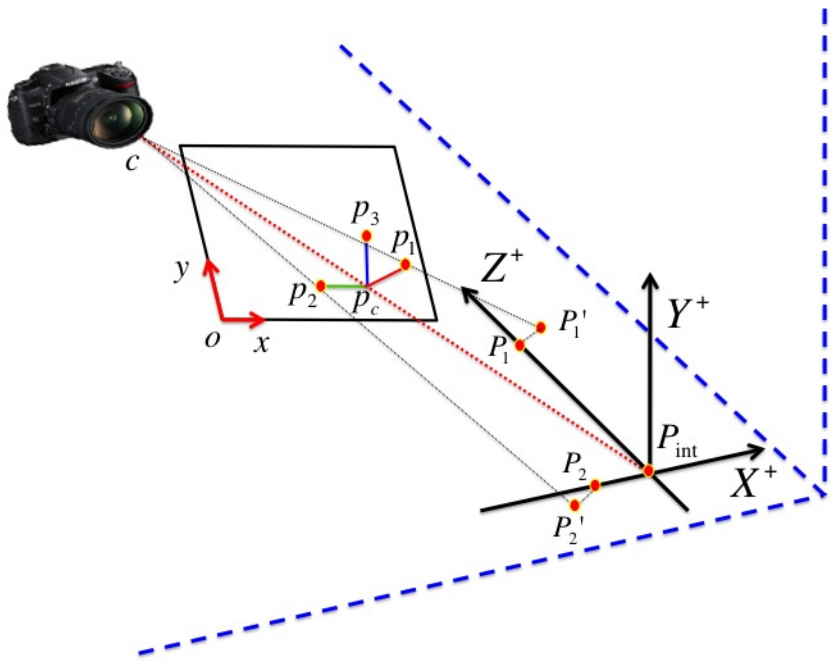

Figure 2.11: Finding the 3D corner of a furniture object from an image corner, conditioned on the current room box. $p_c = (p_{cx}, p_{cy})$ denotes the position of the corner on the image plane, and $P_c = (p_{cx}, p_{cy}, -f)$ is the corner in the 3D camera reference frame, where f is the focal length. We first consider the segment p_3p_c: if $p_{3y} > p_{cy}$ in the image reference frame, the image corner is pointing upwards, meaning it was generated by an object corner on the room floor, delimited by the blue dashed lines (for downwards corners, see Fig. 2.11). Then, we cast a ray through the camera center c and the image corner P_c, and find its intersection with the room floor P_{int}, which defines the 3D position of the object corner. Assuming objects are aligned with the walls, we define the coordinate system (X, Y, Z), aligned with the room walls and centered in P_{int}, with the XZ plane coinciding with the room floor. We know the 3D corner will expand along the positive Y axis, and use the rays between the camera and p_1 and p_2 to determine the directions along the X and Z axis. We first find the intersection P_1' between ray p_1c and the floor, which, due to small errors in the camera estimate, does not lie exactly on the Z or the X axis. We then compute P_1 as the closest point to P_1' on either the X or the Z axis, which determines that the 3D corner in the picture expands along the positive Z axis. We repeat the same for P_2 by using p_2, and in this example the 3D corner expands along the negative X axis. We have thus determined the position of the 3D corner and the direction of its sides in the room reference frame.

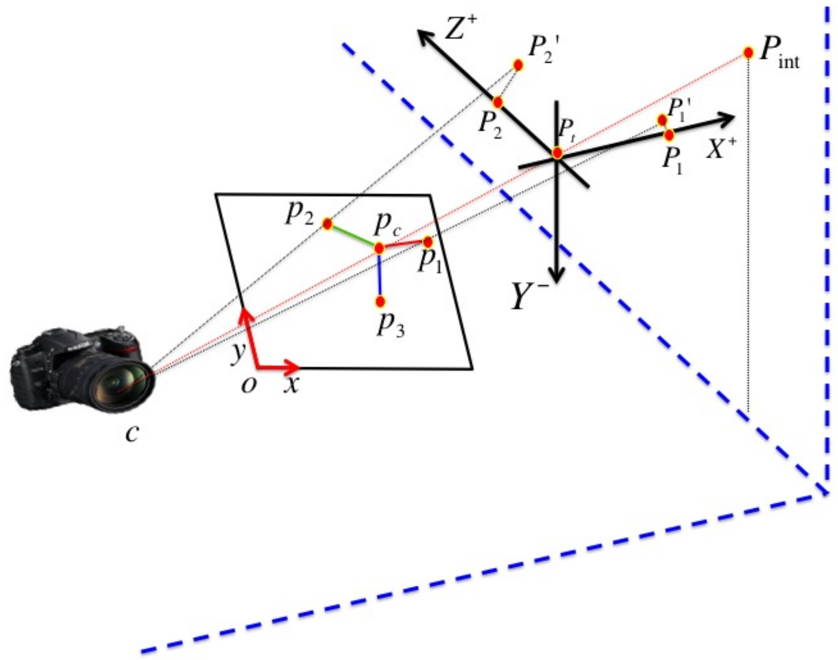

Figure 2.12: Finding the 3D corner of a furniture object from a downwards image corner. We assume that a downwards image corner is generated by a corner on the top of a 3D object (for upwards corner, see Fig. 2.11). We cast a ray through the camera center and the corner position in the image P_c, and find the intersection P_{int} between this ray and the closest room wall. The 3D position of the corner can lie anywhere on the line $((1-t)c + tP_{int})$ between P_{int} and the camera center c. Any $t \in [0,1]$ defines a valid 3D position P_t for the corner, and we pick t by randomly choosing from the interval $[0.4, 1.0]$. We set the lower bound to 0.4, since values of t too close to 0 result in positioning the corner too close to the camera, which is a non realistic configuration. We assume that the object is aligned with the room walls and floor, and knowing that the 3D corner is pointing downwards, only the directions along the X and Z axis are left to be determined. Similarly to Fig. 2.11, we find the intersection P_1' between ray p_1c and a plane parallel to the room floor and passing through P_t, and P_1 as the closest point to P_1' on either the X or the Z axis. P_1 and P_2 determine the corner directions in 3D along X and Z, which in this example are the positive X axis and the positive Z axis

Algorithm 2.3: Proposing a frame from a corner

1. Randomly choose an image corner
2. Find the position of the 3D corner, and the room wall s_i it is anchored to as explained in Fig. 2.13.
3. Sample the proposed height of the object h_i from uniform distribution $\mathcal{U}(\frac{r_h}{10.0}, r_h)$, where r_h is the height of the room box
4. Conditioned on s_i, the frame has either negligible width or negligible length. In the first case, sample from $\mathcal{U}(\frac{h_i}{10.0}, 10.0 * h_i)$ to propose the width of the frame w_i, using h_i from the previous step, and set $l_i = \epsilon$. In the second case, set $w_i = \epsilon$, and sample l_i from $\mathcal{U}(\frac{h_i}{10.0}, 10.0 * h_i)$.
5. Shrink any frame on the same wall that collides with the proposed one
6. If the frame does not fit on the wall, expand it. In case the room box changes here, correct for the position of the objects using Eq. 2.15
7. Sample over the object continuous parameters using Diffusion move 1 (delayed acceptance step)
8. Accept or reject the proposed frame by consulting the MH acceptance formula

In both cases (furniture and frames), if the sample gets rejected at the end, we repeat the full procedure until an object gets accepted. We repeat 10 times, and if the final sample gets rejected each time, no object is added to the scene. We do so because the acceptance rate is still quite low despite the bottom-up mechanism, and using many attempts increases the chances of adding an object to the scene.

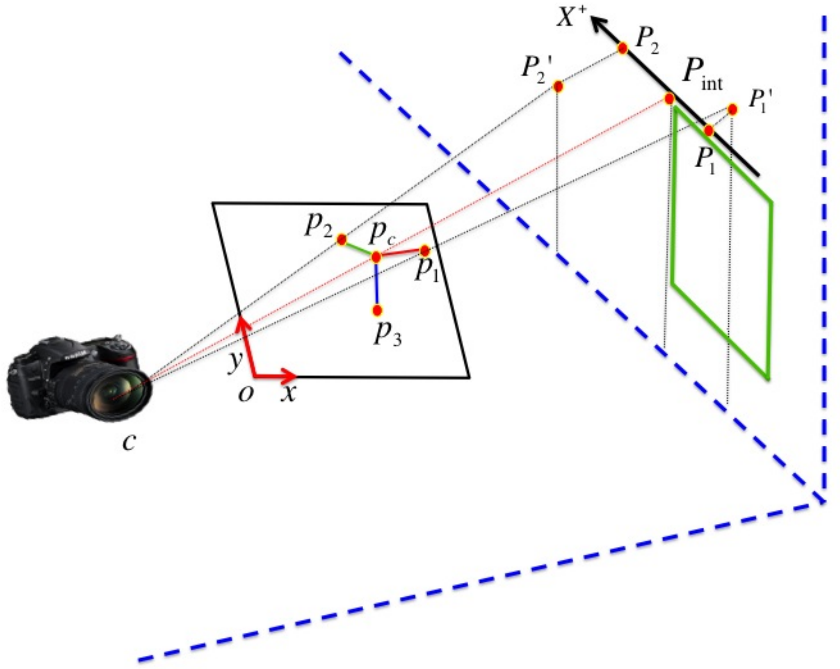

Figure 2.13: Finding the 3D corner of a frame from an image corner. We cast a ray between the camera center c and P_c, and find the intersection P_{int} with the closest wall, which determines the position of the corner in 3D. We know that the frame will expand downwards on the wall, as the corner on the image plane is pointing down, but we have to determine the 3D corner direction along the X axis. We do so by considering P_1' and P_2', which are obtained by intersecting rays p_1c and p_2c with the wall the frame is anchored to. Due to errors in the camera estimate, we cannot expect P_1' and P_2' to lie exactly on X, and we then consider their projections on X, P_1 and P_2. P_1 defines the direction if the distance between P_1 and P_1' is smaller than the distance between P_2 and P_2', we use P_2 otherwise. This is equivalent to choosing the direction that best satisfies the orthogonality constraints. In this example, we used P_1 to set the direction of the frame drawn on the wall

Other jump moves (Jump 2 and 3)

The remaining two jump moves respectively remove an object from the scene (Jump 2), and replace the current scene hypothesis with a new, empty room box (Jump 3). The former simply deletes a randomly selected object, and the proposed change is accepted or rejected using the MH acceptance formula. We also interpret replacing the current scene with a new room box as a jump move. In fact, the current scene hypothesis might contain objects, while the proposed sample will have none. This move is useful when the sampler gets trapped in some regions with low probability from which it cannot recover, and also helps explore more of the output space.

As for the case of objects, a move proposing the room box at a random location would typically result in rejecting the sample. Hence, we developed a procedure to propose the room box from an image corner. Specifically, we rely on the method by Shi et al. (2004), which allows to estimate the 3D orientation of a corner relatively to the camera reference frame, given a projected orthogonal corner and an estimate of the focal length. This is very useful, since, given an image corner and the focal length, we can both estimate the 3D position of the room box corner, and also the camera pose, which in our framework is defined by the pitch ϕ and the roll ψ of the camera, and the yaw γ_r of the room. Notice that the focal length is available from the estimated triplet of orthogonal vanishing points. For the details on how to recover the corner position and the camera pose from the image corner, we refer to the work of Shi et al. (2004).

To propose the room box from the estimated 3D orientation of the corner, we still need to determine its 3D position, which can lie anywhere on the line through the camera center and the corner position on the image plane, as illustrated in Fig. 2.14. However, since we cannot determine absolute positions and sizes from a single image, we arbitrarily position the corner on the line such that the distance between the corner and the camera center is 10 units, and the size of all the objects in the image will be relative to this initial choice.

This still leaves the room box size (h_r, w_r, l_r) as free parameters. Since absolute

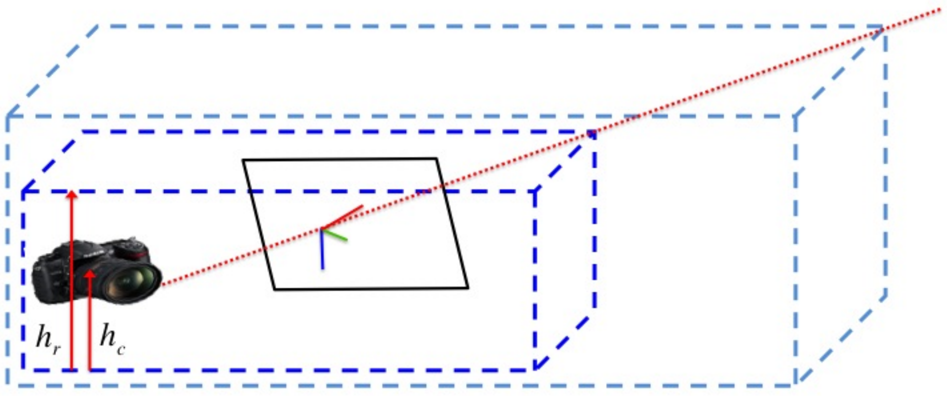

Figure 2.14: Finding the 3D corner of the room box from an image corner. Given an estimate of the focal length and the camera orientation, we cast a ray between the corner on the image and the camera center. We then position the 3D room corner along this line. Different positions generate different room boxes, as illustrated by the two dashed examples above. Once the 3D position is chosen, we set the room height h_r such that the ratio between the height of the camera from the floor h_c and h_r falls in a range of plausible values (see text).

sizes cannot be used when reconstructing from a single image, we set the room height such that the ratio r_{hc} between the height of the camera from the floor h_c and h_r (Fig. 2.14) takes plausible values. Specifically, we sample a plausible value for r_{hc} from $\mathcal{U}(0.3, 0.7)$, which prevents the camera from being too close to either the floor or the ceiling. Given the 3D position (x_c, y_c, z_c) of the room box corner, and knowing that the camera is at the origin of the world reference center, we can compute h_r from r_{hc}. In fact $h_r = \frac{|y_c|}{r_{hc}}$ if the estimated 3D corner is part of the room floor ($y_c < 0$), or $h_c = \frac{y_c}{1-r_{hc}}$ if the 3D corner is part of the room ceiling ($y_c > 0$).

We then set both width and length of the room box by sampling from $\mathcal{U}(0.1 *$

$h_r, 10{*}h_r$), which constrains the room box size to plausible values. Before consulting the Metropolis Hastings acceptance formula, we use delayed acceptance as in the case of object proposal, by briefly sampling over the continuous parameters of the room box using Diffusion move 2.

The procedure just discussed allows to propose a room box and the camera pose from a randomly selected image corner. However, it also proves useful for initializing the camera and room box parameters at the beginning of the inference. Specifically, we propose both the room box and camera pose from each of the detected image corners using the method just describe, and use the pair with the highest posterior to initialize the sampler. Notice that we are not committing to this partial solution, as the other moves using during inference will modify the continuous parameters of this starting room box, and potentially replace it with a completely different one.

Summary of the inference process

We now provide an overview of the entire inference process using pseudo code. This allows us to summarize how we use the sampling moves discussed so far, which are the building blocks of our inference engine.

The whole inference process works as follows.

Algorithm 2.4: Full inference process

```
1. Detect straight line segments on the image plane
2. Estimate a triplet of vanishing points, use them to initialize the focal length
   $f$ of the camera
3. Use the vanishing points and the line segments to find image corners
4. Initialize the remaining camera parameters and the room box from the image
   corners. This provides the initial sample $\theta_0$.
5. Repeat $k = (1, .., K)$ times
```

(a) Generate a new sample θ_k conditioned on the previous sample θ_{k-1} by using one of the following moves, chosen with equal probability

- Jump 1: add an object to the scene (furniture of frame)
- Jump 2: remove an object from the scene
- Jump 3: change the room box
- Diffusion 1: pick a random object and sample over its parameters
- Diffusion 2: sample over the room box parameters only
- Diffusion 3: sample over room box and camera parameters

(b) Reject samples that violate the constraints on the camera position (camera outside the room or inside an object).

6. Return the sample with the highest posterior

2.4 Results

The method proposed in this chapter provides a geometric reconstruction of the scene in terms of simple 3D blocks, which approximate the room box and objects in it. To evaluate the accuracy of this reconstruction we consider the error on the room box estimation, which is a standard criterion in the field of indoor scene reconstruction (Hedau et al., 2009; Wang et al., 2010; Lee et al., 2010). This is sometimes referred to as room layout error.

This criterion compares the projection of the estimated room box against the ground truth, where each pixel was labeled according to the room face it belongs to (ceiling, floor, left, middle and right wall), by computing the ratio of misclassified pixels. The ground truth was prepared by considering the room as empty.

In this chapter, we do not evaluate on the predicted furniture objects and frames directly. However, the room box error is also considered a proxy for measuring the accuracy of the predicted objects, since the room box boundary is often occluded, and obtaining good performances on this criterion requires the algorithm to have some sense of the occluding objects.

We evaluate on two standard datasets used in this field, the Hedau dataset (Hedau et al., 2009), consisting of 314 color images of indoor scenes, and the UCB dataset (Yu et al., 2008), consisting of 340 black and white images. Hedau et al. (2009) provided the ground truth room box labels for the entire Hedau dataset, which are available online[2]. Hedau et al. (2009) introduced this dataset to evaluate their approach to room box estimation, which was trained on a split of 210 images and evaluated on the remaining 104. Other methods also evaluate on the test split only for proper comparison, and also the results we report on the Hedau dataset are relative to this test set of 104 images only. We manually annotated the room box labels for all the UCB images, and this data is available on our website[3]. When we report the room box error for a dataset, we average this score over all the images in it.

In Table 1 we first evaluate a version of our algorithm where we only fit the room box and the camera to an image, without allowing any object in the scene. We do so by initializing the room box and the camera parameters as in our full algorithm, and then alternating continuous sampling over room box and camera parameters (moves Diffusion 2, Diffusion 3), and occasionally proposing a new room box (move Jump 3), for 100 iterations. We then compare this to our full algorithm where we allow a maximum of 10 objects per scene. We can see that adding objects reduces the room box error, as it explains occlusions. This is also illustrated with a few qualitative examples in Figure 2.17.

In Table 2.1 we also compare to other relevant work in scene understanding. We can see that our results approach those of Hedau et al. (2009) and Wang et al. (2010), which use 2D appearance models trained on ground truth room labels, while our method only uses edges. We also report the results of Lee et al. (2010), whose 3D geometric model is very similar to ours, as they approximate both the room box and objects in it with simple 3D blocks. Also their method considers more

[2]`http://vision.cs.uiuc.edu/~vhedau2/Research/research_spatialLayout.html`

[3]`http://kobus.ca/research/data/CVPR_11_room/index.html`

Table 2.1: Performance analysis based on room box error.

	Actual	
Method	Hedau dataset	UCB dataset
Only room box	25.9%	24.8%
Room box and objects	**24.2%**	**23.3%**
Hedau et al. (2009)	21.2%	NA
Wang et al. (2010)	20.1%	NA
Lee et al. (2010)	16.2%	NA
Del Pero et al. (2013)	**12.7%**	**14.0%**

image features, which allow them to reduce the room box error. For completeness, we also report the results of our most recent work (Del Pero et al., 2013), which is the state-of-the-art method on this evaluation criterion. This will be discussed in Chapeter 4. Last, some representative qualitative results are shown in Fig 2.15, while some common failures are discussed in Fig. 2.16.

2.5 Discussion

We have developed a generative modeling framework for understanding Manhattan rooms, and an efficient method for simultaneously fitting the camera and a 3D scene model of unknown structure and dimensionality to image data. Despite using only edge information, we achieve results approaching those of others methods on the task of estimating the main surfaces defining the indoor scene (walls, ceiling and floor), despite using a more limited set of image features.

We emphasize that the key contribution is the alternative top-down Bayesian approach, which is likely to prove more powerful as it integrates more information, such as more sophisticated object models and more robust likelihood functions. Importantly, having separated the modeling and the inference in a Bayesian fashion makes it easy to integrate new sources of evidence. The core inference engine discussed here can in fact handle (with minor modifications), the additions needed to overcome the main limitations of this basic version of the model, as discussed in the

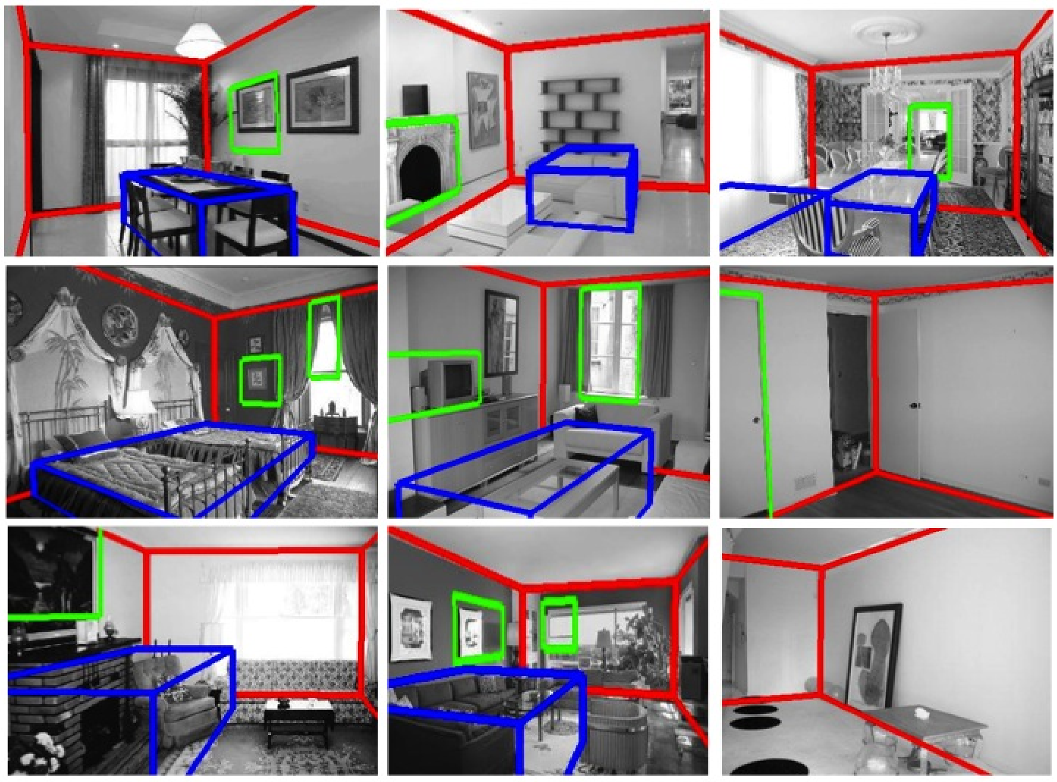

Figure 2.15: A few promising examples of full scene reconstructions estimated by our algorithm. We show in red the projection of the estimated room box, in blue the furniture objects, and in green the frames.

next chapters.

A first main limitation is that the simple 3D geometric constraints used here, such as objects not intersecting each other, are not enough to avoid unrealistic configurations, for example objects with improbable size (Fig. 2.16, top right). Second, while modeling each object with a single 3D block allowed us to improve on estimating the room surfaces, more detailed geometry is needed for better scene reconstruction and understanding. Third, a likelihood using only edges is not robust enough in several cases (Fig. 2.16, bottom row), and we need to integrate other image features that proved useful in this domain, such as geometric context (Hoiem et al., 2005a; Hedau et al., 2009).

Despite these limitations, this is a promising step towards our goal of providing a global understanding of an image. In fact, we found that inferring the room box and the objects in it jointly improves on estimating the room box alone (Table 2.1). Consider for example the top row of Fig. 2.17. Here, estimating the room box alone fails, because the floor is completely occluded, and the predicted floor edges "latch" to the top of the two couches (left). Adding objects, such as the green frame (middle) helps explain occlusions, and allows moving the floor to the correct position (right).

Similarly, joint inference of the camera and scene parameters can help improve the initial camera estimate. In fact, the projected edges of the objects added to the scene provide additional evidence towards finding the correct camera pose and its focal length. This is illustrated in the bottom row of Fig. 2.17, where adding objects (middle) improves the camera and the room box predicted without using objects (left).

These two examples demonstrate our intuition that a global interpretation of an image requires finding the elements in it by also considering their interplay. In the next chapters, we extend our model with the goal of finding more comprehensive and realistic scene interpretations. In Chapter 3, we introduce prior distributions on an object's size and position in the room, which constrain the predicted scenes to more plausible 3D configurations. In Chapter 4, we discuss the advantages of more realistic geometric models than blocks, such as tables with legs or couches with backrests.

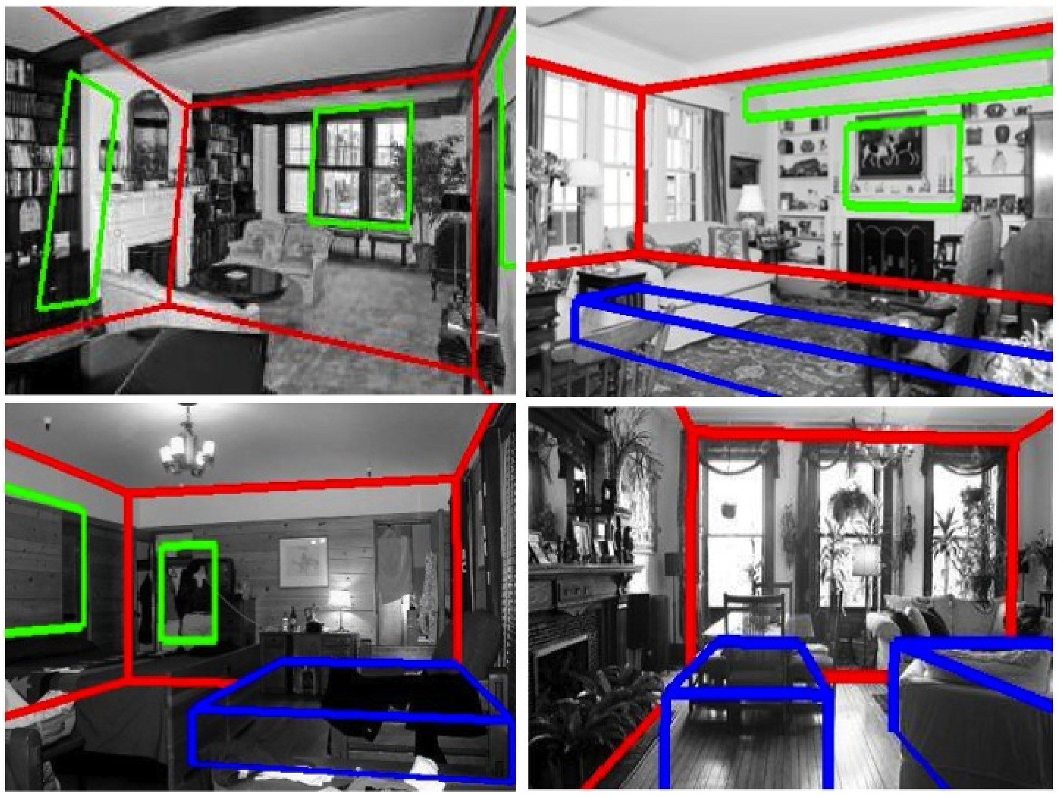

Figure 2.16: Some typical failures. Top left: The algorithm could not recover from a large error in the initial camera estimate. Top right: We often propose objects with unrealistic size, since there is no component in the model constraining object parameters to plausible values. Bottom row: our likelihood function is edge based, and objects that explain image edges very well might not correspond to actual pieces of furniture in the room. In the bottom left image, the contour of the predicted blue box is matched to edges generated by multiple objects in the image (the bed and the armchair).

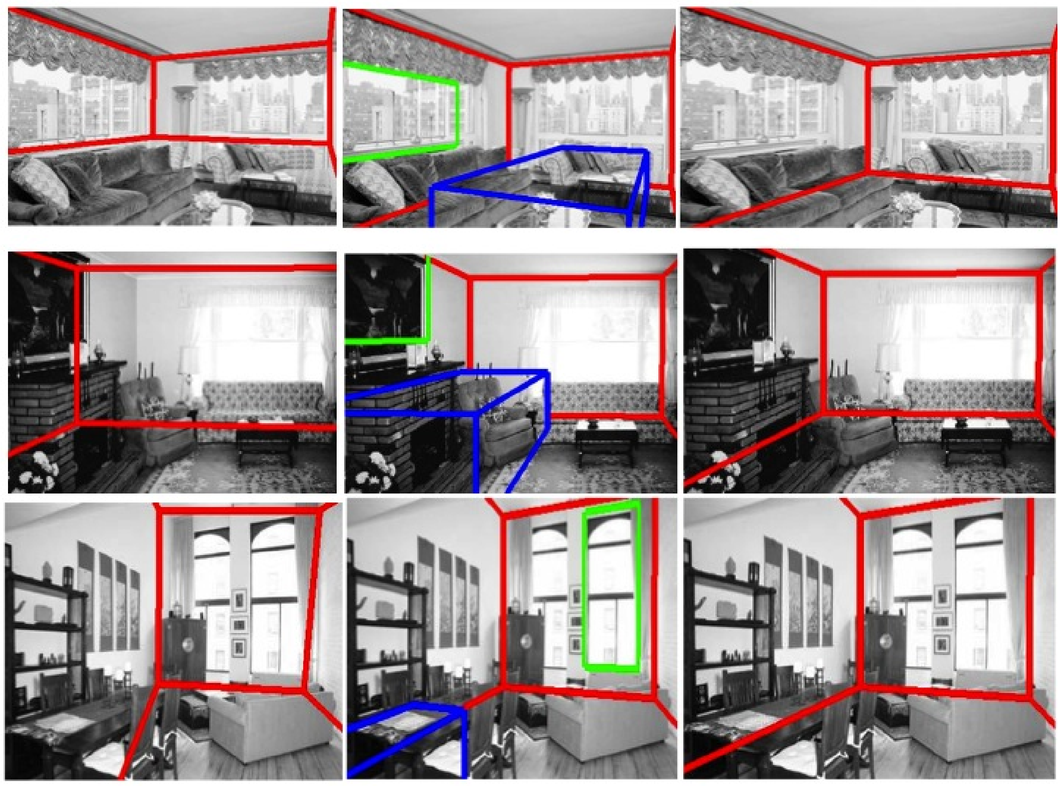

Figure 2.17: Benefits of joint inference of room box, objects and camera. Here we show the room box found by fitting room box and camera only (left), the full scene reconstruction when we allow objects in the scene (middle), and the room box corresponding to this full scene reconstruction (right). Top two rows: Adding objects explain occlusions and improves the estimate of the room box. Bottom row: Adding objects helps improve the estimate of the camera. Here, the projected edges of the objects provide additional evidence to find the correct camera parameters

CHAPTER 3

Modeling realistic indoor objects with priors on 3D size and position

3.1 Introduction

In this chapter, we focus on using statistics from the 3D world to encourage a coherent parsing of the overall scene. More specifically, we focus on how modeling the 3D geometry and location of specific objects is helpful for indoor scene understanding from images[1]. A key intuition is that we can discriminate between many indoor object categories using only gross geometry. Beds are in fact much wider than they are tall, while wardrobes are typically tall and narrow (Fig. 3.1). Similarly, position with respect to the overall indoor scene provides hints to object identity. For example, beds are typically against a wall, while tables are likely found in the center of the room.

Using the identity of an object's to inform the inference of its 3D geometry, and, conversely, guessing the object's identity from the estimated 3D geometry, are indeed powerful ideas. However, this is a classic chicken-and-egg dilemma, as we need to identify an object in order to reason about its geometry, and vice versa, an estimate of the object's geometry is required to find its identity. We thus propose to simultaneously identify objects while fitting their size and location, and show how these two processes help each other, and also improve the global interpretation of the scene.

We demonstrate this idea by using the simple geometric model discussed in Chapter 2, where each object in the room is approximated as a single block. However, we impose further structure by introducing the notion of an object's type, such as bed or couch. Conditioned on its type, we use prior distributions to evaluate how realistic the hypothesized 3D size and position of the object are. Importantly, this

[1]This is the topic of our 2012 CVPR paper (Del Pero et al., 2012)

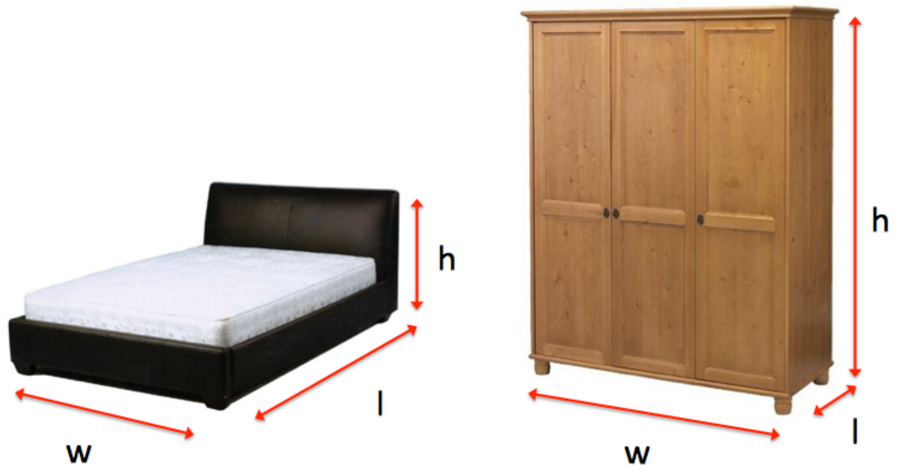

Figure 3.1: Distinguishing objects using their size. The ratio between the height of an object and the largest between its width and length varies considerably between beds and cabinets. In this example, also the ratio between width and length is quite discriminative, as the base of the bed on the left is roughly square, while the base of the wardrobe on the right is more elongated.

kind of prior information naturally integrates in our Bayesian framework.

The main difference with respect to the model discussed in Chapter 2 is that each block in the scene is now labeled (e.g. bed, table, door), and this label constrains its size and position in the room, while before a block was simply used to model a region of the 3D space occupied by a generic object. This introduces additional variables in the model, namely a discrete variable per object in the scene specifying its type, which also need to be inferred from the image data together with the other parameters of the model. Notice also that, thanks to these labels, our model can now predict the identity of the objects in the scene, and can be evaluated on the task of object recognition.

We emphasize that the prior distributions we use are defined over the 3D space, where an object's structure and size have much less variance than if we were to model them on the image plane. For example, using the real size of the object in the 3D world is less ambiguous than considering its 2D size on the image, where objects farther away from the camera are smaller. Similarly, the position of an object in the scene (e.g. a bed is typically against a wall) can be modeled more effectively in 3D than in 2D. Further, learning the parameters of the priors is easier in 3D (we use the text available from furniture catalogs), while it is more challenging in 2D.

Another advantage of using priors is that they can be used to inform the inference process in a top-down fashion. For example, instead of proposing adding a block with random size, we sample its size from the prior. This increases the chances that the sample will be accepted, and makes the inference more efficient.

In what follows, we discuss our strategy for using prior distributions on the 3D geometry of the objects. In Sec. 3.2, we extend the basic version of our model (Chapter 2) to incorporate such priors, and also introduce new image features to make the likelihood function more robust. In Sec. 3.3 we describe the inference process, focusing on how to generate top-down samples from the prior. Here, we also introduce new techniques to speed up the basic inference process from Chapter 2. These include using multiple threads, and allowing them to exchange information. Last, we discuss comprehensive evaluation (Sec. 3.4) showing the benefits of the proposed innovations. In this case, we not only evaluate on the standard room box error criterion, but also on object recognition.

3.1.1 Related work

We now discuss the work relevant to the ideas developed in this chapter, which mostly focuses on using priors on 3D size and position. For a more general overview of the literature on 3D representation for object recognition, and its differences with respect to 2D representations, we refer to Sect. 1.1.2.

Hedau et al. (2010) used the estimated room box to provide 3D context for estimating beds in the scene. The room box provides strong cues on the size and position of the bed, whereas traditional 2D methods (Dalal and Triggs, 2005; Felzenszwalb et al., 2009) cannot make any assumption, and have to rely on 2D sliding windows. Instead, Hedau et al. (2010) slide a 3D cuboid in the hypothesized room box, and evaluate its projection against image appearance. They constrain the 3D size of the box to two possible aspect ratios, which are conditioned on the size of the room box. This is closely related to our use of priors to characterize the size of a 3D object. However, Hedau et al. (2010) use only two fixed aspect ratios, while we model the 3D size of the objects statistically. Second, we fit the objects together with the room box, and this helps the inference of the latter, while Hedau et al. (2010) first estimate the room box and then the objects. Third, we demonstrate our method by distinguishing several types of objects, as opposed to only beds.

In their more recent work, Hedau et al. (2012) extended the method to detect 3D generic cuboids in the context of the room box. However, also in this case the 3D size of cuboid is constrained to a fixed set of aspect ratios, and it is not modeled statistically. Additionally, this work uses a cuboid to model any generic object in the scene, which is similar to the method discussed in Chapter 2, whereas the method we propose here distinguishes among object classes, such as beds and tables.

Another difference with respect to both these methods (Hedau et al., 2010, 2012) is that the appearance of their cuboid hypotheses are first evaluated locally and independently, and then re-scored based on the surrounding context and the total amount of overlap in 3D. Instead, our likelihood scores evaluates the global fit of the entire scene hypothesis and, for now, we do not train specific appearance models for each of the object categories, such as the "bed" model proposed by Hedau et al. (2010).

The way we enforce the consistency on the size of the overall 3D scene statistically relates to the work of Hoiem et al. (2006) on outdoor scenes. However, their method relies on 2D detectors to generate object candidates bottom-up, while we also generate 3D object hypotheses top-down, by sampling from the priors and conditioned on the current scene hypothesis. Further, we combine these top-down cues with bottom-up proposals from image corners (Sect. 3.3), and the integration of these two complimentary types of proposals is also used in the work of Han and Zhu (2005); Zhu et al. (2006).

Additionally, the proposed extensions to the likelihood function use two features developed specifically for images of indoor scenes, orientation maps (Lee et al., 2009) and geometric context (Hoiem et al., 2005a). A difference with respect to previous work using these features (Schwing et al., 2012; Lee et al., 2010; Hedau et al., 2009) is the way we integrate them in a probabilistic framework.

3.2 A scene model with priors on 3D size and position of objects

We rely on the model discussed in Sect. 2.2, where model parameters $\theta = (s, c)$ include scene parameters, s, and camera parameters, c. The camera models the perspective transformation that generated the image (Eq. 2.3). The scene parameters include the room box r, and a set of objects $(o_i, ..., o_n)$, where the number of objects is not known a priori (Eq. 2.4).

We retain the simple 3D geometry introduced in Chapter 2, where the room box r (Eq. 2.5) and each object o_i in the scene are approximated with a single 3D block. However, we augment the object model by introducing variable t_i. (Eq. 2.6) then becomes

$$o_i = (s_i, x_i, y_i, z_i, w_i, h_i, l_i, t_i) \quad . \tag{3.1}$$

The object type t_i is a discrete variable indexing over the available object categories. In this work, we use a set of four categories for furniture objects (bed, cabinet, couch, table), and three frame categories (door, picture frame, window), but our formulation is generic and can be extended to a larger number of categories.

Since all the objects share the same geometric representation, only their size and position can be used to distinguish between different categories. We do so with category-dependent priors, that inform on where objects in a specific category tend to be, and on their typical size. These category-dependent priors allow us to estimate the most likely class for a given object, based on its position and size only. We now discuss how to construct such priors.

Model priors

We develop a general formulation for a prior for a specific object of category $t_i = \tau$, which exploits the intuition that some geometric properties of objects are highly discriminative (Fig. 3.1). Given an object o_i, we denote its prior as $\pi(o_i|t_i = \tau)$, which is conditioned on the type τ of the object. Similarly, we define a prior on the room box and the camera $\pi(r, c)$ to constrain also those parameters to plausible configurations.

The full prior distribution over model parameters is defined as

$$p(\theta) = \pi(r, c) \prod_{i=1}^{N} \pi(o_i|t_i = \tau) \quad , \tag{3.2}$$

where the contributions of the object priors are assumed to be independent of each other, and also of $\pi(r, c)$. This differs from Chapter 2, where we only used a strong priors preventing objects from intersecting each other and the room box. Notice that these containment constraints are enforced also in this version of the model. We now discuss each component of the prior in detail.

Object priors

Object priors are constructed to exploit geometric cues such as those illustrated in (Fig. 3.1). However, priors cannot be defined in terms of absolute sizes or positions in the room, since we are reconstructing from monocular images. This means that there is an overall scale ambiguity, and priors must be defined relatively to the overall scene size. For example, we cannot use the statistics over the absolute height of an object (e.g. the average height of a bed is 2 ft), but we could use a distribution on the height of the object relatively to the height of the scene (e.g. the ratio between the object's height and the room box's height).

Given an object o_i defined in terms of its size (w_i, h_i, l_i), and a room with dimensions (w_r, h_r, l_r), we use priors on the following quantities, which we found to be particularly discriminative:

- ratio between the object height and its largest other dimension $r_{i1} = h_i/max(w_i, l_i)$ (Fig. 3.1). This helps distinguish between categories that are taller than they are wide, such as wardrobes, and short and wide objects, such as beds. Notice that we use this formulation $max(w_i, l_i)$ because we do not know in advance which one is the largest.

- ratio between the object long and short dimensions, $r_{i2} = max(w_i, l_i)/min(w_i, l_i)$ (Fig. 3.1). Again, we we do not know in advance which dimension is the largest. This quantity discriminates between furniture with a roughly square base (e.g. chairs), and furniture with an elongated base (e.g. couches). This component is not used for frames, since either their width or their length is negligible.

- ratio between room height and object height $r_{i3} = h_r/h_i$. Intuitively, the height of a bed is quite small with respect to the room height, whereas the height of a wardrobe or of a door is quite large (Fig. 3.2).

- whether the object is against a wall or not. This is based on the intuition that some objects tend to be against a wall (e.g. beds) more than others (tables). For frames, we use whether or not the frame touches the floor. For example, doors touch the floor, while windows typically do not. We consider an object o_i against a wall (floor) if the distance to the closest wall (floor) is less than

10% of $max(w_i, l_i)$.

The first two ratios carry information on the object structure, and do not depend on the scene, while the last two encode information on the size and position of an object relatively to the room box. We assume that the first three quantities follow a normal distribution

$$\pi_j(o_i|t_i = \tau) = \mathcal{N}(r_{ij}; \mu_{\tau_j}, \sigma_{\tau j}) \quad , \tag{3.3}$$

for $j = 1, 2, 3$. Each category τ has different $(\mu_{\tau j}, \sigma_{\tau j})$, and for object o_i we use the prior distribution for the category it belongs to, denoted by t_i. Notice that from now on we will use the shorthand $\pi_j(o_i)$ for $\pi_j(o_i|t_i = \tau)$. Last, d_i follows a Bernoulli distribution $\pi(d_i)$. Given an object o_i, we combine the components of its prior probability by assuming independence

$$\pi(o_i) = \pi(d_i) \prod_{j=1}^{3} \pi_j(o_i) \quad . \tag{3.4}$$

Room box and camera priors

The room box is defined in terms of the center position in 3D (x_r, y_r, z_r) and its width, height, and length (w_r, h_r, l_r). First, we define a prior over the ratio between the long dimension to the short dimension, with

$$r_{r1} = \frac{max(w_r, l_r)}{min(w_r, l_r)} \quad . \tag{3.5}$$

Again, we do not know in advance which dimension is the largest. We also use a prior on the ratio of the long dimension to the height

$$r_{r2} = \frac{max(w_r, l_r)}{h_r} \quad . \tag{3.6}$$

The prior distributions over these two quantities are set to be relatively non informative, but help reduce the time spent in regions of the sampling space with low probability, especially during the early stages of the inference process. For example,

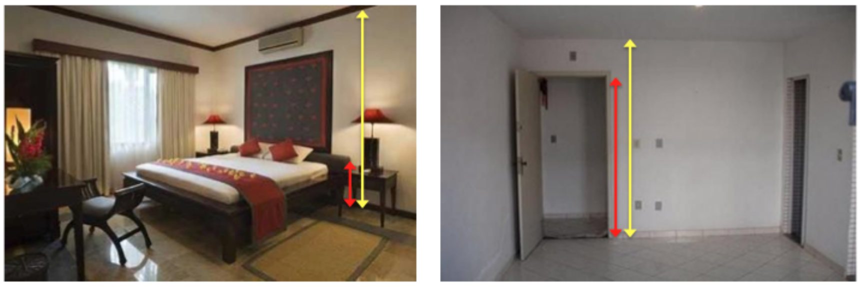

Figure 3.2: Learning the ratio between room height and object height. This quantity is useful to distinguish between indoor object categories, and we estimate its statistics from training images. We use the ratio between the two arrows, provided that the object is against or close to a wall. While ratios of lengths of collinear segments are normally not preserved by projective transformations (only affine), in indoor scenes the vanishing point for vertical segments is usually at infinity, and this method provides a reasonable approximation

the prior on r_{r1} prevents having unrealistic rooms that are 100 times larger than they are wide. Similarly, these two components also prevent the sides of the room that are not visible in the image from expanding arbitrarily, and this also makes the inference more efficient.

We set both parameters to be normal distributions independent from each other

$$\pi(r) = \mathcal{N}(r_{r1}, \mu_{r1}, \sigma_{r1})\mathcal{N}(r_{r2}, \mu_{r2}, \sigma_{r2}) \quad . \tag{3.7}$$

This defines the prior over the room box.

Additionally, we found that the camera height from the floor c_h is a particularly indicative property in indoor scenes, as small variations in this quantity result in major changes in the image plane. Intuitively, pictures of indoor images are rarely taken by putting the camera close to the floor or to the ceiling. Since we cannot use absolute sizes, the prior is defined on the ratio r_{ch} between camera height and room height

$$\pi(c_h) = \mathcal{N}(r_{ch}, \mu_{ch}, \sigma_{ch}) \quad . \tag{3.8}$$

Notice that r_{ch} can be computed as

$$r_{ch} = \frac{r_{coord}(0,0,0) - (\frac{r_h}{2})}{r_h} \quad , \tag{3.9}$$

where $r_{coord}(0,0,0)$ is the position of the camera in room coordinates computed using Eq. 2.7 (remember that the camera is positioned at $(0,0,0)$ in the world reference frame). $\frac{r_h}{2}$ is the position of the floor, also in room coordinates.

Last, we combine $\pi(c_h)$ and $\pi(r)$ by assuming independence

$$\pi(r,c) = \pi(r)\pi(c_h) \quad . \tag{3.10}$$

Setting prior probabilities from data

As mentioned above, the first two components of the object prior do not depend on the scene. For each category τ, we set $(\mu_{\tau 1}, \sigma_{\tau 1}, \mu_{\tau 2}, \sigma_{\tau 2})$ using fifty random examples selected from online furniture and appliances catalogs. We recorded their dimensions, provided in the text description, and the means and variances of the relevant ratios. We used the Ikea catalog[2] for beds, couches, cabinets and tables, and the Home Depot catalog[3] for windows, doors and picture frames.

Setting the parameters for the remaining two components of the object priors is more challenging, since they relate the size of an object category to that of the room, and this information is not available in furniture catalogs. In this case, we use image data, and set (μ_{c3}, σ_{c3}) as explained in Fig. 3.2. We also use images to set $\pi(d)$, which we approximate as the frequency at which an instance of an object of category τ is against a wall, or floor if it is a frame. For training, we used the images in the training split of the Hedau dataset Hedau et al. (2009), omitting images where we could not clearly tell whether a piece of furniture was against the wall or not.

Last, we set the parameters of the priors on camera and room box from training images. We manually fit an empty room box and the camera to images in the Hedau training set, from which we set the prior parameters. This is discussed in more detail in Appendix B.

[2]`http://www.ikea.com/us/en/catalog/categories/departments/bedroom/`

[3]`http://www6.homedepot.com/cyber-monday/index.html`

Augmenting the image model

The image likelihood used in Chapter 2 only uses edges, and we saw in Sec. 2.4 that this feature is often not informative enough. Recent work (Hedau et al., 2009; Lee et al., 2010; Schwing et al., 2012) used two new images features that proved very useful in the domain of indoor scenes, orientation maps (Lee et al., 2009) and geometric context (Hoiem et al., 2005a). Here, we show how to integrate them in our image likelihood.

Orientation maps (Lee et al., 2009) were specifically developed for indoor images, that satisfy the Manhattan World assumption that most surfaces are aligned with three principal, orthogonal directions. Thus, most pixels in the scene are generated by a plane aligned with one of these directions, and we can estimate which one using a geometric algorithm introduced by (Lee et al., 2009). We used their MATLAB implementation to compute orientation maps for the images in the datasets we use for testing.

We compare the pixel orientation O_d detected from the image plane with the orientation surfaces O_θ that we can easily generate from a model hypothesis. We then approximate $p(O_d|O_\theta)$ by the ratio between the number of pixels where the orientation detected on the image plane agrees with the orientation predicted by the model, and the total number of pixels.

Geometric context was instead introduced by Hoiem et al. (2005a) for understanding outdoor scenes. This method uses a probabilistic classifier trained on image appearance (color and oriented gradient) to estimate the geometric class of image superpixels. The original method (Hoiem et al., 2005a) used geometric classes to describe outdoor scenes, such as “ground” and “vertical surface”. Hedau et al. (2009) then modified this model for indoor scenes, by using a different set of geometric classes (left wall, middle wall, right wall, floor, ceiling, object).

We use the MATLAB code and the pre-trained classifier by Hedau et al. (2009) to compute geometric context labels from images. For each pixel p_k, this provides a probability distribution $gc_k = [gc_{k1}, ..., gc_{k6}]$ over the six indoor geometric classes.

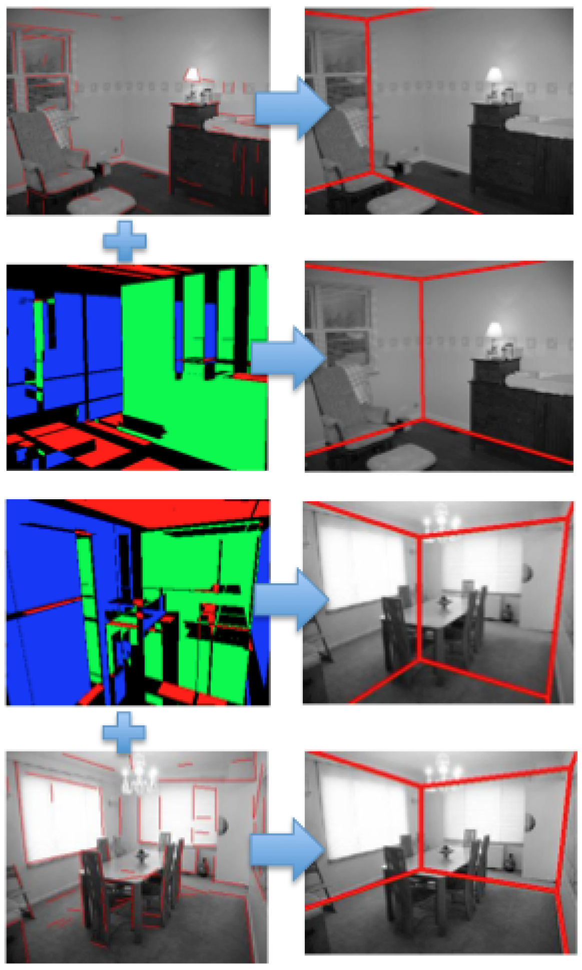

Figure 3.3: Integrating edges and surface orientation in the likelihood function. Here, we visualize orientation maps (second and third row, left) by drawing pixels assigned to one of the Manhattan directions with three different colors: red pixels correspond to horizontal surfaces, blue and green to the two types of orthogonal, vertical surfaces. We show in red the edges found by the edge detector (first and fourth row, left). Faint wall edges are often missed by the edge detector (top left), and the edge likelihood alone would provide the wrong room box, by “latching” the wall edge to the window (top right). However, in this case the orientation surfaces help converge to the right solution (second row). Conversely, mistakes in the orientation map estimation can be overcome by relying on edge information (third and fourth row).

Also in this case we use the model hypothesis to predict the geometric classes we expect to find on the image. Given the label l predicted by the model for pixel p_k, we define $p(gc_k|p_k) = p(gc_k|p_k = l) = gc_l$, and

$$p(GC|\theta) = \frac{\sum_{p_k \in I} p(gc_k|p_k)}{size(I)} \quad , \tag{3.11}$$

where we average the contributions of all pixels ($size(I)$ is the number of pixels in the image). Since the available classifier was trained against data where only furniture was labeled as objects, and not frames, we consider frames as part of the wall they are attached to, and not as objects when we evaluate on geometric context.

Assuming independence among edges, orientation maps and geometric context, we modify our likelihood function as follows

$$p(D|\theta) = p(E_d|E_m)p(O_d|O_m)^{\alpha}p(GC|\theta)^{\beta} \quad . \tag{3.12}$$

α and β weigh the importance of the orientation and geometric context likelihoods, relatively to the edge likelihood. We set $\beta = 12$ and $\alpha = 6$ by running our algorithm on the training portion of the Hedau dataset (Hedau et al., 2009). Here, we used a coarse grid search over α and β, with a step of 2, using the room box error (Sect. 2.4) as an objective function.

We illustrate some of the advantages of integrating geometric context and orientation maps in Fig. 3.3 and 3.4. Errors in the edge detection process can be fixed using the other two features, and vice versa. It is also important to notice that the algorithms for estimating both orientation maps and geometric context need an estimate of the three vanishing points corresponding to the Manhattan World directions. In both cases, we use the vanishing points found with the procedure discussed in Sect. 2.4. Unfortunately, large errors in this initial estimate of the vanishing points compromise the computations of both geometric context and orientation maps. However, edges are detected without using vanishing point information, and this is the only feature that can potentially improve the camera fit when we start from a poor initial estimate of the vanishing points.

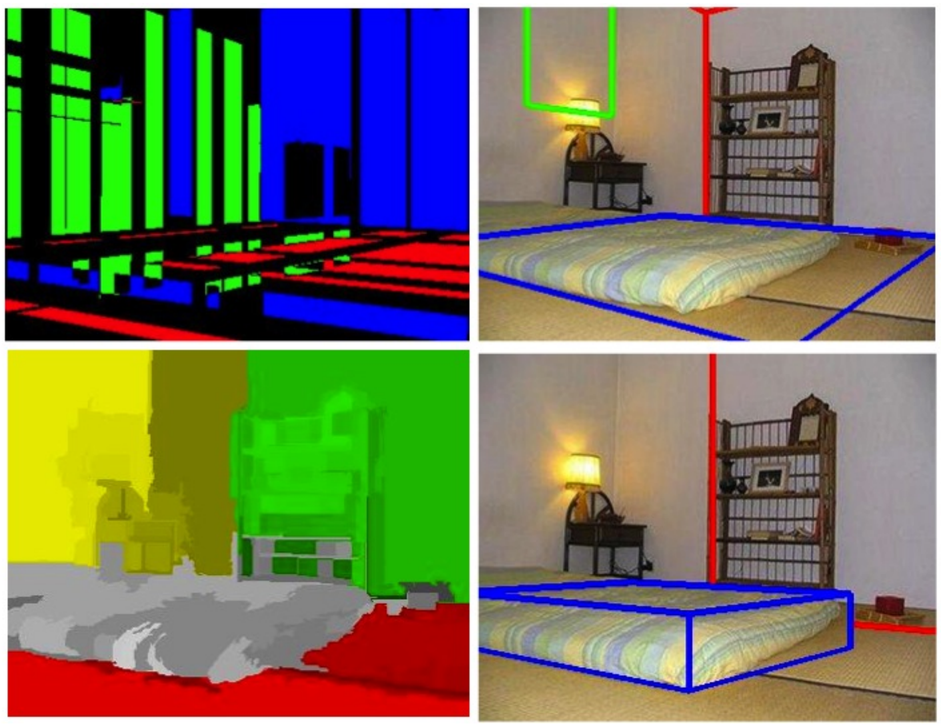

Figure 3.4: Advantages of adding geometric context to the likelihood function. Top left: the estimated orientation map labels for the image in the right column. For the coloring convention, see Fig 3.3. Top right: the predicted scene using only edges and orientation maps. Bottom left: the geometric context labels found using the method by Hedau et al. (2009). We draw in red the pixels with a high probability of being "floor", and use gray for "object", green and yellow for two different kind of walls. Bottom right: the predicted scene model by adding geometric context to the likelihood. The red pixels in the orientation map are ambiguous (top left), as they could have been generated by the floor or by the top horizontal surface of an object, but geometric context helps solve this ambiguity.

We use the inference engine from Chapter 2 to search the output space. In this case, the types of the objects in the scene are additional discrete variables that needs to be inferred. Another difference is that the Bayesian posterior uses a prior distribution and a more complex likelihood. When we evaluate the posterior during inference, we use this new version.

To handle the object type variables, we introduce an additional jump move (Jump 4) to switch the type of an object in the scene, say from couch to table, or from door to window. Notice that changing an object's type also changes the prior distribution used to evaluate its size and position. When we execute this move, we randomly select a new object in the scene, and then randomly choose a new type from the set of available categories. The type change is then accepted or rejected by consulting the Metropolis Hastings acceptance formula.

Further, when we propose adding an object to the scene we also propose its type. We modify Jump Move 1 (Algorithm 2.2) to add an object of a specific type, as opposed to a generic block. Given the proposed type, we sample the object's size from the associated prior, which results in an object proposal with more plausible size, and more likely to be accepted. The priors thus provide top-down cues to drive the inference towards more realistic configurations.

We now discuss this modification to Jump Move 1 in detail. Then, we discuss a further modification to the inference engine in Chapter 2, by introducing a more efficient initialization, and by making the process multithreaded. The latter modification allows to search more of the output space, and we also show how to exchange information among the threads. Last, we provide a summary of the inference algorithm after all these modifications.

Adding objects to the scene using priors

We modify the standard move for adding an object to the scene (Jump Move 1, see Algorithm 2.2) in two ways: 1) we randomly select the type of the object that will be added; and 2) we propose its size from the corresponding prior.

The modified jump move for the case of furniture objects is summarized below.

Algorithm 3.1: Proposing furniture from a corner and the priors

1. Randomly choose an image corner, and determine whether it is pointing up or down (Fig. 2.11)

2. Determine the object category τ by randomly choosing from the available classes (four in this case: bed, cabinet, couch, table). Each class has the same probability.

3. If the corner is pointing up, find the position of the 3D corner on the floor as explained in Fig. 2.11, otherwise find the position of the 3D corner as explained in Fig. 2.12.

4. Sample $\mathcal{N}(r_{i3}; \mu_{\tau_3}, \sigma_{\tau 3})$ to propose the ratio r_{i3} between the room height and that of the proposed object height h_i. Set $h_i = \frac{h_r}{r_{i3}}$, where h_r is the current height of the room box.

5. Sample u from uniform distribution $\mathcal{U}(0, 1)$. If $u > 0.5$, set the width w_i of the object to be larger than its length l_i, if $u \leq 0.5$ set l_i to be larger. The next two steps are defined for the case $u > 0.5$, and can be adapted to the opposite case by swapping w_i with l_i.

6. Sample from $\mathcal{N}(r_{i1}; \mu_{\tau_1}, \sigma_{\tau 1})$ to propose the ratio r_{i1} between h_i and its largest dimension w_i. Set $w_i = \frac{h_i}{r_{i1}}$ using h_i from the previous step.

7. Sample from $\mathcal{N}(r_{i2}; \mu_{\tau_2}, \sigma_{\tau 1})$ to propose the ratio r_{i2} between the object largest and shortest dimensions (w_i and l_i). Set $l_i = \frac{w_i}{r_{i2}}$ using w_i from the previous step.

8. Shrink any furniture object in the scene colliding with the proposed one

9. In case the object does not fit in the room, expand the room box. This might involve lowering the position of the floor, raising the ceiling, or increasing the room width and/or length. In case the room box changes here, we correct for the position of the objects using Eq. 2.15
10. Sample over the object continuous parameters using Diffusion move 1 (delayed acceptance step)
11. Accept or reject the proposed object by consulting the MH acceptance formula

The modified version for frames works as follows.

Algorithm 3.2: Proposing a frame from a corner and the priors

1. Randomly choose an image corner
2. Determine the object category τ by randomly choosing from the available classes (three in this case: door, picture frame, window). Each class has the same probability.
3. Find the position of the 3D corner, and the room wall s_i it is anchored to as explained in Fig. 2.13.
4. Sample $\mathcal{N}(r_{i3}; \mu_{\tau_3}, \sigma_{\tau 3})$ to propose the ratio r_{i3} between the room height and that of the proposed frame height h_i. Set $h_i = \frac{h_r}{r_{i3}}$, where h_r is the current height of the room box.
5. Conditioned on s_i, the frame has either negligible width or negligible length. In the latter case, sample from $\mathcal{N}(r_{i1}; \mu_{\tau_1}, \sigma_{\tau 1})$ to propose the ratio r_{i1} between h_i and width w_i. Set $w_i = \frac{h_i}{r_{i1}}$ using h_i from the previous step, and $l_i = \epsilon$. When w_i is negligible, set $l_i = \frac{h_i}{r_{i1}}$, and $w_i = \epsilon$.
6. Shrink any frame on the same wall that collides with the proposed one

7. If the frame does not fit on the wall, expand it. In case the room box changes here, correct for the position of the objects using Eq. 2.15

8. Sample over the object continuous parameters using Diffusion move 1 (delayed acceptance step)

9. Accept or reject the proposed frame by consulting the MH acceptance formula

Notice that this new version of the move exploits both bottom-down cues, since the detected image corner specifies the 3D position of the object, and top-down cues, since the prior is used to determine the object's size. This allows to combine the relative merits of these two complimentary approaches (Han and Zhu, 2005; Zhu et al., 2006).

We also experimented using the mean of the prior to propose the object's width, height and length instead of sampling from it. We report that this alternative method produces the same results, since the delayed acceptance step "fixes" the object's size to match the image evidence. However, whether we use the mean or a sample, prior typically allow to propose a more realistic object with respect to using a random size. This results in a better acceptance rate.

Initializing the object proposals

We introduce an initialization procedure that allows to generate object proposals that are more likely more often. This is executed after initializing the room box and the camera parameters.

Specifically, we iterate over the detected corners, and, for each object category, we generate an object proposal using Algorithm 3.1 (Algorithm 3.2 for frames). This moves also requires the current estimate of room box and camera parameters, and in this case we used the room box and camera found at initialization. For each proposal, we keep track of the posterior, and associate it with the corner it was used to generate it. Then, we sort the corners based on this posterior.

When we add an object to the scene, we now pick a corner with a probability that is proportional to the posterior computed in this initialization step associated to that corner, as opposed to assigning to all corners equal probability of being chosen. This ensures that corners that generated the most likely proposals in this initial stage will be used more often during the entire inference process. This makes the process more efficient, and at the same time it does not commit to these initial object proposals, which only encourage the sampler to use the most promising corners more often Notice that we cannot iteratively add the most likely object to the scene, since early commitment to partial configurations would lead to error (Lee et al., 2010).

Multithreaded inference

We further extend the inference by using multiple threads, which allows to explore more of the output space. Each thread executes the entire inference procedure after being initialized with a different room box and camera. Notice that the algorithm described here is the multi-threaded extension of Algorithm 2.4, with a few additional changes to handle the other modifications introduced in this chapter.

In Algorithm 2.4, we propose a room box and a camera from each detected corner, and use the sample with the highest posterior to initialize the single-threaded inference. Here, we keep track of the N best samples, and use them to initialize the N threads. When each thread has terminated, we keep the sample with the highest posterior across all threads. As an additional difference with respect to Algorithm 2.4, we greedily assign each object in the final sample to the category maximizing the posterior (Step 7 in Algorithm 3.3). We also introduce a method to exchange information among threads. In most cases, we found that some objects are found only by some of the threads, as illustrated in Fig. 3.5. Hence, we allow threads to exchange information, as explained in the next section.

Below, we provide a summary of the multithreaded inference algorithm, where N is the number of threads. In our experiments, we used $N = 22$.

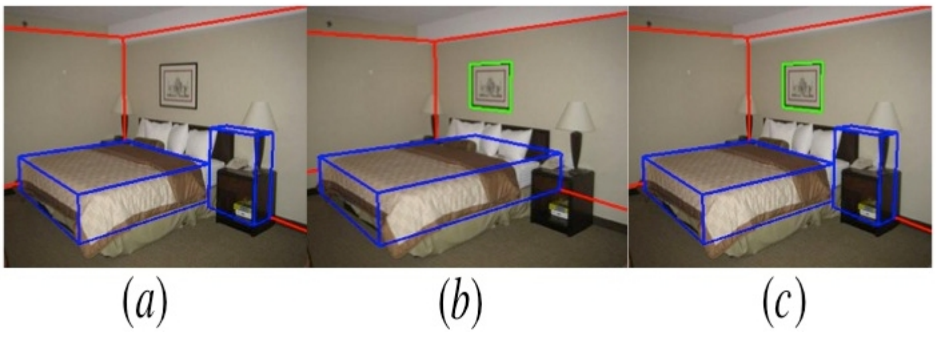

Figure 3.5: Exchanging objects among threads. During inference, some objects are found only by some of the threads. Here, the bed was found only by one thread (a), while the picture frame was found by a different thread only (b). We let threads exchange objects at the end of the inference, producing the result in (c).

Algorithm 3.3: Multi-threaded inference

```
1. Detect straight line segments from the image plane
2. Estimate a triplet of vanishing points, use them to initialize the focal length $f$ of the camera
3. Use the vanishing points and the line segments to find image corners
4. Use each available corner to propose a room box and a camera. Keep the $N$ samples with the highest posterior. Use them to initialize $N$ threads. Denote the initial sample for thread $i$ as $\theta_0^i$.
5. For each thread $i$, sort the corners as explained above. Here, use the room box and camera in $\theta_0^i$ to generate proposals.
6. For each thread $i$, repeat $k = (1, .., K)$ times
    (a) Generate a new sample $\theta_k^i$ conditioned on the previous sample $\theta_{k-1}^i$ by using one of the following moves, chosen with equal probability
        • Jump 1: add an object to the scene (furniture of frame). This move now uses the prior to propose the size of the object
```

- Jump 2: remove an object from the scene
- Jump 3: change the room box
- Jump 4: randomly select an object, and propose changing its type
- Diffusion 1: pick a random object and sample over its parameters
- Diffusion 2: sample over the room box parameters only
- Diffusion 3: sample over room box and camera parameters

(b) Reject samples that violate the constraints on the camera position (camera outside the room or inside an object).

7. For each thread i, keep track of the sample θ^i_{best} with the highest posterior. For each object in θ^i_{best}, try changing its type to all the available categories, and keep the one maximizing the posterior
8. Exchange objects among the samples $(\theta^1_{best}, ..., \theta^N_{best})$ as explained below.
9. Return the sample with the highest posterior across all threads.

Exchanging objects among threads

At the end of the inference process, each thread n outputs the sample θ^N_{best} with the highest posterior. In most cases, we found that some objects are found only by some of the threads, as illustrated in Fig. 3.5. One thread did not find the picture (a), while the other did not find the nightstand (b). While a longer running time could potentially allow each thread to find all objects, we propose instead to let threads exchange objects at the end of the inference (c).

Since object position and size are defined relatively to the room, we need an exchanging mechanism taking into account that different threads have different room boxes. Further, the camera parameters found by each thread are potentially different. Hence, we exchange an object between a source thread and a destination thread by enforcing that the projection of the object in the source matches as closely as possible the projection of the object in the destination. This requires different procedures for furniture objects and frames, which are illustrated in Fig. 3.6 and Fig. 3.7.

At the end of inference, we add to the best sample found by a thread all the objects found by the other threads, one at a time, and keep the object that provides the larges increase in the posterior. We repeat this $K = 10$ times, or until there is no improvement in the posterior. While less greedy methods are possible, this approach works well in practice. The complete procedure for exchanging objects among N threads is as follows.

Algorithm 3.4: Exchanging objects among threads

```
1. For each thread $i$, save in $\theta_i^0$ the best sample found by thread $i$
2. For each thread $i$, construct the set of object found by all the other threads
   $O_i^* = \cup_{j=1}^N O_j$ with $j \neq i$, where $O_j = (o_{j1}, ..., o_{jn})$ is the list of objects in $\theta_j^0$.
3. For each $\theta_i^0$, and for $k = 1, ..., K$
    (a) Set $\theta_i^k = \theta_i^{k-1}$. Compute the posterior $p_i^k$ of $\theta_i^k$.
    (b) for each object $o_w$ in $O_i$, create $\theta_i^{k\_w}$ by adding $o_w$ to $\theta_i^k$, and compute
        the posterior $p_i^{k\_w}$. Set $\theta_{i\_max}^k = \theta_i^{k\_w}$ such that $p_i^{k\_w} = \max_{w'} p_i^{k\_w'}$.
    (c) if the posterior of $\theta_{i\_max}^k$ is larger that $p_i^k$, set $\theta_i^k = \theta_{i\_max}^k$. Otherwise,
        stop and return $\theta_i^K = \theta_i^k$.
4. Return $\theta_i^K$ with the largest posterior.
```

3.4 Results

Also in this case, we evaluate our algorithm on the room box error, by testing on the same datasets used in Chapter 2. We start by evaluating the benefits of using

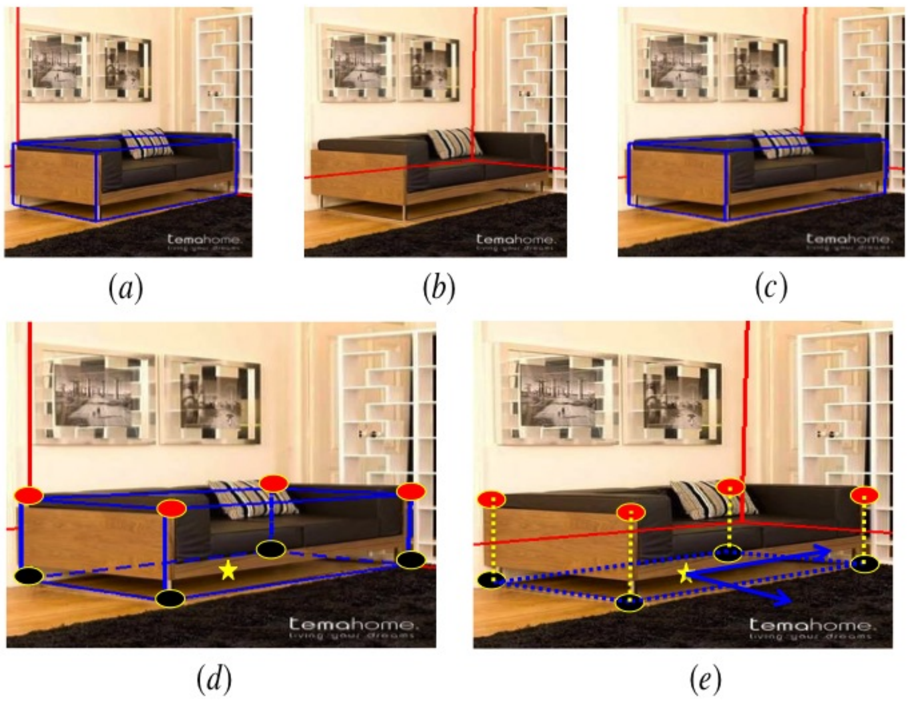

Figure 3.6: Exchanging furniture objects between two samples. Adding the object in (**a**) to (**b**) is difficult, because the camera and the room boxes are different. We address this by ensuring that the projection of the transferred object (**c**) matches that of the original one (**a**). We use the original camera to project the object corners on the floor, shown in black in (**d**). The star denotes the center of the projected "floor" of the object. We cast a ray between the destination camera and the 2D position of the star, and intersect it with the floor of the destination room, which gives the object "floor" center in the destination room box. We similarly "transfer" all the object corners. We enforce the transferred object to be aligned with the destination room box (the arrows in (**e**)). We find the object width and length by intersecting the arrows with the dashed blue lines (**d**), which might not be aligned with the destination walls due to the different cameras. The object height is found from the "transferred" corners from the object top (**d**). For each 2D corner c, we consider ray r_1 through c and the destination camera, and ray r_2 orthogonal to the room floor and through the corresponding 3D corner of the object on the floor. The 3D corner that projects to c is the point on r_2 closest to r_1. We find the height as the average of the 3D length of the four dashed yellow lines. The resulting transferred object is shown in (**c**).

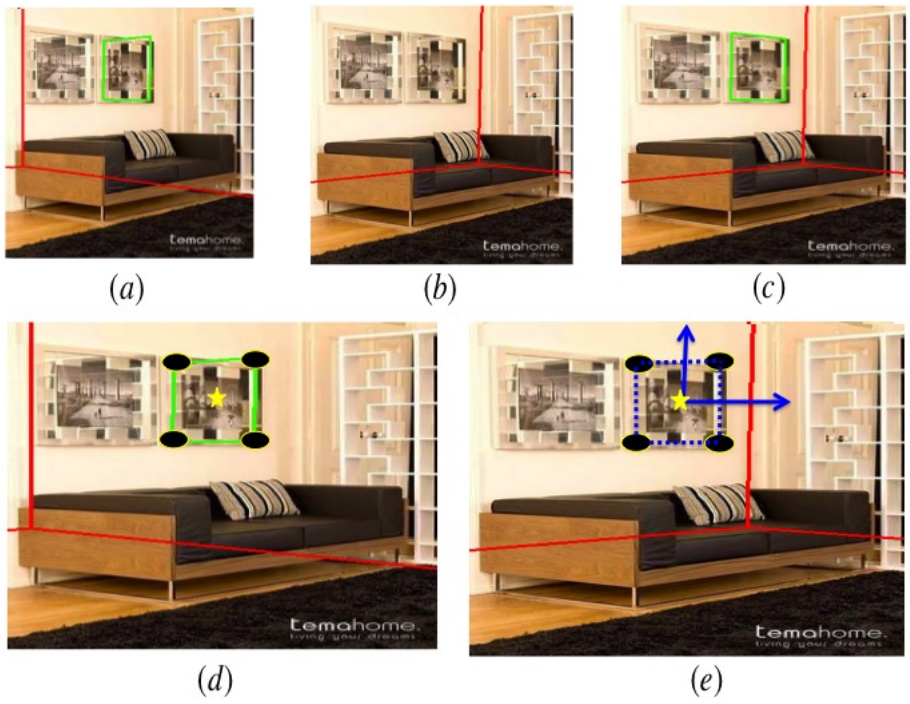

Figure 3.7: Exchanging frames between two samples. Here, we want to add the frame in (**a**) to (**b**), where the camera and the room box are different. As for furniture objects (Fig. 3.6), we move the frame by ensuring that its projection in the destination room (**c**) matches its projection in the original one (**a**). As in Fig. 3.6, we "transfer" the projection of the object center (the star) and of its corners (in black) from the original to the destination room via the image plane. Instead of considering the intersections on the floor, we consider the wall in this case. We find the frame size by intersecting the blue arrows in (**e**), which are aligned with the destination room axes, with the dashed blue lines. The resulting transferred frame is shown in (**c**).

priors by comparing against the algorithm in Chapter 2. For proper comparison, we use the same likelihood from the previous chapter, which only includes edges. Results on the Hedau dataset are shown in Table 3.1 (left), where we see that the room prior reduces the room box error, and so do the object priors. We also see that this trend is confirmed when we add orientation maps to the likelihood. This is an interesting results, since using more realistic objects improves the fit of the overall scene, measured in terms of the accuracy of the room box fit.

We then measure the effect of the components of the likelihood in Table 3.2. Here, we consider the full, multi-threaded algorithm discussed in this chapter, and consider three scenarios: 1) we fit only the room box and the camera without any objects; 2) we also allow objects, with a maximum of 10 per scene, but without allowing threads to exchange information; and 3) we also let threads to exchange objects. For each of these three scenarios, we use different likelihoods. First, we use only edges, then add orientation maps, and last the full likelihood including also geometric context. We see that adding orientation maps and geometric context reduces the error in all three scenarios. We also see that exchanging objects among threads also improves the score for all the different likelihoods we tested with. Interestingly, exchanging objects allows choosing a better room box too, since more objects provide more evidence on the correct room box.

In Table 3.3 we further evaluate on the room box error on both the Hedau and UCB dataset. In the first two lines, we see the improvements with respect to our approach without priors from Chapter 2. We also compare against other relevant work in the field, and also include the results of our most recent work (Del Pero et al., 2013), which is the state-of-the-art method in terms of room box error (Chapter 4).

Evaluation on object recognition

The method discussed in this chapter can be also evaluated in terms of object recognition. In fact, thanks to the object type labels, our model can now be used to predict the identity of objects in the scene.

We ground truthed the UCB and the Hedau dataset by manually identifying the

Table 3.1: Evaluating the benefits of the priors on the room box error. The results here are on the Hedau test set. Priors improve results when we use edges, and also when we add Orientation Maps(OM)

	Edges	Edges+OM
No Prior	24.2%	20.4%
Room prior	23.2%	19.7%
Room + obj prior	**20.7%**	**17.8%**

Table 3.2: Analysis of the components of our approach based on room layout error, evaluated on the Hedau test set.

	Edges	Edges + OM	Edges + OM + GC
No objects	24.1%	21.3%	21.8%
Objects	20.7%	17.8%	17.2%
Exchange objects	**18.2%**	**14.6%**	**13.6%**

seven object classes we experimented with, not considering objects occupying less than 1% of the image. To evaluate detection, we project the 3D object hypothesized by our model onto the image, and compare this with the ground truth object position. If the intersection of the two areas is more than 50% of their union, we consider it a correct detection, which is the criterion used in the PASCAL challenge (Everingham et al., 2010).

First, we measure how many objects we correctly identified for each of the two main categories (furniture and frames), even if there is confusion within the subcategories (e.g. when we label a table as a couch, or a window as a door). We provide precision and recall scores based on this criterion, which is a standard way of evaluating object recognition, used for example by Felzenszwalb et al. (2009). Second, we measure the accuracy we achieved within each of the two categories, as the percentage of objects that were assigned to the correct subcategory.

We report in Table 3.4 that using object priors greatly improves precision and recall, for both furniture and frames. When we do not use priors, objects are not labeled (e.g., we do not know whether a furniture object is a couch or a bed), and

Table 3.3: Performance analysis based on room box layout error.

	Actual	
Method	Hedau dataset	UCB dataset
Full approach no priors	24.2%	23.3%
Full aproach with priors	**13.6%**	**14.2%**
Hedau et al. (2009)	21.2%	NA
Wang et al. (2010)	20.1%	NA
Lee et al. (2010)	16.2%	NA
Del Pero et al. (2013)	**12.7%**	**14.0%**

subcategory accuracy cannot be evaluated. Table 3.5 shows that geometric context improves all recognition scores, except for frame subcategory classification on the Hedau dataset. This is expected, as geometric context was trained only on furniture objects and not on frames.

In general, performances are more modest on the UCB dataset, and this includes also the room layout error. This black and white dataset is in fact more challenging, as several images are extremely blurry. We also notice that geometric context only introduces small improvements in the subcategory classification, if we exclude furniture on the UCB dataset. This is because adding this new feature allows to obtain more accurate object fits, but the "burden" of classification is still entirely on the prior on size and position.

The effects of exchanging objects among threads are available in Table 3.6. There is a trend showing that exchanging objects achieves better recall at similar or slightly lower levels of precision. This is because adding objects from other threads allows to find many more objects, at the cost of a few additional mistakes. Variations in subcategory classification are mostly negligible. Last, we report confusion matrices for both furniture and frames on the Hedau dataset 3.7. We here consider our full approach, including edges, orientation maps, geometric context, priors and object exchanging. We see that we recognize more objects in categories that are better approximated by a single block, such as beds and cabinets. Performances decrease for concave objects like table, that are not well approximated by a convex box. Also,

Table 3.4: Benefits of object priors evaluated on UCB (left) and Hedau (right) datasets. P, R, and S denote Precision, Recall and Subcategory Accuracy

	P	R	S	P	R	S
Furn no prior	19.4%	10.4%	NA	27.1%	9.2%	NA
Furn prior	**31.0%**	**20.1%**	**38.0%**	**32.5%**	**20.3%**	**50.0%**
Frames no prior	21.8%	14.0%	NA	23.1%	19.5%	61.2%
Frames prior	**27.2%**	**19.7%**	**60.0%**	**36.1%**	**27.5%**	**62.6%**

Table 3.5: Benefits of geometric context (GC) evaluated on the UCB (left) and Hedau (right) datasets

	P	R	S	P	R	S
Furn no GC	31.0%	20.1%	38.0%	32.5%	20.3%	50.0%
Furn GC	**33.7%**	**27.7%**	**50.1%**	**50.0%**	**24.2%**	**51.6%**
Frames no GC	27.2%	19.7%	60.0%	36.1%	27.5%	**62.6%**
Frames GC	**28.2%**	**24.3%**	**60.8%**	**38.4%**	**28.3%**	61.2%

there is more confusion between object categories similar in size, such as couches and tables, or windows and picture frames.

Qualitative results are available in Fig. 3.8, where we show a few examples of the scene reconstructions provided by the full algorithm. Some failures are shown in Fig. 3.9. Here, it is important to notice that confusion among objects of different types (e.g. a bed confused for a couch) mostly happen when objects are similar in size. This is because all the objects share the same geometric representation (a single block) and only position and size are used for classification. Further, while

Table 3.6: Effects of exchanging objects among threads on the UCB (left) and Hedau (right) datasets

	P	R	S	P	R	S
Furn no swap	**33.7%**	27.7%	**50.1%**	50.0%	24.2%	**51.6%**
Furn swap	33.0%	**29.7%**	50.0%	**53.5%**	**28.5%**	51.3%
Frames no swap	28.2%	24.3%	**60.8%**	**38.4%**	28.3%	61.2%
Frames swap	**28.4%**	**37.3%**	59.8%	37.5%	**35.5%**	**66.0%**

Table 3.7: Confusion matrices on Hedau test set

	Bed	Cabinet	Couch	Table		Door	Picture	Window
Bed	14	1	9	1	Door	7	0	7
Cabinet	1	12	2	2	Picture	0	29	13
Couch	6	1	7	5	Window	8	5	28
Table	4	3	4	6				

the priors and the new likelihood components improve performances on recognition and room box error, many objects are still missed, and we still report failures on room box and camera estimation.

3.5 Discussion

Adding priors to constrain the size and position of objects improved the results both in terms of recognition (Table 3.4), and also with respect to scene reconstruction, as measured in terms of room box error (Table 3.1). This second result is of particular interest, as it shows that improving the fit of an element in the scene, (the objects) improves the fit of the rest of the scene (the room box). This is a promising finding towards our goal of a global scene interpretation. This is in fact an example where exploiting the interplay among parts of the scene allowed finding a better overall estimate of the scene.

The recognition results are promising, and also show the benefits of using a 3D representation. In fact, the priors used in this chapter are defined in 3D, where we do not need to worry about variations due to the camera, and where we can use prior knowledge on the 3D scene that is independent of the image data, such as the text available from furniture catalogs that we used to set the priors.

However, the experiments also show that using only size and position for recognition is not particularly robust, especially when object classes are similar in size, such as tables and couches. We posit that having different geometric models for object classes, such as tables with legs or couches with backrests would improve this score. We discuss this idea in Chapter 4.

Further, while the new likelihood introduced here is more robust than the one

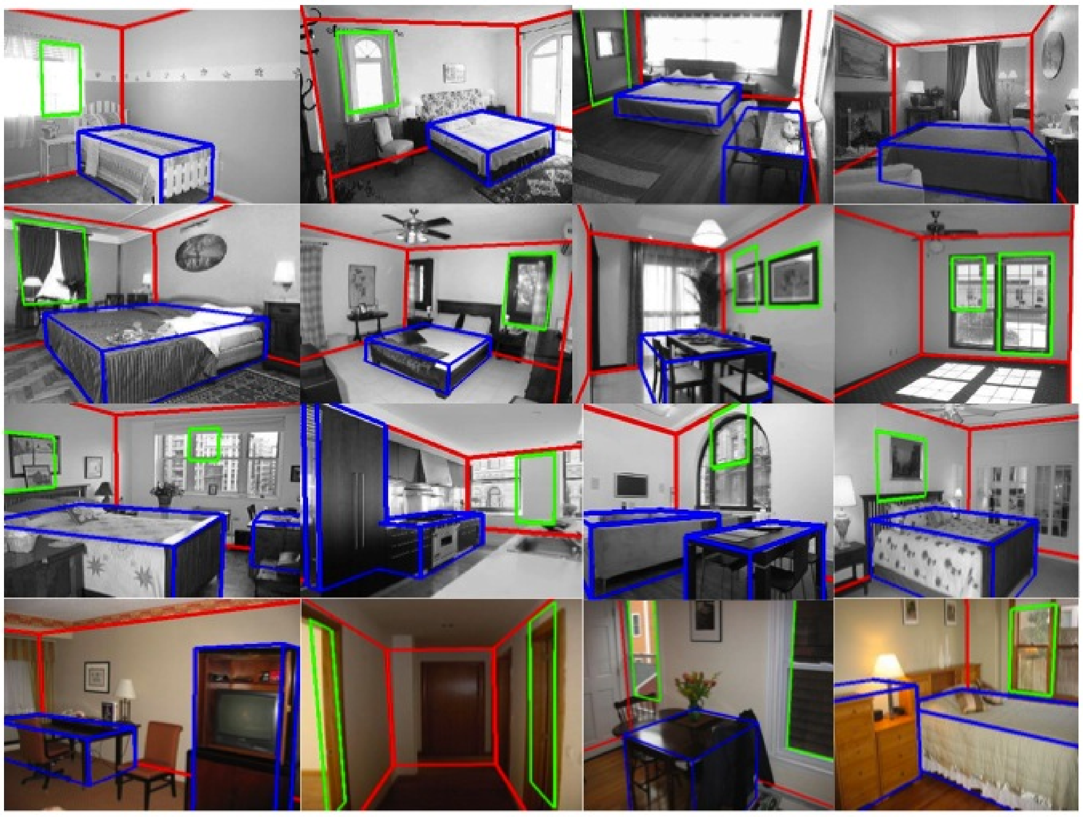

Figure 3.8: Promising scene reconstructions provided by our full approach. We show the estimated room box in red, furniture objects in blue and frames in green

based only on edges, we still report several mistakes, as we both miss objects and have false positives (Fig. 3.9). Better appearance models could potentially improve results. For example, our likelihood function is global, and it could benefit from appearance models trained for each specific object category.

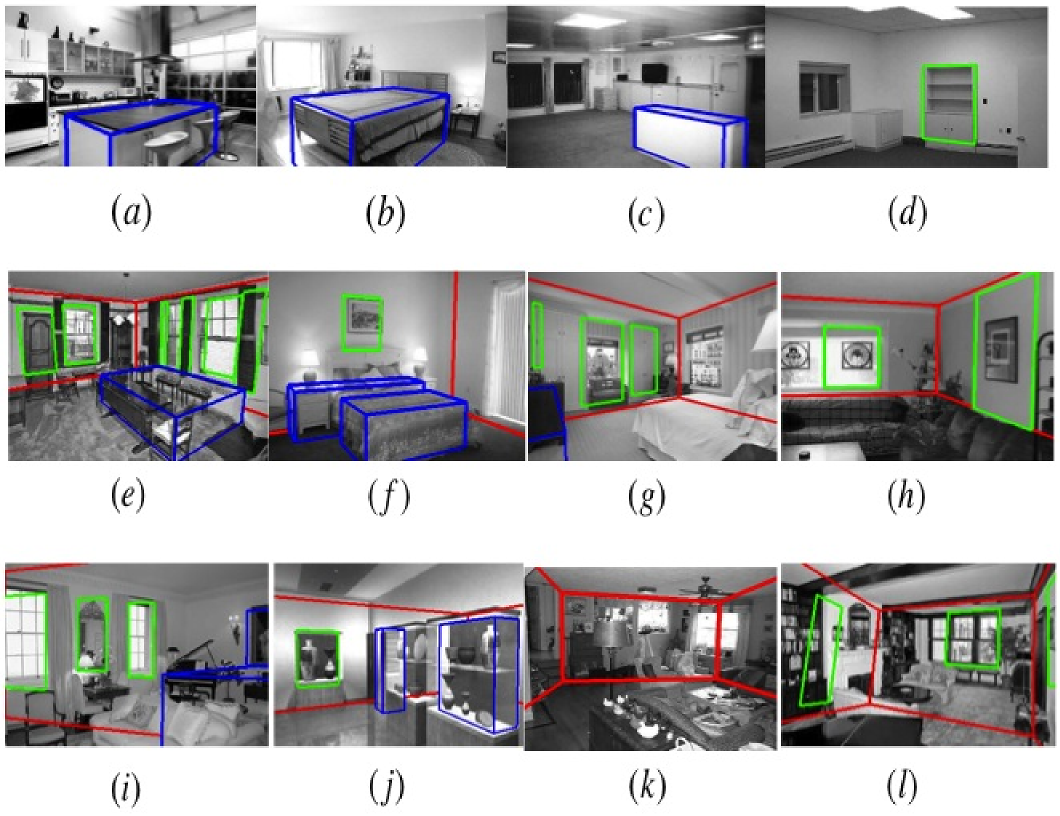

Figure 3.9: Some typical failures. Top row: We often confuse object categories similar in size and position. The table in (**a**) is wrongly labeled as a bed, the opposite happens in (**b**). The wall in (**c**) is labeled as a cabinet, and we sometimes confuse an object that is directly facing the camera for a frame (**d**). Due to clutter we sometimes hallucinate objects (**e-f**). However, in these two cases our approach still provides a reasonable approximation of the space occupancy of the scene. In (**g-h**) the method completely missed the objects in the image. In (**i-j**), the algorithm estimated the wrong room box. This happens mostly when the edge detector misses the edges between the room walls and the ceiling completely. In (**k-l**), two catastrophic failures are due to bad initial estimates of the camera, from which the algorithm could not recover.

CHAPTER 4

Using composite 3D models for indoor objects

4.1 Introduction

The method discussed so far relies on a simple geometric model, where a single box is used to approximate the room box, and also to represent any object inside it, such as beds and tables. This representation is very common in most recent work on indoor scene understanding (Lee et al., 2010; Hedau et al., 2010, 2012; Gupta et al., 2011), as it allows promising reconstruction results, and holds the advantage that gross geometric structure simplifies inference.

However, these representations have four main limitations which we illustrate using Fig. 4.1. First, bounding boxes of concave objects projected into images tend to include much background, which is confusing evidence for inference. For example, the middle-top image shows the output of the geometric context classifier (Hoiem et al., 2005a), where pixels with higher probability of being part of an object instead of the wall or the floor are colored gray. Fitting a single 3D block to this feature map will be hampered by the confusing evidence, whereas a more articulated table model with legs and top explains the classification results for pixels between the legs of the table.

Second, even if a single-bounding-box representation succeeded in discovering the presence of an object in the image, the parameters of a single fitted block have only modest power to distinguish objects. Results in Chapter 3 showed that it is possible to classify furniture objects based only on 3D bounding box dimensions, but with much confusion when objects are similar in size. Having a class-dependent topological structure should help resolve such ambiguities, and in this case a composite table model is clearly a better fit than a simple box (Fig. 4.1, bottom left).

Third, plain blocks cannot capture complex spatial configurations, for example

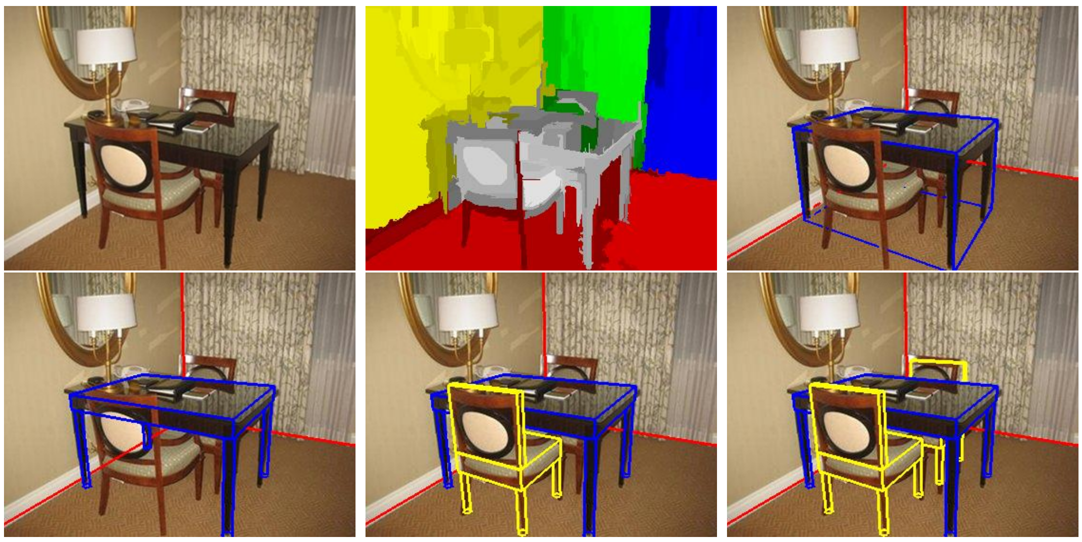

Figure 4.1: Benefits of using accurate 3D models of objects. 1) Detailed geometric models, such as tables with legs and top (bottom left), provide better reconstructions than plain boxes (top right), when supported by image features such as geometric context (Hoiem et al., 2005a) (top middle), or an approach to using color introduced here. 2) Non convex models allow for complex configurations, such as a chair under a table (bottom middle). 3) Detailed geometry allows for distinguishing categories that are similar in size.

they would not allow sliding chairs under the table (Fig 4.1, bottom row). Finally, a finer representation is also more useful for robotics applications. For example, a small robot would be able to infer that there is a possible path between the table legs.

These observations led us to extend our framework for indoor scene understanding with representations for articulated objects[1], such as the table and the chairs in Figure 4.1. Specifically, we use a representation based on re-usable geometric parts, from which we build more complex models (Sect. 4.2). For example, we build a chair from a set of legs, a seat and a backrest (Fig. 4.2, top left). Parts are re-used in different object models, for example the set of legs used to model chairs is also used to model tables. This establishes an interesting link between geometric parts and function, for example most objects used for "sitting" share a seat and backrest.

Additionally, we propose a new feature based on color, which encourages the projections of the predicted objects to be uniform in color (Sect. 4.3). We demonstrate that this feature both supports, and benefits from, more accurate geometric models. For example, one should see from Fig. 4.1 that a table with legs would provide a better grouping than a single block, just like for the case of geometric context. The single block contains in fact many background pixels, whose color does not agree with that of the table.

We also introduce new and more sophisticated data-driven inference mechanisms to support the inference of more complex geometric structure (Sect. 4.4). We designed specific inference for each of the geometric parts, which are naturally shared by all the objects containing that part. For examples, we implement a strategy to propose legs from detected pairs of vertical segments, which is shared by all furniture categories supported by legs, such as tables and chairs. Interestingly, this framework allows a modeler to create models using the available parts without having to worry about the inference.

Further, we add more semantic structure to our model in the form of contextual relationship among objects (Sect. 4.4.3). So far, objects interacted with each other

[1]This is the main contribution of our 2013 CVPR paper (Del Pero et al., 2013)

by avoiding intersections, and with the room box, which conditions their position. For example, objects labeled as "beds" are more likely to be against a wall than in the middle of the room.

Here, we want to consider more meaningful relationships to solve ambiguities in the imaging process, for example to address significant occlusion or weak image evidence. We demonstrate this by showing how to improve recognition of chairs by looking around tables. There is often little evidence supporting the identification of a chair, perhaps just a leg or the top of a backrest (Fig 4.1, bottom right), but this can be addressed in a top-down fashion, by looking for chairs in places that are likely based on the current model hypothesis (e.g. around tables).

Last, we evaluate in Sect. 4.5 the impact of the proposed innovations (more accurate geometry, color feature, and contextual relationships) on the room box error and on object recognition. We test on the same benchmarks used in Chapter 2 and 3.

4.1.1 Related work

A few recent methods proposed to go beyond the standard single-block representation used to model indoor scenes. Hedau et al. (2012) used a plane to model backrests on top of generic blocks. However, if we exclude the backrest, their model still relies on the block representation, and there is no attempt to distinguish among object classes according to their geometry. Satkin et al. (2012) then proposed to match detailed 3D models of bedrooms and living rooms available from Google Warehouse, but they do not allow any variability in the size and the arrangement of the objects in the model.

Our part-based representation relates to several methods for object recognition both in 2D and 3D, discussed in detail in Sec. 1.1.2. The proposed geometric model is closely related to the work of Schlecht and Barnard (2009b), who used geometric parts to model the 3D structure of furniture. However, their goal is quite different, as they set out to learn the topological structure of an object category, say tables, and its variations, from a set of training images. In our case, the topology of a class is given, as the modeler specifies that tables consist of a set of a legs and a top, and also provides the statistics on their variations in the form of prior distributions. As an interesting consideration, our approach can potentially use the object topologies learned using the method by Schlecht and Barnard (2009b), as opposed to the ones we manually provide. This would be an additional step towards a fully automated system.

For a detailed discussion on the advantages of using parts, we also refer to Sec. 1.1.2. One of them is that we can design inference strategies for a specific part, which are naturally shared among all object models containing that part. Another one is scalability, as several models can be built from a modest set of parts.

The use of color to encourage uniform groupings of image pixels has been used in other areas of vision, such as image segmentation, for example in the work of Shi and Malik (2000, 1997). However, in our case the groupings are provided top-down by the 3D model hypothesis, and we do not need to consider all the arbitrary groupings that a bottom-up segmentation algorithm would need to entertain without any such guidance.

Last, during inference we use pairs of detected vertical segments to generate data-driven proposals of objects with legs, such as tables and chairs (Sec. 4.4.2). Also Hedau et al. (2012) use a similar feature, which they call "peg". However, in their work pegs are part of the likelihood, as a way to explain the missing edges between the legs of a table, which is still modeled with a single block. Here, we use pegs only to generate proposals during inference, as our likelihood does not need to explain that missing edge, since not finding it is predicted by our strong geometric model with legs.

4.2 Modeling indoor objects with reusable parts

The main contribution of this work is representing object models as assemblages of re-usable geometric primitives (parts), as opposed to simple bounding boxes. For example, Figure 4.2 shows a chair built by stacking a set of four symmetric legs, and

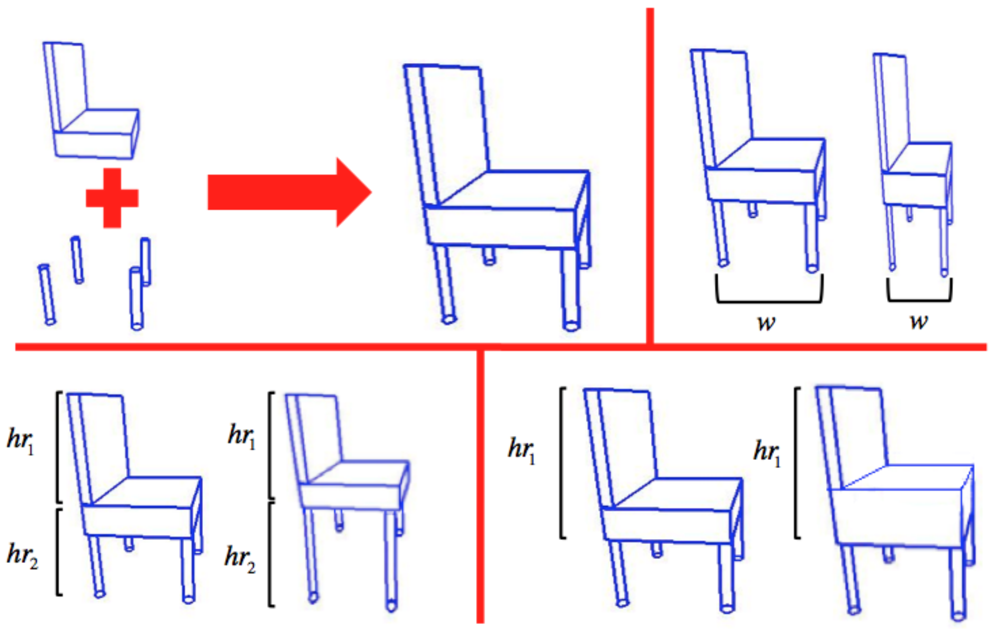

Figure 4.2: Object models as assemblages of parts. Top left: A chair is built by stacking two parts, a group of four legs and an L-shaped component. Top right: Changes in the object bounding box propagate to each part. Here the object width is divided by two, and this results in parts that are half as wide. Bottom left: Parts are stacked vertically, with their height defined as a ratio of the total object height (two different ratios shown). Bottom right: Changing the internal parameters of a part, here the L shaped one, while keeping the part bounding box fixed.

an L-shaped structure. Here, we provide a generic formulation that is independent of the parts used, and in Sect. 4.2.2 we describe the parts used in our experiments. We use the same Bayesian model introduced in Chapter 2 and 3, and we augment it with this new part-based representation of objects.

As previously discussed, the way we model the 3D geometry of the objects has to take into account that reconstruction from single images introduces an overall scale ambiguity. For this reason, the size and position of objects and their parts are defined relatively to the overall room box size. In fact, we cannot estimate that a table is 3-feet high, but we can provide a coherent scene reconstruction where its height is realistic relative to the overall height of the room. Similarly, we cannot estimate that the chair backrest is 2 feet, but we can estimate that it is roughly the same height as the chair legs.

To parameterize objects and their parts relatively to the size of the scene, we retain the concept of an object's bounding box, whose size is defined relatively to that of the room box. The geometry of the parts is in turn defined relatively to this bounding box. We thus extend the object parameters defined in Eq. 3.1 as

$$o = (s_i, x_i, y_i, z_i, w_i, h_i, l_i, t_i, \sigma_{ti}) \quad , \tag{4.1}$$

where σ_{ti} specifies the topological structure of the object, i.e. the geometric parts in the object model, and their arrangement. Notice that σ_{ti} is conditioned on the object category t_i, and is parameterized relatively to the object bounding box (x, y, z, w, h, l).

For each object category, the modeler has to specify the topological structure, by choosing from a palette of available parts. The arrangement of the chosen parts within the object bounding box also needs to be specified. For now, we constrain the modeler to stack the parts vertically, although extensions are possible. Notationally,

$$\sigma_{ti} = (p_1, ..., p_n, hr_1, ..., hr_n) \quad . \tag{4.2}$$

Here $(p_1, ..., p_n)$ is the collection of parts, which is fixed for each object class. Variable hr_i denotes the height of the ith part expressed as a ratio of the total object height h_i, with $\sum_{i=1}^{n} hr_i = 1$.

The part heights together with the object bounding box determine a bounding box for each part in the model. Consider part p_j of object o_i: given the part heights $(hr_1, ..., hr_n)$, the object center (x_i, y_i, z_i) and its size (w_i, h_i, l_i), the bounding box for p_j will be centered at

$$(x_i, y_i + h_i(-\frac{1}{2} + \frac{hr_j}{2} + \sum_{k=1}^{k<j} hr_k), z_i) \quad , \tag{4.3}$$

with size

$$(w_i, h_i(hr_j), l_i) \quad . \tag{4.4}$$

All these coordinates are defined with respect to the room reference frame. Notice that, since parts are stacked vertically, the bounding box for a part has the same width and length as the entire object, while only the height changes. We now provide a full example to better illustrate this parametrization.

Example 4.1: Relative parametrization of the parts of a chair

Consider the chair in Fig. 4.2, bottom left, which consists of two parts (p_1, p_2), an L-structure and the legs. The chair center in the room is denoted by (x_i, y_i, z_i), and its size by (w_i, h_i, l_i). Imagine that for this specific instance of chair, the legs account for 45% of its entire height h_i, implying implying $hr_1 = 0.45$ and $hr_2 = 0.55$. This means that the bounding box for the legs is centered in $(x_i, y_i + h_i(-\frac{1}{2} + 0.225), z_i)$, with size $(w_i, 0.45h_i, l_i)$. The bounding box for p_2 is instead centered at $(x_i, y_i + h_i(-\frac{1}{2} + 0.275 + 0.45), z_i)$, with size $(w_i, 0.55h_i, l_i)$.

Parts are ordered vertically from bottom to top, and we will refer to the bottom one as the *support*. Each part p_i comprises a set of internal continuous parameters $(p_{\theta 1}, ..., p_{\theta n})$ and a discrete variable c_i which we call configuration

$$p_i = (p_{\theta 1}, ..., p_{\theta n}, c_i) \quad . \tag{4.5}$$

$(p_{\theta 1}, ..., p_{\theta n})$ are defined relatively to the bounding box occupied by that part. The ratio between the height of a chair's seat and that of the entire part (the L-component) is an example of an internal part parameter, and Fig. 4.2 (bottom right) shows changing it while keeping the part bounding box fixed.

c_i specifies the configuration of the part, and we use this to model discrete internal variations of the part. For now, we use this variable in the case of non symmetric parts, as illustrated for the L-component below (Example 4.2). However, c_i could be used also for additional discrete variations, for example the part used to model a table's top could either be a block or a cylinder. We denote the number of possible configurations for a part p_i as N_{pi}. Notice that $N_{pi} = 4$ in the case of the L-component, while we set $N_{pi} = 1$ for parts that are completely symmetric. We now provide a full example clarifying the way the internal parameters work.

Example 4.2: Internal parameters of the L-component

We illustrate how internal parameters work using the L-component (Fig. 4.3, top left) as an example. This part, which we call p_L, is defined by parameters (c_L, p_{L1}, p_{L2}). Discrete variable c_L denotes the attachment of the vertical block, which can take four values (back side of the base, left side, right, and front, shown from left to right in Fig. 4.3, top left). Notice how this is a discrete variation introduced by the non symmetric structure of the part. This modeling choice was made considering that objects, and thus their parts, are aligned with the room surfaces, implying that for this non-symmetric component only four discrete configurations are possible. This is equivalent to modeling the rotation of the L-component around its y-axis with a discrete angle variable that can take only four values (0°, 90°, 180°, 270°). Continuous parameters (p_{L1}, p_{L2}) are, respectively, the ratio between the height of the seat and that of the entire part, and the ratio between the width of the seat and the width of the part (or length, depending on c_L). These two parameters are denoted by the red arrows in Fig. 4.3, bottom left. Given (c_L, p_{L1}, p_{L2}) and the part bounding box, one can univocally derive the position and size of the two 3D blocks forming the part (e.g. the base and the vertical block). Notice that this parametrization is very compact and removes dependencies among parameters as much as possible. Modeling the L-component as the position and size of the base and of the vertical block would instead require several parameters (12 in this case), which would also be correlated. For example, given the height of the part bounding box and that of the base, the height of the vertical block is not free.

To summarize, an object is a vertical stack of parts. The object size and position in the room are specified by its bounding box, while part heights $(hr_1, ..., hr_n)$ determine bounding boxes for each part. Last, internal part parameters are defined relatively to these boxes. An advantage of this representation is that object parameters are subdivided into three sets, and this is is very convenient for inference (Sect. 4.4). Changes in the bounding box parameters propagate to all the parts

(Figure 4.2, top right), changes in the part heights propagate to the parameters for the affected parts (bottom left), and changing the internal parameters results in changes local to the specific part (bottom right). Further, having parts that capture some of the complexities of the objects, while inheriting their bounding box, simplifies the work of the modeler.

Additionally, we impose the simple containment constraints from Chapter 2, meaning that objects must be entirely inside the room and they cannot overlap. However, precise geometry enables configurations that bounding boxes would not, for example sliding a chair under a table (Fig. 4.1). For efficiency, during inference we first check if the bounding boxes of two objects collide, and only then do we check collisions using the geometry of the individual parts. We also use prior distributions as discussed in Chapter 3, which we still evaluate only on the position and size of the entire object as specified by its bounding box (i.e. not on the individual parts).

4.2.1 Building object models

The proposed framework allows creating a model for an object category by choosing from the palette of geometric parts (we explain in the next section how to add new parts to the palette). To create a new category model (e.g. "chair"), the modeler chooses the parts (legs and L-component), and specifies their arrangement in the vertical stack (the L-component is on top). She also needs to provide the height ratios $(hr_1, ..., hr_n)$ and the internal parameters of each part $(p_{\theta 1}, ..., p_{\theta n})$, which are defined relatively to the part bounding box, encoded at the next level up in the model. Additionally, we leave the option of modeling the object using the single block approximation from the previous chapters.

The modeler can either provide a single value for each of these parameters, which will be kept fixed for each instance of that category during inference, or a valid range. In the latter case, we assume that each value in the range is equiprobable. Rather then include the internal parameters of an object as part of its prior, which leads to additional model selection problems, we simply set them as part of the model. In this work, we set values by manually fitting models to training images. Last,

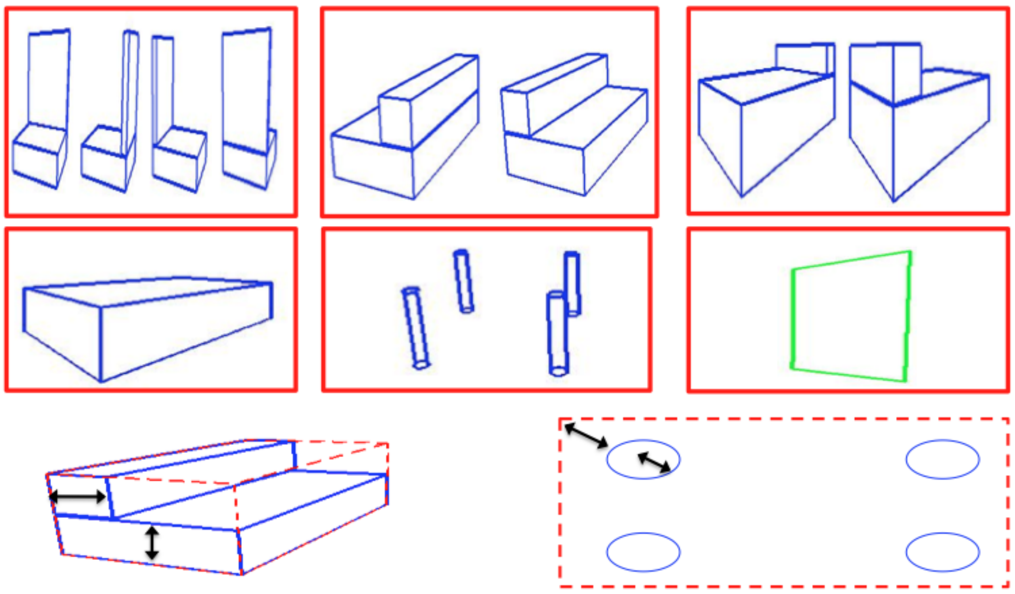

Figure 4.3: The geometric parts used in the expetiments. Top row: we use an L-shaped component to model a backrest on top of a seat and similar structures. From left to right, we use three different versions (L1, L2, L3), distinguished by constraints on where the vertical "back" can be attached. L1 is completely free (four configurations available), and we show all four possibilities. L2 and L3 are for restricting to the long side and to the short side of the horizontal base respectively (two configurations available). Middle: a single block, a set of four symmetric legs, and a frame. Bottom: the free parameters of the L-shaped component and of the set of legs are shown by the double arrows.

parameters for the prior distributions on the object's size and position, as described in Sect. 3.2, must also be provided.

4.2.2 Designing object parts

We designed object parts to be modular so that they can be re-used in the modeling phase. For each part, we chose a parametrization as compact as possible, trying to remove dependencies between parameters, which create problems during inference (see Example 4.2). Remember also that the internal parameters of the part must be relative to its bounding box.

Adding a new part to our framework requires adding rendering methods for that part, which are required to compute the likelihood function (Appendix A). In this work, we rely on an intermediate representation in terms of basic geometric primitives, for which we implemented the needed rendering methods. For each part, we specify how to map its compact parametrization (Eq. 4.5) to a set of such basic primitives that we know how to render.

More specifically, we denote as $G = (g_1, ..., g_{|G|})$ the set of basic rendering primitives. In this work, we use only two primitives, a 3D block and a cylinder. The former is defined by its eight 3D vertices, the latter by its radius, height, center of the base and of the top (see Appendix A). This representation allows using standard OpenGL primitives for rendering.

Given a part p_j of object i, we need a function

$$f_{map}(p_j, o_i) = (g_{j1}, ..., g_{jM}) \tag{4.6}$$

that generates a set of geometric primitives in G given the part internal parameters and the bounding box of the object. For example, in the case of the L-component, we use two 3D blocks as rendering primitives. Given the part bounding box, the base height, and the width of the vertical block, it is straightforward to generate the vertices of these two blocks.

This basic representation is also used to compute collisions, in order to check whether objects overlap. Notice that we can no longer consider only the overlap of the bounding boxes, as this would prevent configurations such as sliding chairs under a table. We implemented efficient collision detection among all the primitives in G (i.e. between two blocks, between a cylinder and a block, and between two cylinders[2]).

In order to detect a collision between two objects o_1 and o_2, we consider the two sets $G_1 = \cup_{p_i \in o_1} f_{map}(p_i, o_1)$ and $G_2 = \cup_{p_j \in o_2} f_{map}(p_j, o_2)$. The two objects overlap if we detect a collision between any primitive in G_1 and any primitive in G_2. For efficiency, this procedure is executed only if we first detect that the bounding boxes of o_1 and o_2 overlap, which requires much less computation.

To summarize, adding a new part requires providing a compact parametrization relative to its bounding box, and a function mapping this parametrization to the representation used for rendering. Optionally, we allow dedicated inference for each part, which takes advantage of part-specific characters to deal with the challenges of fitting complex geometry. These inference strategies defined for a part are naturally shared by all objects using that part, and are transparent to a modeler assembling new objects. We now list the parts used in this work, and we will discuss the corresponding dedicated inference in Sec. 4.4.

The parts and models used in this work

We use three kinds of parts in this work (Fig. 4.3): an L-shaped component built from two blocks, a set of four symmetric cylindrical legs, and a single block. Details on the L-component are available in Example 4.1 We provide three different kinds of L-component (Fig. 4.3, top): L1, where the vertical block can be along any side (four possible configurations), L2, where the vertical block is restricted to a long side of the horizontal block (two configurations), and L3, where it is on a short side (two configurations).

The set of cylindrical legs is parametrized in terms of the leg radius and the offset between the leg position and the corner, both of which are shared among all

[2]For efficiency, when we compute collisions involving cylinders, we approximate them as blocks having a square base, whose side equals the radius.

legs. This is shown in Fig 4.3 (bottom right), where the free parameters are denoted by the black arrows. Finally, the simple block part does not require any parameters, as we assume that it is as big as the part bounding box, which is encoded at the next level up.

This modeling system, together with the modest set of parts, is able to capture a large number of configurations common in the base structure of much furniture. We created 6 different furniture models: simple beds (a single block), beds with headrests (an L3 component), couches (an L2 component), tables (a stack of four legs and a single block for the top), chairs (a stack of four legs and an L1), and cabinets (a single block). Lastly, we still use thin blocks attached to a room wall to model frame categories (Figure 4.3, middle right), and we use the same categories from Chapter 3 (door, picture frame, window). Notice also that, with respect to Chapter 3, we use two more furniture categories, chairs, and beds with headboards.

4.3 Color likelihood

We augment the likelihood model from Chapter 3 with a new component promoting that pixels from the same color distribution are grouped together (Fig. 4.4). This is similar to evaluating the quality of the grouping provided by a 2D segmentation algorithm, with the key difference that the grouping is provided top-down by the model hypothesis. Note that the possible grouping hypotheses are significantly constrained compared with segmenting an image without any such guidance. In this scope, detailed geometry and 3D reasoning play an important role, as shown in Figure 4.1, where structures with legs provide a much better grouping than a plain block. Further, the reasoner does not need to entertain arbitrary groupings, as it would if it were doing bottom up clustering.

We consider two pixels in the same group if they are both part of the projection of the same object, or of the same room surface, but we consider walls as a single group, as they tend to be of the same color. Intuitively, two pixels p_i and p_j have a high probability of being together if their normalized distance d_{ij} in feature space

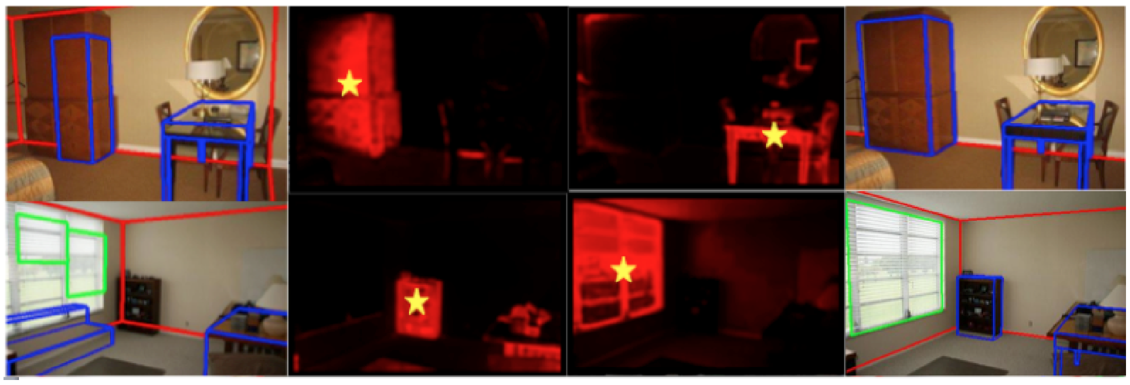
Figure 4.4: Our color likelihood encourages pixels similar in color to be grouped together. For each pixel, we compute a color histogram in LAB space. In columns 2 and 3, we select a pixel (marked with a yellow star) and compute the chi-square distance between its histogram and that of all other pixels. We show this in a heat map fashion, where we set the red channel inversely proportional to this distance. In the first row, pixels within the cabinet are very close in LAB space. We can see the benefits of using color by comparing the best fit found without color (column 1) and with color (column 4).

is small. We use

$$p(p_i, p_j|\theta) \approx {d_{ij}}^{(1-I_g(p_i,p_j,\theta))} * (1-d_{ij})^{\,I_g(p_i,p_j,\theta)}\ , \tag{4.7}$$

with $d_{ij} \in [0,1]$, and $I_g(p_i, p_j, \theta)$ is an indicator function that takes value 1 if the model assigns the two pixels to the same group, 0 otherwise. Notice that we have a high $p(p_i, p_j|\theta)$ in two complimentary cases: 1) the two pixels are assigned to the same group and their distance is small, as we want groups to be perceptually uniform; and 2) the two pixels are in different groups and the distance is large, as groups (objects) tend to be different from each other. We measure the global quality of the grouping over the entire image by averaging the pairwise contributions

$$p(I|\theta) \approx \frac{\sum_{i=1}^{i\leq N} \sum_{j=i}^{j\leq N} g(p_i, p_j|m)}{(N^2-N)/2} \tag{4.8}$$

where N is the number of pixels in the image.

This is a generic formulation of a grouping function, that allows for any choice of feature space. In this work, we experiment with color and use $d_{ij} = \chi(CH_i, CH_j)$, where $\chi(CH_i, CH_j)$ is the chi-square distance between the color histograms computed at pixels i and j over a window of size $n = 15$. We use histograms instead of simple pixel intensities because we want to capture the color distribution of objects and surfaces, which arise at a larger scale than the pixel level. We experimented with the LAB color space, and used a 3-dimensional histogram with 8 bins per dimension, where elements are softly assigned to bins. We denote the contribution of this grouping function based on color distance by $p(C|\theta)$.

For efficiency reasons, color histograms are pre-computed at initialization. To further reduce computation time, we only use the center pixels of 5-by-5 grid cells. We further cache computations between successive model hypothesis proposals.

This color component is then added to the full likelihood function in Eq. 3.12

$$p(D|\theta) \approx p(E_d|E_\theta)p(O_d|\theta)^\alpha p(GC_d|\theta)^\beta p(C|\theta)^\delta \quad . \tag{4.9}$$

As we did for α and β in Sec. 3.2, we set $\delta = 30$ by running our algorithm on the training portion of the Hedau dataset using the room box error as an objective function. Again, we used coarse grid search with a step of 2. For black and white images, we set $\delta = 10$, as only a third of the information is available.

4.4 Inference

We now discuss the modifications to the inference method used in Chapter 3, needed to handle the more complex geometry of the objects. Each object has now an additional set of parameters that needs to be inferred, which include the relative heights of its parts, and all the part parameters. We will see that simple modifications to three of the existing moves (Sec. 4.4.1) allow us to handle most of the complexities introduced by this new version of our model. In fact, the rest of the inference operates exactly as described in Sec. 3.3.

Then, we discuss dedicated data-driven strategies for specific parts, which are shared by all the object categories containing that part. This is needed because the inference of parts with complex geometry is more exacting than that of simple bounding boxes, and bottom-up cues are helpful in this process. We demonstrate this on the “legs” part. Objects supported by legs do not typically generate corners, which are needed for our standard data-driven proposal mechanism, and in Sec. 4.4.2 we discuss alternative bottom-up proposals for this type of objects.

Last (Sec. 4.4.3), we show how to inform the inference by using contextual relationships among the objects in the scene, which helps detect objects otherwise hard to find. Specifically, we propose chairs around tables in the current scene hypothesis.

4.4.1 Modifications to the existing sampling moves

We first modify the diffusion move Diffusion 1, which now consists of two steps. We first sample over the object center and dimensions as in the original version, and then sample over the subset of parameters $(hr_1, ..., hr_{n-1}, p_1, ..., p_n)$, using stochastic dynamics for 20 leapfrog steps. Notice that we do not sample over hr_n, which is not free, as the part heights must sum to 1. Intuitively, the first part of the move fits the object bounding box, whereas the second step infers the internal structure of the object. Since this move will be executed several times during inference, these two steps will be alternated many times. Importantly, we do not sample over internal part parameters that the modeler marked as fixed (Sec. 4.2.1). For all the others, we enforce that they are within the allowed range, and whenever they are outside, we bound them.

We then change the move adding an object to the scene (Jump Move 1, see Algorithm 3.1) to handle the new geometric model. This impacts only adding furniture, since frames are still approximated with a single block anchored to a wall. Whenever we add a new furniture object, we still rely on the usual method of proposing its position from a corner, and its size from the prior (steps 1-7 in Algorithm 3.1 are left unchanged). After this, we add a step to initialize the part heights and internal parameters. The part types and their arrangement is determined by the object category chosen in step 2 of Algorithm 3.1. Each parameter is either set to the corresponding fixed value, or to the mean of the corresponding allowed range, depending on the modeling choice. At this point, we choose the configuration c_i for each part of the object. Most of the parts are symmetric and this step is not necessary. For other type parts, we try all N_{pi} possible configuration, and keep the one maximizing the posterior. For example, for an object containing an L1 component, we need to determine to which side of the base the vertical block is attached to (Fig. 4.3, top row), and we try the four possible configurations. For objects containing an L2 or L3 component, we only need to try two configurations. We then modify steps 8 and 9 of Algorithm 3.1, by computing collisions among the parts, rather than just the bounding boxes of the objects. Last, during delayed acceptance (step 10) we use the modified version of Diffusion 1, which samples also over the part parameters.

We also modify Jump Move 4, which proposes changing the type of an object. In Chapter 3 this only changed the prior used to evaluate the object, now it also changes its internal structure. We do so by keeping the bounding box of the entire object fixed. For example, in case we propose changing a simple bed into a table, we replace the single block representation of the bed with four legs and a top, whose two bounding boxes combined will occupy the same amount of space as the bed block. When we switch to a different category, we initialize the part internal parameters as discussed above for Jump Move 1.

4.4.2 Data-driven inference for object parts

An advantage of replacing bounding boxes with detailed geometry is that we can use tailored inference strategies for the different topologies. We demonstrate this by introducing a new data-driven mechanism to propose objects with legs from pairs of adjacent vertical segments, called **pegs**. This method is used on top of the usual proposals from corners, which we keep using for all object categories. However, objects with legs typically do not generate corners, while the projection of a leg can be effectively approximated with a peg (Fig. 4.5, left).

To detect pegs we consider all segments converging to the vertical vanishing point, using the criterion illustrated in Fig. 2.9, with $\alpha < 0.12$. We then consider as pegs all the pairs of segments such that their distance is less than 20 pixels (Fig. 4.5, left). A peg can be then used to propose a four-legged component the same way that a corner is used to propose a block. We construct corners from pegs by first positioning a 2D corner at the bottom of the peg, and then creating corner segments through the corner position and the vanishing points (Fig. 2.10).

More effective proposals can be generated from two or even three pegs (Figure 4.5). In both cases, we back-project the bottom end of each peg from the image plane onto the room floor, just like we do with image corners (Fig: 2.11). When we use only two pegs, the distance between the two corresponding 3D points on the floor gives us either the width or the length of the object, depending on whether the line between the two points is aligned with the x-axis or the z-axis of the room box. We then sample the width (or length) from the prior for the category like in Sec. 3.3, and we repeat for the height. Notice that here we do it in a reverse order, as we sample the height conditioned on the width, while before we were doing the opposite. If the line between the two corners is not aligned with either dimension on the room box, we assume that these two points correspond to two legs along the object diagonal, and this gives us both width and length. We then need to sample only the height from the prior. When three pegs are used, the lines between the corresponding points on the floor must be roughly orthogonal, and we do not generate object proposals if this is not the case. This determines both width and length of the object, and only the height is left to sample from the prior.

Whether we used two or three pegs, we then treat this proposal as those generated from corners, meaning that we first propose the internal part parameters with the modified version of Jump Move described above. Then, we execute steps 8 through 11 of the standard Algorithm 3.1 (collision detection and delayed acceptance). What we just described is the equivalent of our jump move for adding a furniture object from a corner, where instead of using corners we use one, two or three pegs.

We also modify our procedure for initializing the object proposals. Before, we sorted the detected corners according to the posterior of the initial proposal generated from each of the corners (Sec. 3.3). Now, we also consider each peg, and each pair and triplet of pegs. We then sort the proposals generated from corners and those generated from pegs together based on the posterior, and record how they

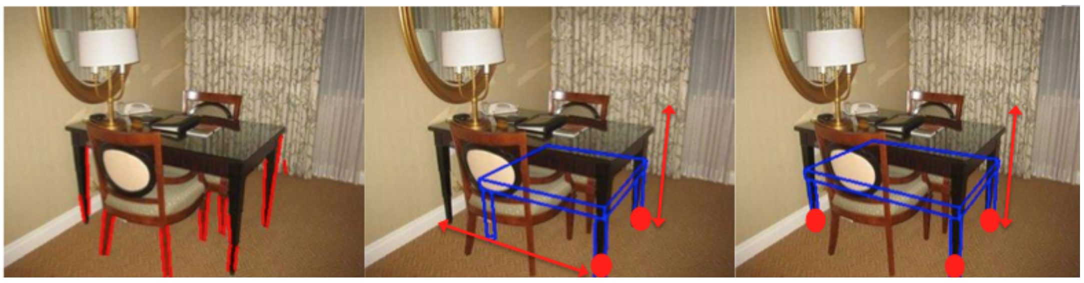

Figure 4.5: Using “pegs” to propose furniture with legs. Pegs are pairs of vertical segments detected on the image plane (left). A single peg can be used to propose a table or a chair just like an orthogonal corner is used to propose blocks. Proposing a table from two pegs (middle) requires estimating the width (or length) and the height of the table, proposing it from three (right) only leaves the height as a free parameter.

were generated (e.g from which corner, from which peg, from which pair of pegs, and so on). During inference, when we select a proposal from this list, we generate a sample from the recorded feature.

One last important detail is that we use the peg proposal mechanism only for object categories whose support (e.g. the lowest part in the vertical stack) is the “legs” part. Hence, during initialization and also for the rest of the inference, we only use pegs to propose samples of tables and chairs.

4.4.3 Top-down inference using context

Several objects in indoor images are hard to detect because of clutter and heavy occlusions. Chairs are an example, since they are often occluded, like the chair behind the table in Fig. 4.1. However, this problem can be addressed by considering contextual relationships between indoor objects in a top-down fashion.

Here, we bias the sampler to propose for chairs around detected tables, as shown in Figure 4.6. Given a table hypothesis (shown in blue), we look for chairs in the red areas, whose size and position is defined relatively to the table, by making sure that the backrest of the chair is facing the table. For each proposed chair, we briefly sample over its continuous parameters with Diffusion Move 1, and then accept it or reject it using the Metropolis-Hastings acceptance formula. This allowed us to

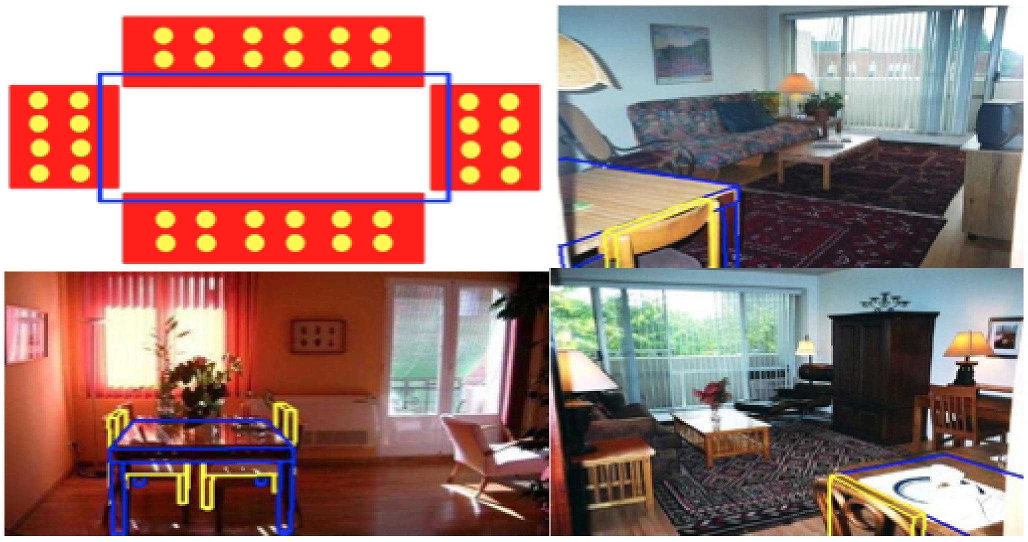

Figure 4.6: Using context to find chairs around tables. Given a table hypothesis (seen from above in blue, top left), we propose chairs around it. We consider the red areas around each side of the table, and propose a chair centered at each of the yellow dots shown. We use two lines of six dots on each side of the table. For each chair, we then briefly sample over its continuous parameters, and accept it or reject it using the Metropolis-Hastings acceptance formula. Chairs found with the help of this procedure are shown above in yellow.

Table 4.1: Effects of our color model on the room box layout error.

Method	Hedau dataset	UCB dataset
no color	13.6%	14.2%
with color	**12.7%**	**14.0%**
Schwing et al. (2012)	12.8%	NA

find the chairs drawn in yellow in Fig. 4.6, that were missed by inference without context cues.

At the end of the regular inference algorithm, we use this procedure for each table in the final sample, and results in the next section show that this allows to detect chairs that would be missed otherwise. Notice that this just a simple example demonstrating that there is merit in using contextual relationships. However, these promising results suggest that it is worth investigating such strategies in similar cases where contextual cues are strong hints of where to look.

4.5 Results

We start by evaluating our method on the room box error on the same two datasets used in the previous chapters. Table 4.1 shows the benefits of adding color, which are more modest on the black and white dataset, where only intensity information is available. We can see that the method proposed in this chapter (second row) performs slightly better than what was the previous state-of-the-art (Schwing et al., 2012).

We then evaluate on object recognition, where we expect that the innovations introduced here will provide significant improvements. We rely on the same criteria used in Chapter 3. We first measure how many objects we correctly identified for each of the two main categories (furniture and frames), even if there is confusion within the subcategories (e.g. when we label a table as a couch, or a window as a door). We provide precision and recall scores based on this criterion. Second, we measure the accuracy we achieved within each of the two categories, as the percentage of objects that were assigned to the correct subcategory.

In Table 4.2, we compare with the results of our method in Chapter 3, and we evaluate on the two main innovations introduced here, color and accurate geometric models. Results on the left refer to the Hedau dataset, results on UCB are on the right. As a first general observation, we found that the UCB dataset is more challenging, since most images are blurry and with heavy clutter, and they are all black and white, which results in the color model being less useful than on the color Hedau dataset.

We then consider results on furniture objects. For proper comparison with Chapter 3, where chairs were not part of the categories used for evaluation, we separately report results with and without chairs. Also, we consider beds with headrest and beds without headrest as both part of the category "bed". We can see that color and geometry combined improve all scores (row 3). In row two, we show the impact of adding color color to the model from Chapter 3, where every category is still approximated as a block. The color model segments objects from the background and from each other, thus improving global precision and recall. However, it does not improve subcategory recognition (on the UCB dataset we report a slightly worse score), since the priors are still the only component that discriminates among different object types.

When we add more accurate geometric models (row three), subcategory recognition improves, as now different categories have different geometry. On the other hand, the contribution of geometry to global precision and recall is more modest. Further, we report fewer false positives for categories with more sophisticated models, such as tables, because more refined geometry is less likely to be predicted in absence of enough supporting evidence. We see this in Table 4.3, where we compare our method using only blocks (left) with the one discussed here (right). We see in the last row of the confusion matrices that more geometry reduces the number of "table" false positives significantly (Table 4.3, last row). However, there is still confusion among categories relatively similar in shape and size, such as couches and beds.

When we also consider chairs, subcategory classification improves, despite the task being harder due to a larger number of categories (compare row three and row four in Table 4.2) However, recall suffers, as chairs are relatively small and often heavily occluded. Nonetheless, we notice the benefits of using context for proposing, which improves all measures (compare row four and five). This is a promising step towards dealing with heavy occlusion and scarce image evidence using top-down information. In the case of the Hedau dataset, context allowed us to identify seven more chairs at the cost of one false positive. Additionally, the trend that more complex geometry reduces the number false positives is confirmed by chairs. These objects are more complex than beds or couches, and in fact we report less "chair" false positives (Table 4.4, third row).

Last, we discuss results on frames. In this case, geometry has no direct impact, since all frames share the same geometric model (a thin block on the wall), and only size and position are helpful for classification. However, we see that the method discussed in this chapter (row seven) improves precision and recall also for frames, thanks mostly to color. We also report small improvements in subcategory recognition (see also the confusion matrices in Table 4.5). We posit that this is caused indirectly by color, which allows estimating the size of the frame more accurately, which is one of the hints we use to distinguish among frame categories.

4.5.1 Qualitative analysis

We provide detailed qualitative results to illustrate the merits and the current limitations of our approach. Promising results are shown in Fig. 4.7 and Fig. 4.8. We see that our method provides promising scene reconstructions for both color and black and white images. More results are available in Fig. 4.9, where we focus on the detected chairs and tables. The results in these three figures show that our method can recover a wide variety of objects, under very different poses.

An interesting observation is that our strong geometric model is robust to minor variations in the objects. For example, our couch model does not have armrests, but

Table 4.2: Precision, recall, and subcategory classification accuracy on the Hedau (left) and UCB (right) datasets.

Furniture no chairs	p	r	sc	p	r	sc
Chapter 3	53.5	28.5	51.3	33.0	29.7	50.0
color	53.8	34.8	51.6	37.2	31.0	49.5
geometry + color	**53.9**	**35.7**	**57.3**	**38.9**	**32.0**	**52.5**
Furniture with chairs						
geometry + color	53.8	26.2	58.6	37.8	22.0	52.5
geometry + color + context	**54.9**	**28.3**	**61.3**	**38.1**	**22.2**	**53.4**
Frames						
Chapter 3	37.5	35.5	66.0	28.4	37.3	59.8
geometry + color	**44.9**	**41.8**	**69.3**	**33.3**	**42.4**	**63.6**

Table 4.3: Furniture confusion on Hedau test set without including chairs. Results using the method in Chapter 3 (left) and the method discussed here (right)

	BED	**CAB.**	**COU.**	**TAB.**	**BED**	**CAB.**	**COU.**	**TAB.**
BED	14	1	9	1	19	1	14	4
CABINET	1	12	2	0	1	15	2	6
COUCH	6	1	7	5	4	6	11	2
TABLE	4	3	4	6	0	0	1	10

still allows to identify a few couches that have armrests (see the leftmost images in the second and third row in Fig. 4.7). Further, our models encode the key structure of the objects, and this is robust to minor mistakes in the fitting of the geometry. For example, we found the table in Fig. 4.7, bottom right, even if the predicted top is too thick.

A first limitation of our method is that, despite great improvements, there is still confusion among object categories, as illustrated in Fig. 4.10. For example, the table in the top left was confused with a cabinet, and the nightstand on the top right was wrongly labeled as a couch. In both cases the estimated 3D space occupancy (i.e. determining which 3D regions are free and which are occupied by something) is

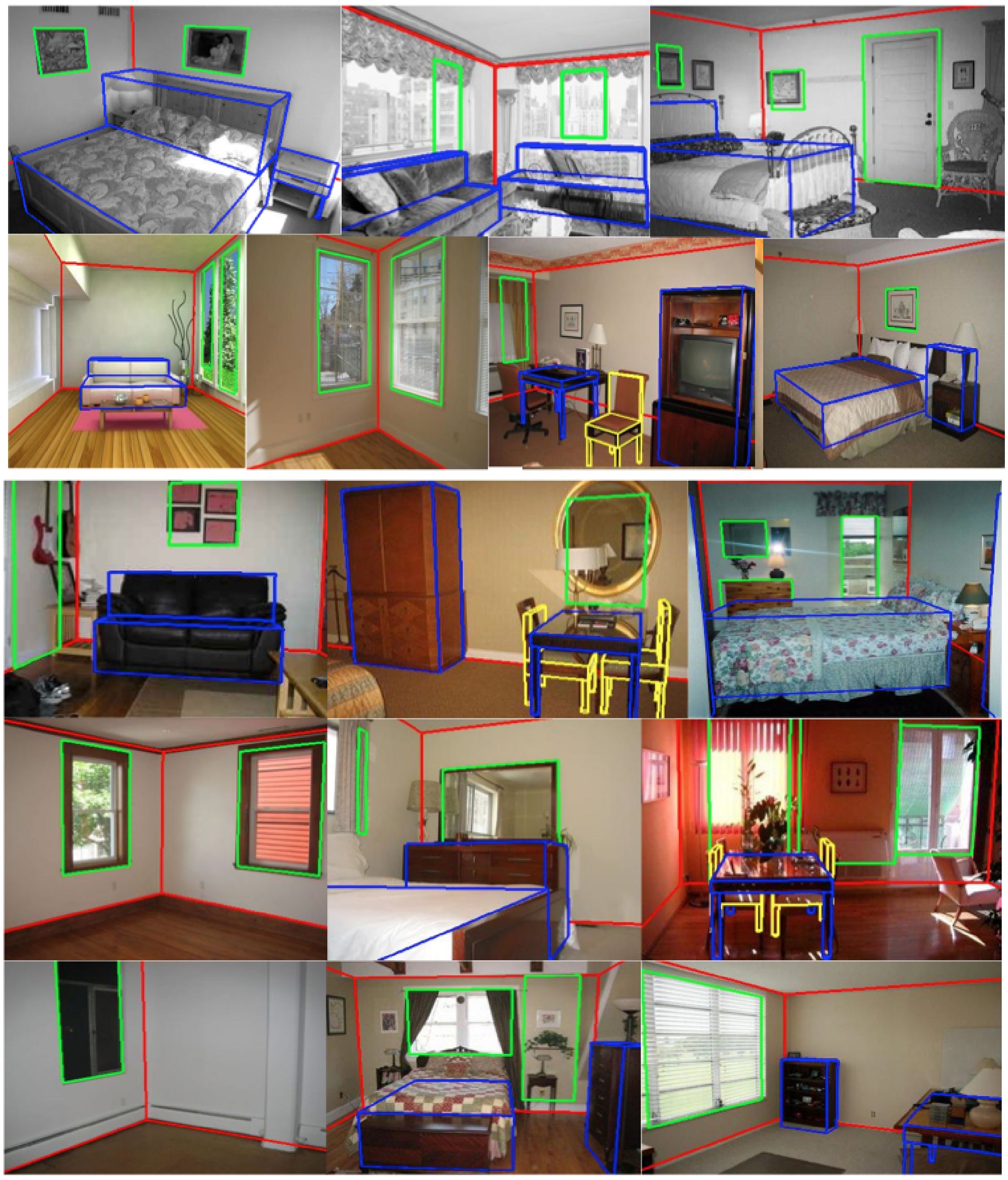
Figure 4.7: Promising examples of scene reconstructions

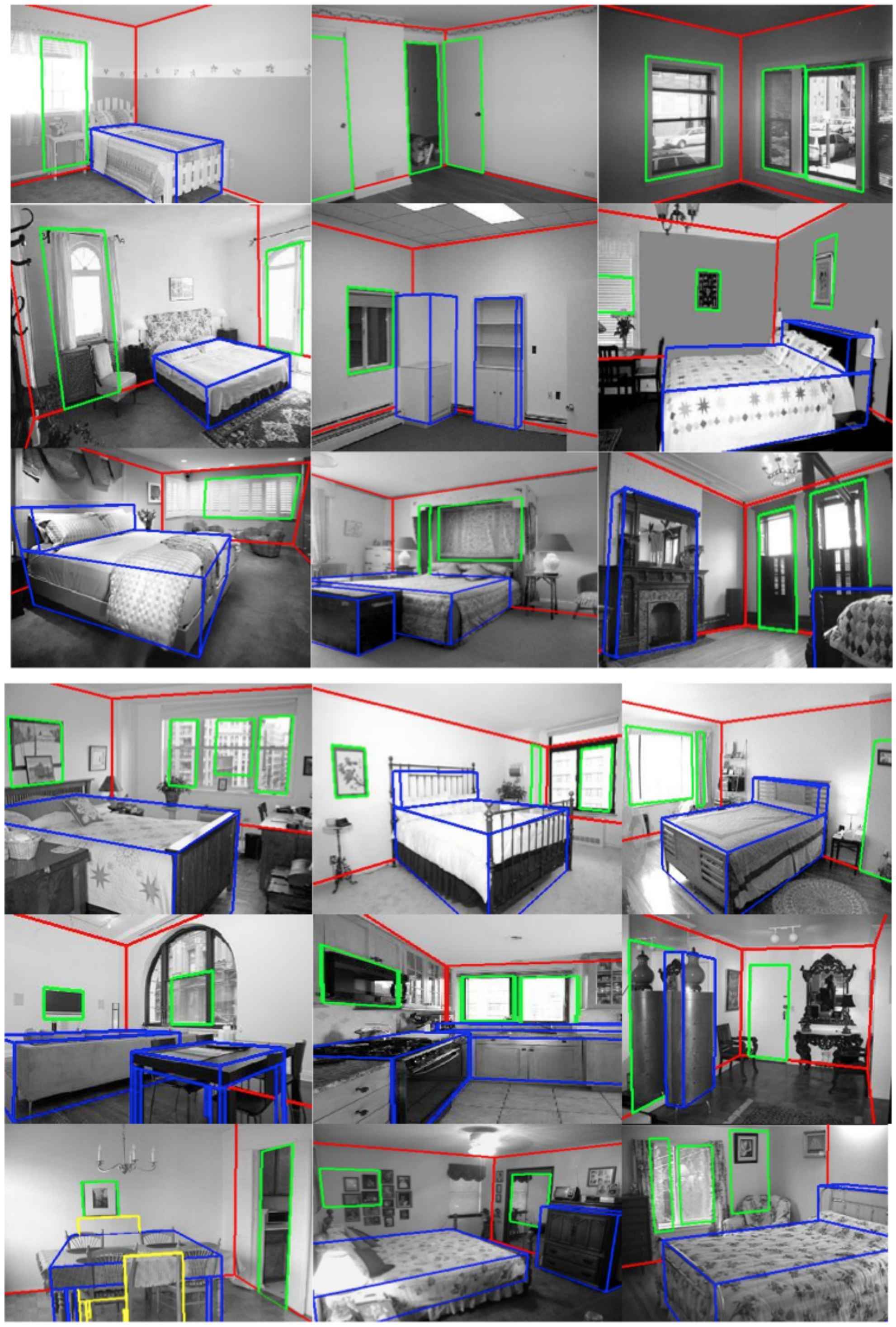

Figure 4.8: Promising examples of scene reconstructions from the black and white dataset

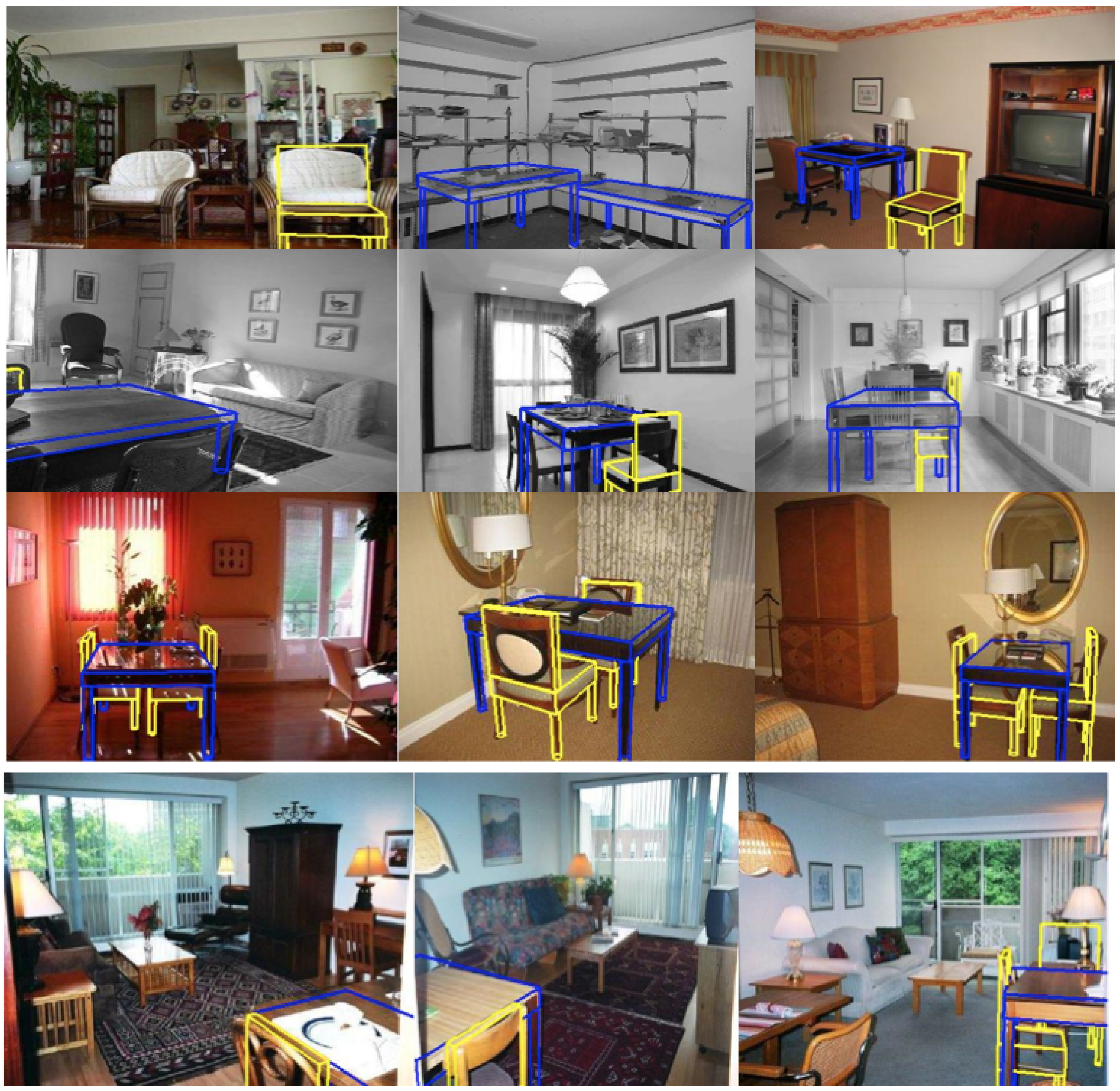

Figure 4.9: Some examples of chairs and tables found by our algorithm. The bottom three rows show chairs found using context. In the bottom right corner we see the only mistake made when using context on the color dataset. The chair on the left of the table is detected, but the fit is not good, while the chair predicted behind the table is not present in the image.

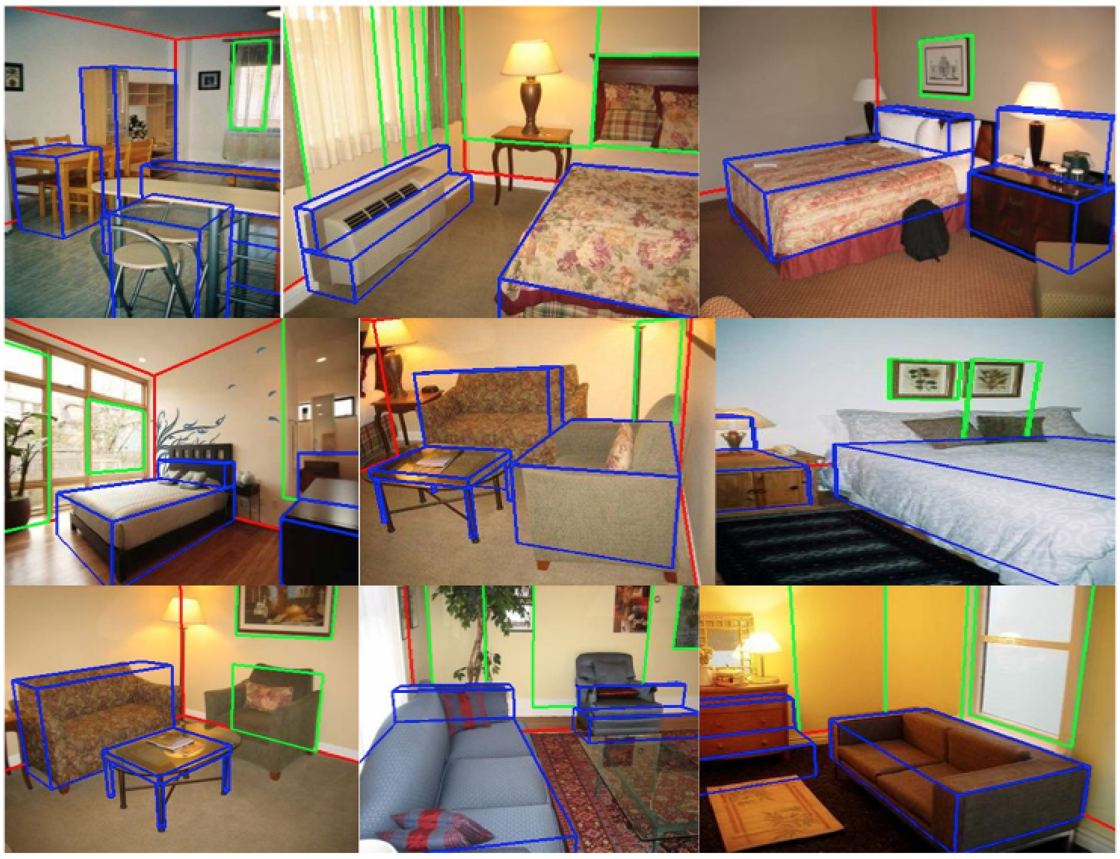

Figure 4.10: Some typical failures. Here, we show several cases where our method confused objects of different categories. For example, the table in the top left was confused with a cabinet, and the nightstand on the top right was wrongly labeled as a couch.

Table 4.4: Furniture confusion on Hedau test set, including chairs

	BED	**CABINET**	**CHAIR**	**COUCH**	**TABLE**
BED	19	1	0	14	4
CABINET	1	15	0	2	6
CHAIR	0	0	10	0	0
COUCH	4	6	0	11	2
TABLE	0	0	0	1	10

Table 4.5: Frame confusion on Hedau test set. Results using the method in Chapter 3 (left) and the method discussed here (right)

	DOOR	**PIC.**	**WIN.**	**DOOR**	**PIC.**	**WIN.**
DOOR	7	0	7	15	1	20
PICTURE	0	29	13	1	33	4
WINDOW	8	5	28	5	4	31

accurate, but the object was mislabeled. This is a common mistake, and we can see another example in the rightmost image in the second row. Here, the couch backrest is fit to objects on top of the nightstand, such as the lamp. In the middle image of the third row a bed is hypothesized where a couch is, the couch armrest being confused for the bed's headboard. While several beds with headboards are correctly detected (first row, right image, second row, left image) sometimes the headboard is confused with a frame, like in the middle image in the first row. Additionally, couches are sometimes labeled as bed (third row, left and right), and this is mostly due to the edge detector completely missing the couch inner edges. In one case, the armchairs is confused with a frame (bottom left). We believe that having different color and appearance models for each object class might address some of these failures, as we report much confusion among object categories that are similar in shape and size.

Many of these failures are also caused by mistakes and confusing evidence in the feature detection process, as shown in Fig. 4.11. Here, all the beds in the first row

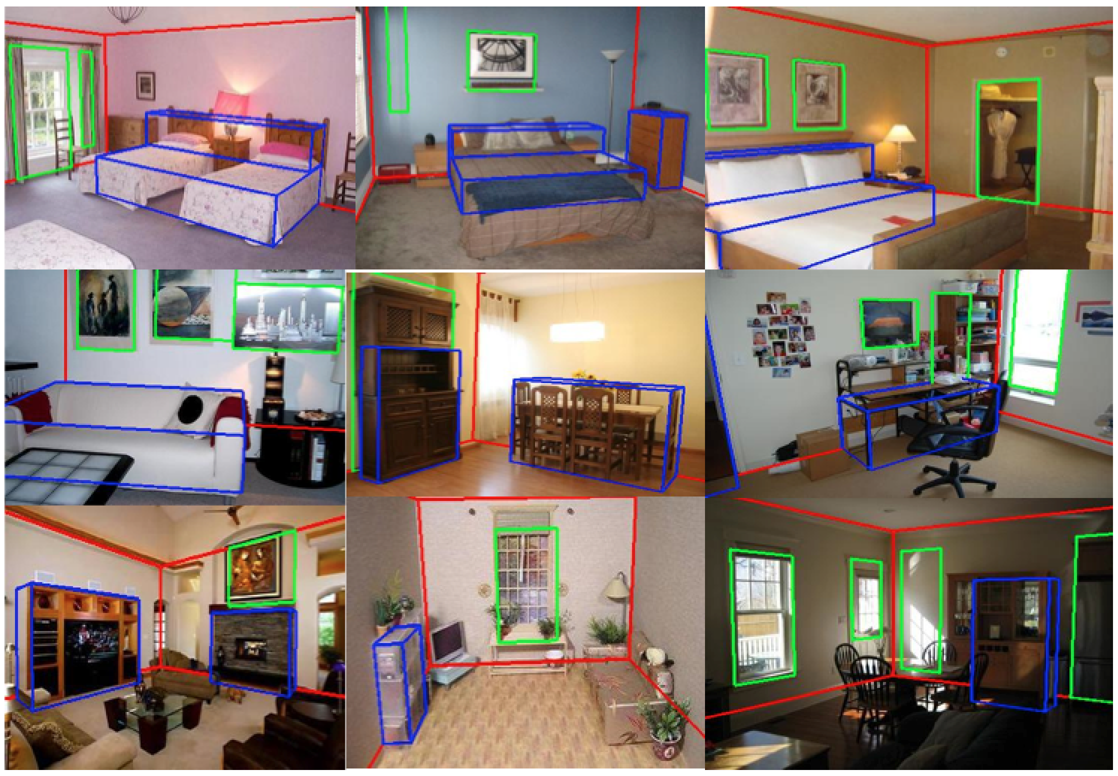

Figure 4.11: More typical failures. Mistakes shown here are due to errors and confusing evidence in the feature detection process. For example the edge between the floor and the bed in the top row (middle) is completely missed, and the model fits a couch to the edges corresponding to the blue blanket. See text for details.

were confused with couches. Consider for example the bed in the middle image. The edges between the bed silhouette and the floor are completely missed by the edge detector, and also the color model gets confused, since the bed and the floor are very similar in color. On the other hand, the edge detector finds the edges created by the blue blanket on the bed, and this explains the fit in the image. In the second row (left), we see the opposite problem, where the edge detectors misses the internal edges of the couch, and this results in our algorithm fitting a bed. Sometimes heavy clutter greatly confuses the edge detector and the other features detector we use, and this results in confusing tables with cabinets or beds (second row, middle and right image). Last, we see in the third row that several objects are missed in blurry images, especially if they are small in size or when most of the object silhouette is missed by the edge detector, such as the couch in the middle image.

More similar examples are shown in Fig. 4.12. In general, the lack of finesse of the edge detector is a problem in images with heavy clutter, where most edges get missed, such as those in the second row. Further, it is hard to recover from large errors in the initial vanishing point estimation, which results in a poor initial estimate of the camera, as shown in the last row, where the camera error gets less severe as we go from left to right.

Last, we consider errors on estimating the room box. Small mistakes in positioning the floor have a negative impact on object estimation, as illustrated in Fig. 4.13. Error in room box estimation are often caused by mistakes in the process of detecting orientation maps and geometric context (Fig. 4.14).

4.6 Discussion

The main contribution of this chapter is a framework for fitting 3D complex models of objects in the context of the overall scene. Our results show that this improves object recognition, particularly when it comes to distinguishing among different object types (e.g. tables from beds), since accurate geometry is a very strong cue for this purpose.

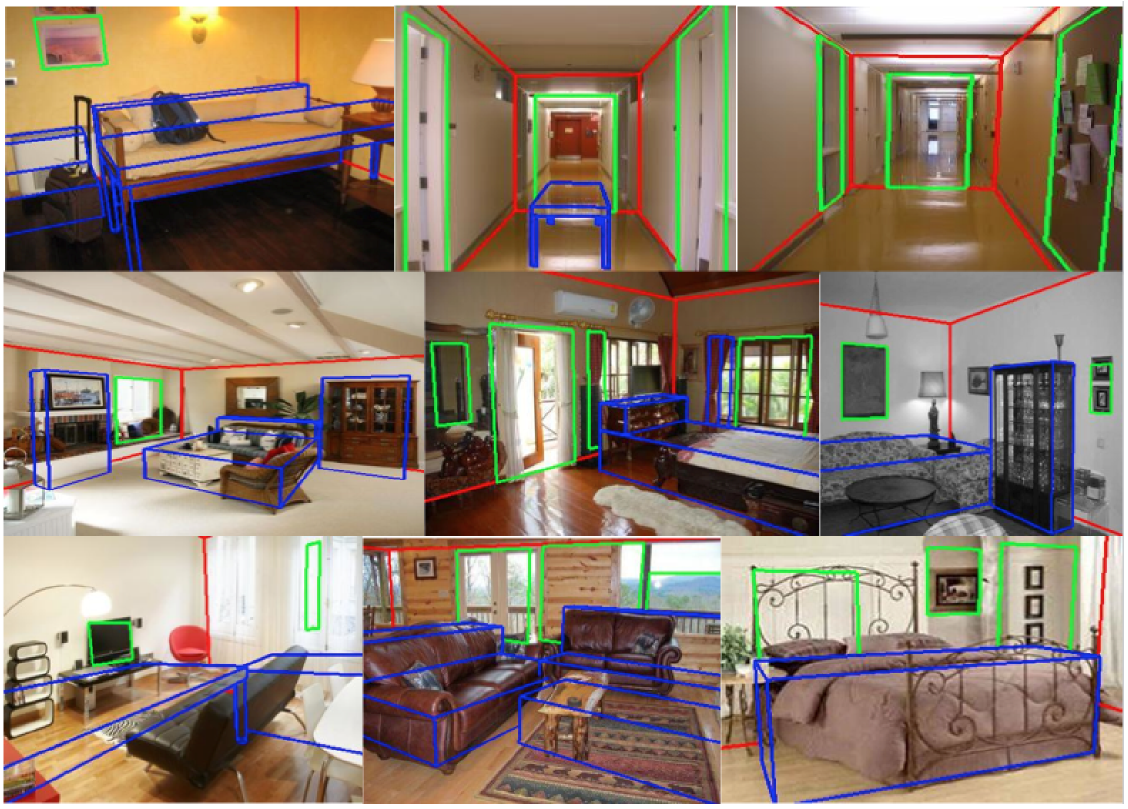

Figure 4.12: More failures due to lack of finesse in the detected features. First row: some objects hallucinated by our algorithm due to edges arising from shadows (the table in the middle image), or artifacts (the door in the right image). Second row: heavy clutter is a major sort of confusion, and some times objects are grouped together. Third row: bad camera parameters due to large errors in the initial estimate of the vanishing points.

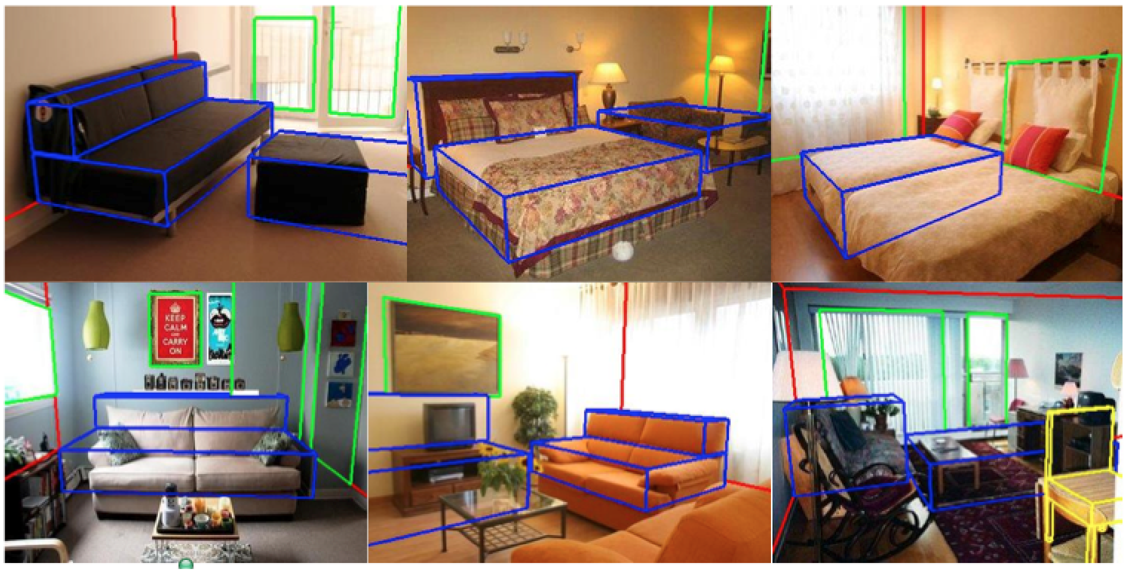

Figure 4.13: Failures in estimating the room box. Small mistakes in positioning the floor greatly affect object estimation. For example, positioning the floor too far behind results in having a cabinet behind the bed as opposed to a bed with a headboard (top middle), or poor couch estimates (top left, bottom left and middle). Sometimes the walls are not positioned correctly (bottom right).

The part-based representation allows sharing functionalities both in the modeling and in the inference. Dedicated inference for the available parts introduces two advantages: 1) it allows adding new models built from the available parts without worrying about inference; and 2) it helps fitting more complex geometry, which is harder than simple 3D blocks, by exploiting part-specific characters (for example, proposing legs from "pegs").

The proposed parametrization of objects and parts further demonstrates the advantages of a strong 3D representation. A very important one is that variation within instances of an object category is reduced because the camera does not contribute to it, and also defining parts relatively to an object's size instead of absolute values further reduces the variability among classes. Consider for example the table model, where we do not impose tables to be any particular height, but the relative amount for the legs part versus the top part is kept within a small learned range (around 90% for the legs). Despite using a small range, learned from a limited

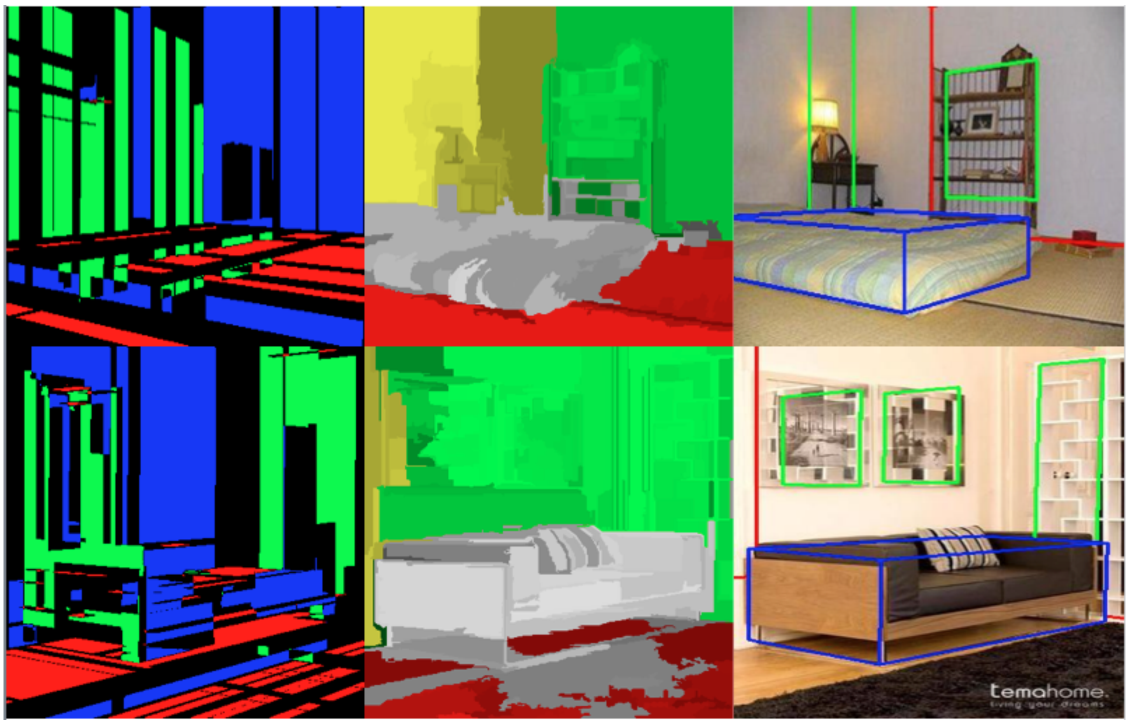

Figure 4.14: Errors in estimating the room box caused by orientation maps and geometric context. In each row, we show from left to right, the estimated orientation maps, the geometric context labels and the final fit. For the orientation maps, we draw pixels assigned to one of the Manhattan directions with three different colors: red pixels correspond to horizontal surfaces, blue and green to the two types of orthogonal, vertical surfaces. The color coding for geometric context is: red-floor, yellow-left wall, green-middle wall, blue-right wall, gray-object,light green-ceiling. First row: both features agree in positioning the boundary between the walls where the cabinet is, and this confuses our model, which uses a door to explain the edge between the left and the right wall. A similar situation occurs in the second row.

amount of training data, we could detect a variety of tables in the test data, ranging from small coffee tables (Fig. 4.10, bottom left) to dining tables (Fig. 4.6, bottom left). Additionally, since we encode the key structure of an object, minor variations in the object parts do not necessarily create problems. For example, in Fig. 4.4 a table is detected even if the predicted top is too thick (bottom right).

The experiments also showed that the proposed inference can handle complex 3D models, which introduce a larger (and unknown) number of variables, without being too sensitive to local optima. This is enabled by the fact that objects only interact with others in minimal ways via occlusions and space occupancy constraints. Hence, proposing a complex alternative to a bounding box is like an independent local part of the inference, unless it changes what is occluded with what, like switching a block into a table so that chairs can be tucked underneath. However, truly complex objects, such as an exuberant indoor plant, will require additional and potentially quite different approaches.

Another important observation is that the more precise mapping of object hypotheses to image pixels enables better exploitation of appearance cues, such as the approach to color proposed here. The way we use color would in fact be confused by the background if we modeled a concave object like a table with a convex block (Fig, 4.1). Results show that color improves object recognition and reduces the room box error. However, this model is very simple, and assumes that each object is roughly uniform in color. This is a strong assumption, and softer models, perhaps allowing multiple modes in the color distribution of a projected object, might be needed. Further, we do not exploit the difference in appearance among different classes, since we use color only to segment objects from the background and from each other. Training more sophisticated, class-dependent color and texture models is a natural extension of our method.

It is also important to notice the difference between the way we use color and the other components in the likelihood. In fact, edges, orientation maps and geometric context are estimated bottom-up at at the beginning of the inference, and this requires making "hard" choices, such as grouping pixels in the same edge or in the same oriented surface. Unfortunately, errors in this process propagate throughout the entire inference. On the other hand, we evaluate color on the grouping provided top-down by the model hypothesis, and this does not require any “hard” choices before the inference has even started (the pre-processing we do is only for speeding up the computation). Common failures caused by mistakes and lack of finesse in the detection of the other features (figures 4.10 through 4.14) suggest that using a more top-down approach to feature detection might be beneficial.

Last, the way we detect chairs around tables is a promising step towards using contextual relationships to drive the inference in a top-down fashion. Notice that proposing chairs around tables would not have been possible with the traditional single-block approximation, which does not allow sliding a chair under a table. However, in this work we only used this relationship, which was defined manually for demonstration purposes. A more principled way to model such contextual relationships is needed in order to exploit a larger number of more sophisticated interdependencies. Defining new relationships will also require adding more objects and parts to our framework, to augment the relatively small set used here.

CHAPTER 5

Discussion

The work presented here makes a substantial contribution to the goal of scene understanding from a single image, applied to the domain of indoor scenes. We introduced a principled representation to jointly infer the camera parameters, reconstruct the 3D layout of the environment, and identify the objects in the scene, all from a monocular image. Our representation unifies geometry and semantics and allows them to interact to provide a global, coherent interpretation of the image. We advocate that this **unified representation** distinguishes our work from traditional methods for indoor scene understanding.

We developed this representation within a generative **Bayesian** framework, which allowed us to model the rich structure of indoor scenes in a principled way. We showed how our model can be easily extended to integrate both new evidence from the data and new prior knowledge of the world. In fact, throughout this work we extended our method to incorporate more sophisticated image likelihoods and 3D models, and the inference process, which is separated from the modeling in our Bayesian framework, naturally supported these additions with minor modifications.

In terms of **geometry**, we introduced a much more accurate and realistic representation than the standard paradigm of approximating each object with a block. We demonstrated this by showing that detailed models of furniture, such as tables with legs, improve object recognition. Modeling the camera, the 3D geometry and the image likelihood separately allowed us to address the ambiguities of the imaging process using properties of the 3D world, which are independent of images. For example, we modeled the statistical variations of an object's structure in 3D, where they are not impacted by the variance introduced by the camera.

In terms of **semantics**, our method goes beyond previous work enforcing simple physical constraints of the scene, since in our case the geometry of an indoor object depends on its semantic category and on the overall scene. We demonstrated this by using prior distributions to condition an object's size and position on its type and on the room box, and this improved object recognition and also the global scene reconstruction.

We developed an **inference** method to estimate all the elements in the scene **jointly** while also capturing and exploiting their interdependencies, since we believe that understanding an image as a whole is more powerful than estimating the individual elements separately. This was demonstrated by showing that joint inference of objects, room box, and camera, improved over estimating each of them independently. Importantly, as our model became more realistic and more global, the reconstruction of the entire scene improved. For example, using objects of realistic size indirectly improved the reconstruction of the room box. Similarly, contextual relationships among objects improved object recognition.

We introduced an **MCMC engine** that supported the inference of such a rich model and of its very complex dependencies. This is another distinctive feature of our work, which can prove very useful to the field. It is in fact challenging to extend the traditional structured prediction techniques to handle the rich structure and the vast dimensionality of the output space used here, but we showed that our sampling method can handle these complexities.

This was achieved by letting **top-down and bottom-up cues** interact to inform the inference. Proposing the 3D position of an object from an image corner, and its size from its categorical prior, is an example of this. Our use of top-down enabled tackling the inherent ambiguities in recovering 3D structure from 2D image data. This is exemplified by our ability of recovering the room box even if its boundaries are mostly hidden, by jointly inferring the objects in front of the camera to explain the occlusions. We also demonstrated this using contextual relationships between tables and chairs to improve the performances on chair detection, which is usually hard due to heavy occlusion and limited image evidence.

All these concepts were **evaluated** on standard benchmark datasets using conventional criteria. We provided extensive comparison to the relevant methods in the field, and showed that our approach is comparable and often exceeds state-of-the-art methods on the task of indoor scene reconstruction. We provided extensive ground truth annotations for an existing dataset, and this will allow other researchers to compare to our method, as well as facilitate the dissemination of the ideas introduced here.

We conclude by emphasizing that our method enables much more than just recognition in images. In fact, the way we extract the 3D layout of the scene and its semantics from an image enables interacting with the 3D environment. Our 3D representation could allow an agent to navigate in the environment autonomously. However, we extract much more than just 3D geometry, since the retrieved semantic information can support several other high-level activities. Inferring the identity of the objects in the scene informs about their function and their properties. An agent could then perform more complex tasks than basic navigation, for example searching for an object in the scene, or predict and assist human activities. We believe that the unified representation and inference of indoor scenes developed in this work are promising steps towards these exciting goals.

5.1 Future work

Straightforward extensions of the algorithm include adding more object and part models to the framework, as well as more contextual relationships. In this scope, we are developing a GUI that allows the modeler to create new category models from the available parts in a simple and interactive way. Further, we want to model contextual relationships in a more principled way, such that the modeler can add new ones to the framework with a simple interface, just like for the case of object models.

A current limitation of the algorithm is that shape and size are sometimes not discriminative enough to distinguish between objects. Notice that there is no component in our image model considering the likelihood of the appearance of a specific object given its class, since all our likelihood components are a global function of the entire scene. Hence, we plan to train specific appearance models for each object category. Specifically, we plan to use Histogram of Oriented Gradients (HOGs), which proved to be very discriminative in the domain of 2D object classifiers. However, we plan to model the appearance in 3D, by mapping the image pixels to the 3D surface that generated it according to the current scene and camera hypothesis. This should remove the variations in the object's appearance introduced by the camera, as it is the case for the variations in the 3D geometry of the objects. Further, the 3D hypothesis would enable reasoning about occlusions, which is challenging in the case of 2D methods based on appearance. Similarly, we want to extend the approach to color discussed in Chapter 4, by learning a specific color model for each object category.

Another direction of research we plan to investigate is making the features detectors used here more robust, by considering top-down information. Features such as edges are detected bottom-up before inference, and this requires making "hard" choices, such as grouping pixels in the same edge. As a consequence, errors in this process propagate throughout the entire inference, for example by trying to fit an object to noise that the edge detector mistook for an edge. Further, these "hard" choices do not integrate well with our Bayesian reasoning. However, the color model introduced in Chapter 4 does not require any early grouping, and is evaluated top-down given the model hypothesis. We are keen to investigate whether this kind of reasoning is possible for other features, like edges.

Last, we are currently trying to understand images of indoor scenes with available text descriptions. The text potentially provides valuable information on the objects in the scene, their arrangement in 3D, and their appearance, and could thus be used to inform the inference. Conversely, the model reconstructed from the image could help disambiguate the natural language in the text, which suggests that these two problems should be solved jointly. Interestingly, this is similar to having a speaker instructing a machine about the structure of the surrounding environment.

APPENDIX A

Efficient rendering of model hypotheses with OpenGL

This Appendix will contain the technical details on how to efficiently render a model hypothesis during inference by using OpenGL and modern rendering hardware and software.

APPENDIX B

Manually estimating camera and room box from an image

This Appendix will contain the details on how we manually prepared the ground truth camera and 3D model parameters for the images we used for training and evaluation. This goes beyond the manual 2D labels traditionally used for evaluation in this domain.

REFERENCES

Agin, G. J. and T. O. Binford (1976). Computer Description of Curved Objects. *IEEE Trans. Computers*, **25**(4), pp. 439–449.

Andrieu, C., N. d. Freitas, A. Doucet, and M. I. Jordan. (2003). An introduction to MCMC for machine learning. *Machine Learning*, **50**(1), pp. 5–43.

Berg, A. C., T. L. Berg, and J. Malik (2005). Shape Matching and Object Recognition Using Low Distortion Correspondences. In *CVPR (1)*, pp. 26–33.

Bernstein, E. J. and Y. Amit (2005). Part-Based Statistical Models for Object Classification and Detection. In *CVPR (2)*, pp. 734–740.

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, **94**, pp. 115–147.

Blei, D., A. Ng, and M. Jordan (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, **3**, pp. 993–1022.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**(6), pp. 679–698.

Chakravarty, I. and H. Freeman (1982). Characteristic Views As A Basis For Three-Dimensional Object Recognition. pp. 37–45. doi:10.1117/12.933609.

Cornelis, N., K. Cornelis, and L. Van Gool (2006). Fast Compact City Modeling for Navigation Pre-Visualization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pp. 1339–1344. ISSN 1063-6919. doi:10.1109/CVPR.2006.118.

Coughlan, J. M. and A. L. Yuille (1999). Manhattan World: Compass Direction from a Single Image by Bayesian Inference. In *International Conference on Computer Vision*, pp. 941–947.

Coughlan, J. M. and A. L. Yuille (2003). Manhattan World: Orientation and Outlier Detection by Bayesian Inference. *Neural Computation*, **15**(5), pp. 1063–1088.

Crandall, D. J. and D. P. Huttenlocher (2006). Weakly Supervised Learning of Part-Based Spatial Models for Visual Object Recognition. In *ECCV (1)*, pp. 16–29.

Dalal, N. and B. Triggs (2005). Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pp. 886–893.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. *Computer Vision, IEEE International Conference on*, **2**, pp. 1403–1410 vol.2. doi:10.1109/iccv.2003.1238654.

Del Pero, L., J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard (2012). Bayesian geometric modeling of indoor scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Del Pero, L., J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard (2013). Understanding Bayesian rooms using composite 3D object models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Del Pero, L., J. Guan, E. Brau, J. Schlecht, and K. Barnard (2011). Sampling bedrooms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Delage, E., H. Lee, and A. Ng (2006). A Dynamic Bayesian Network Model for Autonomous 3D Reconstruction from a Single Indoor Image. In *CVPR*.

Delaitre, V., D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros (2012). Scene Semantics from Long-Term Observation of People. In *ECCV (6)*, pp. 284–298.

Everingham, M., L. Gool, C. K. Williams, J. Winn, and A. Zisserman (2010). The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*, **88**(2), pp. 303–338. ISSN 0920-5691. doi:10.1007/s11263-009-0275-4.

Fei-Fei, L., R. Fergus, and P. Perona (2003). A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories. In *ICCV*, pp. 1134–1141.

Fei-Fei, L., R. Fergus, and P. Perona (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*.

Fei-Fei, L. and P. Perona (2005). A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *IEEE Comp. Vis. Patt. Recog (CVPR)*.

Felzenszwalb, P., R. Girshick, D. McAllester, and D. Ramanan (2009). Object Detection with Discriminatively Trained Part-Based Models. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*.

Felzenszwalb, P., D. McAllester, and D. Ramanan (2008). A Discriminatively Trained, Multiscale, Deformable Part Model. In *CVPR*.

Felzenszwalb, P. F. and D. P. Huttenlocher (2005). Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, **61**(1), pp. 55–79.

Fergus, R., P. Perona, and A. Zisserman (2003). Object Class Recognition by Unsupervised Scale-Invariant Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ferrari, V., L. Fevrier, F. Jurie, and C. Schmid (2008). Groups of Adjacent Contour Segments for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Fidler, S., S. Dickinson, and R. Urtasun (2012). 3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model. In Bartlett, P., F. Pereira, C. Burges, L. Bottou, and K. Weinberger (eds.) *NIPS*, pp. 620–628.

Fischler, M. and R. Elschlager (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computer*, **22**(1), pp. 67–92.

Flint, A., C. Mei, I. D. Reid, and D. W. Murray (2010). Growing semantically meaningful models for visual SLAM. In *CVPR*, pp. 467–474.

Flint, A., D. W. Murray, and I. Reid (2011). Manhattan scene understanding using monocular, stereo, and 3D features. In *ICCV*, pp. 2228–2235.

Forsyth, D. A., J. A. Haddon, and S. Ioffe (2001). The Joy of Sampling. *International Journal of Computer Vision*, **41**(1/2), pp. 109–134.

Fouhey, D. F., V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic (2012). People Watching: Human Actions as a Cue for Single View Geometry. In *ECCV (5)*, pp. 732–745.

Furukawa, Y., B. Curless, S. M. Seitz, and R. Szeliski (2009). Reconstructing building interiors from images. In *ICCV*, pp. 80–87.

Gibson, J. J. (1977). *The Theory of Affordances*. Lawrence Erlbaum.

Girshick, R. B., P. F. Felzenszwalb, and D. A. McAllester (2011). Object Detection with Grammar Models. In *NIPS*, pp. 442–450.

Grabner, H., J. Gall, and L. J. V. Gool (2011). What makes a chair a chair? In *CVPR*, pp. 1529–1536.

Grauman, K. and T. Darrel (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**(4), pp. 711–732.

Green, P. J. (2003). Trans-dimensional markov chain monte carlo. In *Highly Structured Stochastic Systems*.

Gupta, A., S. Satkin, A. A. Efros, and M. Hebert (2011). From 3D Scene Geometry to Human Workspace. In *CVPR*.

Han, F. and S.-C. Zhu (2005). Bottom-up/top-down image parsing by attribute graph grammar. In *International Conference on Computer Vision*, volume 2, pp. 1778–1785.

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

Hedau, V., D. Hoiem, and D. Forsyth (2009). Recovering the Spatial Layout of Cluttered Rooms.

Hedau, V., D. Hoiem, and D. Forsyth (2010). Thinking Inside the Box: Using Appearance Models and Context Based on Room Geometry. In *ECCV*.

Hedau, V., D. Hoiem, and D. A. Forsyth (2012). Recovering free space of indoor scenes from a single image. In *CVPR*, pp. 2807–2814.

Hoiem, D., Y. Chodpathumwan, and Q. Dai (2012). Diagnosing Error in Object Detectors. In *ECCV (3)*, pp. 340–353.

Hoiem, D., A. Efros, and M. Hebert (2005a). Geometric Context from a Single Image. In *ICCV*.

Hoiem, D., A. A. Efros, and M. Hebert (2005b). Automatic photo pop-up. *ACM Trans. Graph.*, **24**(3), pp. 577–584. ISSN 0730-0301. doi:10.1145/1073204.1073232.

Hoiem, D., A. A. Efros, and M. Hebert (2006). Putting Objects in Perspective. In *CVPR*.

Huttenlocher, D. and S. Ullman (1990). Recognizing solid objects by alignment with an image. *IJCV*, **5**(2), pp. 195–212.

Jurie, F. and C. Schmid (2004). Scale-invariant shape features for recognition of object categories. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–90–II–96 Vol.2. ISSN 1063-6919. doi:10.1109/CVPR.2004.1315149.

Karsch, K., V. Hedau, D. Forsyth, and D. Hoiem (2011). Rendering synthetic objects into legacy photographs. In *SIGGRAPH Asia*. ISBN 978-1-4503-0807-6. doi:10.1145/2024156.2024191.

Kim, Y. M., N. J. Mitra, D.-M. Yan, and L. Guibas (2012). Acquiring 3D indoor environments with variability and repetition. *ACM Trans. Graph.*, **31**(6), pp. 138:1–138:11. ISSN 0730-0301. doi:10.1145/2366145.2366157.

Koenderink, J. (1976). The singularities of the visual mapping. *Biological Cybernetics*, **24**(1), pp. 51–59. Cited By (since 1996) 88.

Koppula, H. S., A. Anand, T. Joachims, and A. Saxena (2011). Semantic Labeling of 3D Point Clouds for Indoor Scenes. In *NIPS*, pp. 244–252.

Kosecka, J. and W. Zhang (2002). Video Compass. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pp. 476–490. Springer-Verlag, London, UK, UK. ISBN 3-540-43748-7.

Kosecka, J. and W. Zhang (2005). Extraction, Matching and Pose recovery based on dominant rectangular structures.

Kushal, A., C. Schmid, and J. Ponce (2007). Flexible Object Models for Category-Level 3D Object Recognition. In *CVPR*.

Lazebnik, S., C. Schmid, and J. Ponce (2003). Affine-invariant local descriptors and neighborhood statistics for texture recognition. In *International Conference on Computer Vision*.

Lee, D., A. Gupta, M. Hebert, and T. Kanade (2010). Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces. In *NIPS*.

Lee, D., M. Hebert, and T. Kanade (2009). Geometric reasoning for single image structure recovery. In *CVPR*.

Leordeanu, M., M. Hebert, and R. Sukthankar (2007). Beyond Local Appearance: Category Recognition from Pairwise Interactions of Simple Features. In *CVPR*.

Liebelt, J. and C. Schmid (2010). Multi-view object class detection with a 3D geometric model. In *CVPR*.

Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, **31**, pp. 3555–3395.

Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(5), pp. 441–450.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**(2), pp. 91–110.

Micusk, B., H. Wildenauer, and J. Kosecka (2008). Detection and matching of rectilinear structures. In *CVPR'08*, pp. –1–1.

Mikolajczyk, K. and C. Schmid (2004). Scale & Affine Invariant Interest Point Detectors. *Int. J. Comput. Vision*, **60**(1), pp. 63–86. ISSN 0920-5691. doi: 10.1023/B:VISI.0000027790.02288.f2.

Mundy, J. L. (2006). Object Recognition in the Geometric Era: A Retrospective. In *Toward Category-Level Object Recognition*, pp. 3–28.

Neal, R. M. (1993). Probabilistic inference using Markov Chain Monte Carlo methods. Technical report.

Newman, P. M., D. M. Cole, and K. L. Ho (2006). Outdoor SLAM using Visual Appearance and Laser Ranging. In *ICRA*, pp. 1180–1187.

Ott, P. and M. Everingham (2011). Shared parts for deformable part-based models. In *CVPR*, pp. 1513–1520.

Pollefeys, M., D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles (2008). Detailed Real-Time Urban 3D Reconstruction from Video. *Int. J. Comput. Vision*, **78**(2-3), pp. 143–167. ISSN 0920-5691. doi:10.1007/s11263-007-0086-4.

Pontil, M. and A. Verri (1998). Support Vector Machines for 3D Object Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**(6), pp. 637–646.

Pope, A. R. and D. G. Lowe (1996). Learning Appearance Models for Object Recognition. In *Object Representation in Computer Vision*, pp. 201–219.

Ren, X., L. Bo, and D. Fox (2012). RGB-(D) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2759–2766. ISSN 1063-6919. doi:10.1109/CVPR.2012.6247999.

Roberts, L. G. (1965). Machine Perception of three-dimensional solids. *Optical and Electrooptical Information Processing*, pp. 159–197.

Rother, C. (2002). A new approach to vanishing point detection in architectural environments. **20**(9-10), pp. 647–655.

Satkin, S., J. Lin, and M. Hebert (2012). Data-Driven Scene Understanding from 3D Models. In *BMVC*.

Savarese, S. and L. Fei-Fei (2007). 3D generic object categorization, localization and pose estimation. In *IEEE Intern. Conf. in Computer Vision (ICCV)*.

Savarese, S. and L. Fei-Fei (2008). View synthesis for recognizing unseen poses of object classes. In *European Conference on Computer Vision (ECCV)*.

Saxena, A., S. H. Chung, and A. Y. Ng (2005). Learning Depth from Single Monocular Images. In *NIPS*.

Saxena, A., S. H. Chung, and A. Y. Ng (2008). 3-D Depth Reconstruction from a Single Still Image. *International Journal of Computer Vision*, **76**(1), pp. 53–69.

Schiele, B. and J. L. Crowley (2000). Recognition without Correspondence using MultidimensionalReceptive Field Histograms. *Int. J. Comput. Vision*, **36**(1), pp. 31–50. ISSN 0920-5691. doi:10.1023/A:1008120406972.

Schindler, G. and F. Dellaert (2004). Atlanta World: An Expectation Maximization Framework for Simultaneous Low-level Edge Grouping and Camera Calibration in Complex Man-made Environments. In *In Int. Conf. on Computer Vision and Pattern Recognition*, pp. 203–209.

Schlecht, J. and K. Barnard (2009a). Learning models of object structure. In *NIPS*.

Schlecht, J. and K. Barnard (2009b). Learning models of object structure. Technical report, University of Arizona.

Schlecht, J., K. Barnard, E. Spriggs, and B. Pryor (2007). Inferring grammar-based structure models in 3D microscopy data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Schneiderman, H. and T. Kanade (2000). A Statistical Method for 3D Object Detection Applied to Faces and Cars. In *CVPR*, pp. 1746–1759.

Scholkopf, B. and A. J. Smola (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA. ISBN 0262194759.

Schwing, A. G., T. Hazan, M. Pollefeys, and R. Urtasun (2012). Efficient Structured Prediction for 3D Indoor Scene Understanding. In *Proc. CVPR*.

Schwing, A. G. and R. Urtasun (2012). Efficient Exact Inference for 3D Indoor Scene Understanding. In *Proc. ECCV*.

Shi, F., X. Zhang, and Y. Liu (2004). A new method of camera pose estimation using 2D-3D corner correspondence. *Pattern Recognition Letters*, **25**(10), pp. 1155 – 1163. ISSN 0167-8655. doi:DOI:10.1016/j.patrec.2004.03.010.

Shi, J. and J. Malik (1997). Normalized cuts and image segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 731–737. IEEE Computer Society Press, Los Alamitos, CA.

Shi, J. and J. Malik (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(9), pp. 888–905.

Shotton, J., A. Blake, and R. Cipolla (2005). Contour-Based Learning for Object Detection. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 - Volume 01*, ICCV '05, pp. 503–510. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-2334-X-01. doi: 10.1109/ICCV.2005.63.

Silberman, N., D. Hoiem, P. Kohli, and R. Fergus (2012). Indoor Segmentation and Support Inference from RGBD Images. In *ECCV (5)*, pp. 746–760.

Sivic, J., B. Russell, A. Efros, A. Zisserman, and B. Freeman (2005). Discovering Objects and Their Location in Images. In *ICCV 2005*.

Song, H. O., S. Zickler, T. Althoff, R. B. Girshick, M. Fritz, C. Geyer, P. F. Felzenszwalb, and T. Darrell (2012). Sparselet Models for Efficient Multiclass Object Detection. In *ECCV (2)*, pp. 802–815.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *Annals of Statistics*, **22**, pp. 1701–1762.

Torralba, A., K. P. Murphy, and W. T. Freeman (2004). Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection. In *CVPR (2)*, pp. 762–769.

Tsai, G. and B. Kuipers (2012). Dynamic visual understanding of the local environment for an indoor navigating robot. In *IROS*, pp. 4695–4701.

Tsai, G., C. Xu, J. Liu, and B. Kuipers (2011). Real-time indoor scene understanding using Bayesian filtering with motion cues. In *ICCV*. ISBN 978-1-4577-1101-5.

Tu, Z., X. Chen, A. L. Yuille, and S.-C. Zhu (2005). Image parsing: Unifying segmentation, detection and recognition. *International Journal of Computer Vision*, **63**(2), pp. 113–140.

Tu, Z. and S.-C. Zhu (2002). Image segmentation by data-driven markov chain monte-carlo. *IEEE Trans. Patt. Analy. Mach. Intell.*, **24**(5), pp. 657–673.

Underwood, S. A. and C. L. Coates (1975). Visual Learning from Multiple Views. *IEEE Trans. Comput.*, **24**(6), pp. 651–661. ISSN 0018-9340. doi:10.1109/T-C.1975.224277.

Wang, H., S. Gould, and D. Koller (2010). Discriminative Learning with Latent Variables for Cluttered Indoor Scene Understanding. ECCV.

Xiang, Y. and S. Savarese (2012). Estimating the aspect layout of object categories. In *CVPR*, pp. 3410–3417.

Xiao, J., B. Russell, and A. Torralba (2012). Localizing 3D cuboids in single-view images. In Bartlett, P., F. Pereira, C. Burges, L. Bottou, and K. Weinberger (eds.) *NIPS*, pp. 755–763.

Yu, C.-N. J. and T. Joachims (2009). Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1169–1176. ACM, New York, NY, USA. ISBN 978-1-60558-516-1. doi:10.1145/1553374.1553523.

Yu, S. X., H. Zhang, and J. Malik (2008). Inferring Spatial Layout from A Single Image via Depth-Ordered Grouping. In *POCV*.

Yuille, A. L., P. W. Hallinan, and D. S. Cohen (1992). Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, **8**(2), pp. 99–111.

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

Zhu, L., Y. Chen, and A. Yuille (2006). Unsupervised learning of a probabilistic grammar for object detection and parsing. In *NIPS*.

Zhu, L., Y. Chen, and A. L. Yuille (2010). Learning a Hierarchical Deformable Template for Rapid Deformable Object Parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(6), pp. 1029–1043.

Zhu, S. and D. Mumford (2006). A Stochastic Grammar of Images. *Foundations and Trends in Computer Graphics and Vision*, **2**(4), pp. 259–362.

Zhu, S.-C., R. Zhang, and Z. Tu (2000). Integrating topdown/bottom-up for object recognition by data driven Markov chain Monte Carlo. In *IEEE Computer Vision and Pattern Recognition*.