Preprint of a paper to appear in Advanced topics in Computer Vision" edited by Giovanni Maria Farinella, Sebastiano Battiato and Roberto Cipolla, Springer, Advances in Computer Vision and Pattern Recognition Series, 2013.

Top-down Bayesian inference of indoor scenes

Luca Del Pero and Kobus Barnard

Abstract The task of inferring the 3D layout of indoor scenes from images has seen many recent advancements. Understanding the basic 3D geometry of these environments is important for higher level applications, such as object recognition and robot navigation. In this chapter, we present our Bayesian generative model for understanding indoor environments. We model the 3D geometry of a room and the objects within it with non overlapping 3D boxes, which provide approximations for both the room boundary and objects like tables and beds. We separately model the imaging process (camera parameters), and an image likelihood, thus providing a complete, generative statistical model for image data. A key feature of this work is using prior information and constraints on the 3D geometry of the scene elements, which addresses ambiguities in the imaging process in a top-down fashion. We also describe and relate this work to other state-of-the-art approaches, and discuss techniques that have become standard in this field, such as estimating the camera pose from a triplet of vanishing points.

1 Introduction

There has been much recent interest in estimating the 3D layout of indoor scenes from monocular images [2, 8, 9, 10, 14, 15, 19, 23], as this provides crucial geometric context for higher level tasks, such as object recognition and prediction of human activities. Consider for example Fig. 1. From just a single image, human observers can infer the 3D structure of the room, even when many cues are hidden. For example, we can estimate the boundary between the floor and the walls, even

Luca Del Pero
University of Arizona, e-mail: delpero@email.arizona.edu

Kobus Barnard
University of Arizona, e-mail: kobus@sista.arizona.edu

Fig. 1 Inferring the 3D layout of a room from an image is challenging due to occlusions and clutter. Here, the boundary between floor and walls is occluded by the furniture. We show on the right the correct position of the 3D box approximating the room

if it is mostly occluded by furniture. This kind of geometric understanding informs interaction with the environment. For example, one could infer that the bed provides a surface for sitting, or that a possible path to the couch goes around the bed.

A number of computational methods for recovering the 3D scene layout have been recently developed based on modeling rooms with simple geometric primitives, such as sets of orthogonal surfaces [14], or 3D boxes [8, 15]. Despite the coarse geometric approximation, these approaches have enabled several interesting applications. For example, Gupta et al. [7] used extracted 3D information to identify locations where people can sit or lie, while Hedau et al. [9] showed how knowledge of the 3D environment helps detecting pieces of furniture, such as beds. Other notable applications include inserting realistic computer graphics objects into indoor images [13], and robot navigation [21].

This chapter extends our previous work on indoor scene understanding [3, 4]. Specifically, we provide additional details on the model and the inference discussed in these papers. New contributions include making the inference multi-threaded by allowing threads to exchange information (§3.4), a new method to make the inference more robust (§3.1.4), and a comprehensive evaluation of all the techniques proposed (§4).

1.1 Background

In the domain of inferring 3D geometry from indoor images, it is common to parametrize the scene layout as a 3D box [8, 9, 10, 15, 19, 23]. This is often referred to as *room box*, as it coarsely models the space of a room as if it were empty. An example is shown in Fig. 1, where the estimated room box is shown on the right. Inferring the room box from an image entails determining the 3D position and orientation of the floor, ceiling and walls that typically define an indoor environment. This process presents two main challenges. First, recovering the 3D box requires inferring the perspective transformation that generated the image (camera estimation). Second, clutter and occlusions are a major source of confusion, since the room box boundaries are often mostly hidden (Fig. 1).

Estimating the camera in indoor scenes has been tackled by using a strong model [8, 9, 10, 15, 19, 23], specifically the Manhattan world assumption [1]. This

states that most surfaces in the scene are aligned with three principal orthogonal directions, which can be estimated by detecting a triplet of orthogonal vanishing points on the image plane [17]. The parameters of the perspective projection can then be recovered analytically from the position of the vanishing points, and this is a well understood problem [25]. In Manhattan world indoor scenes, the three orthogonal directions are invariably those of the room box.

Addressing occlusions and clutter requires reasoning about the objects in the room. Hedau et al. [8] detect clutter in 2D with an appearance based classifier. More recently, Lee et al. [14] showed the benefits of reasoning about objects in 3D. Specifically, they proposed modeling objects as 3D boxes, which provide reasonable bounding approximations of objects typically found in rooms, such as beds and tables, and further exploit the Manhattan assumption by constraining these boxes to be aligned with the room box. Their results showed that joint inference of the room and the 3D objects, which addresses ambiguities caused by clutter and occlusions in a top-down fashion, improves on estimating the room box. Our approach is related to Lee's work [14], but we advocate an even more top-down approach with a more unified representation.

1.2 Overview of our approach

Similarly to Lee et al. [14], our goal is to simultaneously estimate the camera, detect and localize in 3D the floor, ceiling, and walls comprising the room "box," and determine the position of the objects in the room. However, instead of using an object box only to explain occlusions, we want to identify it (e.g., a couch or a bed) as well. This simultaneously 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.

Our goal is to provide a comprehensive parsing of the scene that is globally consistent in 3D, both geometrically and semantically, which is on the lines of the work of Hoiem et al. [12]. This introduces complex constraints, which make the inference process challenging. Examples of constraints include preventing objects from overlapping in 3D, and enforcing that each object is contained in the room box. Previous work [8, 15, 19] relied on the framework of structured prediction for inferring indoor scene models, where inference and modeling are strongly coupled. This makes it harder to adapt the method to handle more complex sources of information, such as conditioning the position and size of objects on their identity.

Bayesian inference is a natural way to handle these complexities, as it allows to separate the modeling and the inference. Specifically, we propose a Bayesian generative model for images of indoor scenes, where we separately model the 3D geometry (room box and objects), the imaging process (camera parameters), and an image likelihood. We assume that the image features are generated statistically from

the projected 3D scene under the camera parameters. We impose further structure by introducing the notion of an object type. The Bayesian framework naturally allows using prior information of the world, which in our case are prior distributions on a box's 3D size and position conditioned on its type and on the room box. For example, beds are typically against a wall, while tables are likely found in the center of the room. In this work, we allow four different types of object boxes, approximating beds, cabinets, couches and tables. We also introduce the notion of *frames*, which are thin boxes "anchored" to a wall, to approximate objects such as doors. We use three types of frames: doors, picture frames, and windows.

Bayesian inference allows us to jointly infer all the elements in our model, without having to commit to partial solutions. For example, previous work [8, 9, 10, 15, 19, 23] rely on initial estimates of the camera parameters, and they cannot recover from mistakes in this step. Similarly, both Hedau et al. [8] and Gupta et al. [7] use an initial estimate of the room box for identifying beds and predicting human activities in the room, respectively. Again, errors in the estimate of the room box cannot be recovered from. In principle, our approach does not have this problem.

In what follows, we detail with our model for the 3D scene geometry and the camera (§2). We then develop priors that distinguish among object boxes based on their position and size (§2.1), and then detail the imaging model (§2.2), including an analysis of the standard image features used in this field. We then describe the Markov chain Monte Carlo sampling method that we use for inference (§3). Important aspects include how to handle constraints during sampling, how to use data-driven methods for efficient sampling, and how to do inference with multiple threads. Finally, we provide extensive evaluation of our approach on two standard datasets (§4).

2 A Bayesian generative model for indoor scenes

We use a Bayesian generative model, where we assume that images are generated by the projection of the 3D scene. We partition model parameters, θ, into scene parameters, s, encoding the 3D geometry, and camera parameters, c, modeling the perspective transformation

$$\theta = (s, c) \quad . \tag{1}$$

We define the posterior distribution as

$$p(\theta|D) \propto p(D|\theta)p(\theta) \quad , \tag{2}$$

where D are features detected on the image plane and $p(\theta)$ is the prior distribution over model parameters.

Scene parameters include the room box and objects in it

$$s = (r, o_1, ..., o_n) \quad , \tag{3}$$

where the number of objects n and their type are not known a priori. We model the room as a right-angled parallelepiped whose floor is parallel to the x-z plane of the world reference frame. It is defined by its 3D center, width, height, length, and rotation angle, γ_r, around an axis parallel to the world y-axis and through the room center (yaw):

$$r = (x_r, y_r, z_r, w_r, h_r, l_r, \gamma_r) \quad . \tag{4}$$

Objects in the room are similarly modeled by blocks, but include an object category, t_i, (e.g., bed, table, door). 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 approximated with thin blocks (frames) attached to a wall. All objects share the same orientation γ_r of the room block, following the Manhattan world assumption. Objects must be entirely inside the room box, and they can not intersect each other.

Objects coordinates are relative to the coordinate frame defined by the room center, and whose axes are aligned with the room walls, which we call "room coordinates". We define a coordinate transformation function $r_{coord}(x,y,z) = (x^r, y^r, z^r)$ for later use, to transform a point defined relatively to the world coordinate system (x,y,z) 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. An advantage of storing objects in room coordinates, is that it allows for efficient computation of intersections among objects, and between an object and the room box.

Each object is "anchored" to a room surface, whose index is stored as a discrete variable (s_i). Furniture objects lie on the floor ($s_i = 4$), implying that, given the object height, the y coordinate of the object center is not a free parameter. Specifically, $y_i = -(y_r/2) + (h_i/2.0)$, where $-(y_r/2)$ is the position of the floor in room coordinates. Similarly, frames are anchored to one of the walls, which analogously constrains their parameters (see Fig. 2). To summarize the object parameters,

$$o_i = (t_i, s_i, x_i, y_i, z_i, w_i, h_i, l_i) \quad . \tag{5}$$

The imaging process is modeled with a standard perspective camera

$$c = (f, \phi, \psi) \quad , \tag{6}$$

where f, ϕ and ψ are, respectively, the focal length, the pitch, and the roll angle. Since absolute positions cannot be determined when reconstructing from single images, we arbitrarily position the camera at the origin of the world coordinate system, pointing down the z-axis. The extrinsic camera rotations, specified by three degrees of freedom, are fully determined by ϕ, ψ and the yaw γ_r of the room (see Fig. 3). Pitch and roll are constrained within ranges of plausible values for indoor scenes ($\phi \in [-60^\circ, 60^\circ]$), $\psi \in [-10^\circ, 10^\circ]$), while the focal length has to be positive. We further assume unary aspect ratio, no skew, and that the principal point coincides with the image center.

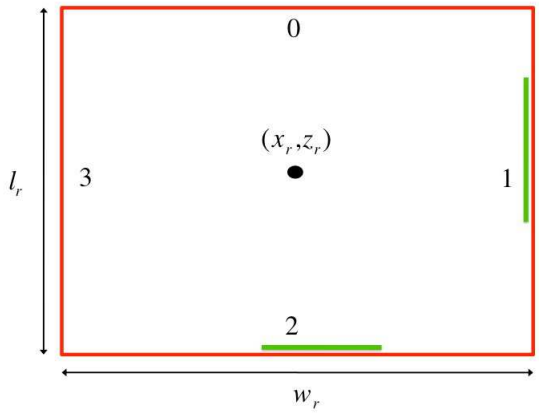

Fig. 2 The coordinates of the objects are stored in a coordinate frame relative to the center of the room box, and whose axes are aligned with the room walls. Here, two frames are "anchored" to a wall of the room box, seen from above. Walls are numbered from 0 to 3, and this index is stored in variable s_i, which imposes constraints on the parameters in Eq. 5. For example, for the frame on the right, we have $s_i = 1$ and $x_i = (w_r/2)$. Further, since frames are approximated with thin blocks, we set $w_i = \varepsilon$, with $\varepsilon = 0.01$ units. For the frame at the bottom, $s_i = 2$, $z_i = (l_r/2)$, and $l_i = \varepsilon$

2.1 Model priors

Priors on scene elements improve global scene understanding by helping resolve ambiguity during inference, and also support identifying objects based on geometry cues, such as size and location, alone. Previous work has used blocks in the scene to explain occlusions [15] and to infer what regions of the 3D space are occupied by generic objects [10]. In this work, we assign a semantic label to our blocks (e.g. couch, door, etc.), with each label corresponding to specific prior probabilities on object size and position in 3D. Because we are reconstructing from a single image, we have one overall scale ambiguity, and thus priors on object "size" and position are defined relatively to the room box.

As an example of information captured by the priors, wardrobe cabinets are tall and narrow and typically against a wall, while tables are usually shorter, wider, and in the middle of the room. Similarly, a door is quite tall and touches the floor, while picture frames are much shorter and are typically found in the higher half of a wall.

Assuming independence, the overall prior for the model parameters is given by

$$p(\theta) = \pi(r)\pi(c)\prod_{i=1}^{n}\pi(o_i) \quad , \tag{7}$$

where $\pi(r)$ is the prior on the room box, $\pi(c)$ is the prior on camera parameters, and $\pi(o_i)$ is the prior for one of n objects in the room. We now describe each of these components and how we set their parameters from training data.

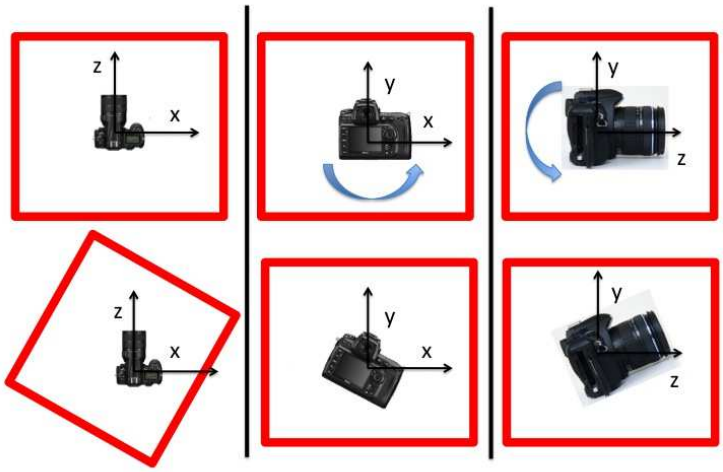

Fig. 3 Camera parameters. The extrinsic parameters define the position and orientation of the camera with respect to the world reference frame. Since absolute sizes and position cannot be determined, we arbitrarily position the camera at the origin of the world coordinate system, pointing down the negative z-axis. The room box 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

2.1.1 Prior on room box

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

$$r_{r1} = \frac{max(w_r, l_r)}{min(w_r, l_r)} \quad . \tag{8}$$

We use this formulation since we do not know in advance which dimension is the largest. We also put a prior on the ratio of the long dimension to the height

$$r_{r2} = \frac{max(w_r, l_r)}{h_r} \quad . \tag{9}$$

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

$$\pi(r) = \mathcal{N}(r_{r1}, \mu_{r1}, \sigma_{r1}) \mathcal{N}(r_{r2}, \mu_{r2}, \sigma_{r2}) \quad . \tag{10}$$

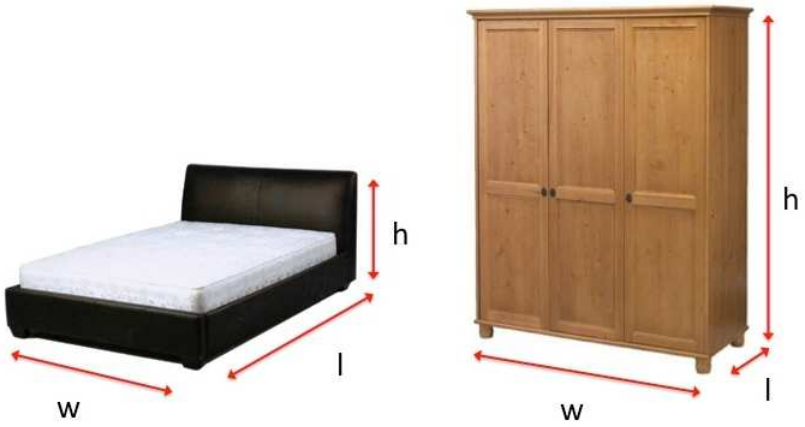

Fig. 4 Distinguishing objecs using their size. The ratio between the height of an object and the largest between its width and length varies considerably between beds and cabinets. In this example, also the ratio between width and length is quite discriminative

2.1.2 Prior on camera parameters

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

$$\pi(c) = \mathcal{N}(r_{ch}, \mu_{ch}, \sigma_{ch}) \quad . \tag{11}$$

2.1.3 Prior on objects.

Several categories of furniture and frames have a very distinctive size (Fig. 4). Here we introduce a general formulation for a prior for a specific object of category $t_i = \tau$ that exploits this intuition. 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 the following quantities

- ratio between the object height and its largest other dimension $r_{i1} = h_i / max(w_i, l_i)$ (Fig. 4). This helps distinguish between categories that are taller than they are wide, such as wardrobes, and short and wide objects, such as beds.
- ratio between the object long and short dimensions, $r_{i2} = max(w_i, l_i)/min(w_i, l_i)$ (Fig. 4). Again, we use this formulation since we do not know in advance which dimension is the largest. This quantity discriminates between furniture with a roughly square base, and furniture with a rectangular base. This component is not used for frames, since one of their width or length is always negligible.

Fig. 5 Learning the ratio between room height and object height. This quantity is very informative, 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 this domain the vanishing point for vertical segments is usually at infinity, and this method provides a reasonable approximation

- 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. 5).
- 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.

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. The first three quantities follow a normal distribution

$$\pi_j(o_i|t_i = \tau) = \mathcal{N}(r_{ij}; \mu_{\tau_j}, \sigma_{\tau j}) \quad , \tag{12}$$

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 as follows

$$\pi(o_i) = \pi(d_i) \prod_{j=1}^{3} \pi_j(o_i) \quad . \tag{13}$$

2.1.4 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 [1] for beds, couches, cabinets and tables, and the Home Depot catalog [2] for windows, doors and picture frames.

Setting the parameters for the remaining two 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. 5. 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 [8], omitting images where we could not 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 parameters [4].

2.2 The image model

Our image model is similar to the one used by Schlecht and Barnard [18]. Specifically, we assume that image features $D = (f_1, ..., f_s)$ are generated by the projection of the 3D scene under the given camera. We use three feature types that proved useful in this domain, specifically edges [3, 4, 14], orientation surfaces [14, 15], and geometric context [8, 15].

2.2.1 Image edges

We assume image edges to be generated by the blocks in the scene. We measure the quality of a fit by comparing the set of edges E_d detected on the image plane to the set of edges E_θ generated by projecting the model. As in Schecht et al. [18], we define a likelihood function $p(E_d|E_\theta)$, which we specify using the following intuitions:

- An edge point $e_{dj} \in E_d$ detected in the image plane should be matched to an edge point $e_{\theta k} \in E_\theta$ generated by the model. If the match is good the two points should be very close to each other, and the difference in orientation between the two edges should be minimal. We use $p(e_{dj}|e_{\theta k}) = \mathcal{N}(d_{jk}, 0, \sigma_d)\mathcal{N}(\phi_{jk}, 0, \sigma_\phi)$, where d_{jk} is the distance between the points, and ϕ_{jk} the difference in orientation between the edges.
- We penalize a detected edge point that is not matched to any model edge (noise). We define p_n as the probability of such an event occurring.
- We explain points in E_θ not matched to any point in E_d as missing detections, and define probabilities p_{hmiss} and p_{smiss}. The former is used for "hard" edges

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

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

arising from occlusion boundaries, such as the edges that belong to the silhouette of an object. The latter is used for "soft" edges that are less likely to be found by the edge detector, such as the room edges and non-silhouette edges from objects. It is less likely that a detector will miss a "hard" edge compared with a "soft" edge. Note that one of the advantages of using a full 3D model, is that we can determine whether hypothesized edge points in E_θ are hard or soft.

We then have

$$\tilde{p}(E_d|E_\theta) = p_n^{N_n} p_{smiss}^{N_{smiss}} p_{hmiss}^{N_{hmiss}} \prod_{(j,k)\in matches} p(e_{dj}|e_{\theta k}) \quad , \tag{14}$$

where N_n is the number of edge points labeled as noise, and N_{smiss} (N_{hmiss}) the number of missed soft (hard) edges. We match points in a greedy fashion by finding the closest point e_θ to a data edge e_d along the edge gradient, provided that this distance is smaller than 40 pixels, and the difference in orientation is less than 0.7 radian. 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

$$p(E_d|E_\theta) \approx \tilde{p}(E_d|E_\theta)^{(N_{hmiss}+N_{smiss}+N_n+N_{matches})^{-1}} \quad . \tag{15}$$

2.2.2 Orientation surfaces

Based on the Manhattan world assumption, most pixels in the scene are generated by a plane aligned with one of three orthogonal directions, and we can estimate which one using the approach by Lee et al. [14]. We compare the pixel orientation O_d detected from the image plane with the orientation surfaces O_θ generated by projecting our model. We 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.

2.2.3 Geometric context

Following Hedau et al. [8], we also consider geometric context labels, which estimate the geometric class of each pixel, choosing between object, floor, ceiling, left, middle and right wall. This is done using a probabilistic classifier trained on a mixture of color and oriented gradient features [11]. We use the code and the pre-trained classifier available online [11]. For each pixel p_k, this provides a probability distribution $gc_k = [gc_{k1}, ..., gc_{k6}]$ over the six classes. 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

$$p(GC|\theta) = \frac{\sum_{p_k \in I} p(gc_k|p_k)}{size(I)} \quad , \tag{16}$$

where we average the contributions of all image pixels. 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.

2.2.4 Combining the three features

Assuming independence among the features, we define our likelihood function

$$p(D|\theta) = p(E_d|E_m)p(O_d|O_m)^{\alpha}p(GC|\theta)^{\beta} \quad , \tag{17}$$

where α and β weigh the importance of the orientation and geometric context likelihoods, relative to the edge likelihood. We set $\beta = 12$ and $\alpha = 6$ by running our algorithm on the training portion of the Hedau dataset [8]. Here we used a coarse grid search over α and β, with a step of 2, using the room box layout error (defined in §4) as an objective function.

Later, in the results (Fig. 18) we illustrate how these three features work together. Errors in the edge detection process can be fixed using orientation surfaces and geometric context, and vice versa. Using edges also helps improving the camera fit when starting from a wrong estimate of the vanishing points, which are detected at the beginning and used to initialize the camera parameters. In fact, since the algorithms for computing orientation maps and geometric context depend on the initial vanishing point estimation, this feature is compromised by this initial error, whereas edges are not.

3 Inference

We use Markov chain Monte Carlo (MCMC) sampling to search the parameter space, defined by camera and room box parameters, the unknown number of objects, their type, and the parameters of each object. 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 Metropolis-Hastings (MH) [5, 6]. To change the continuous parameters, which comprise the room box, the camera, and size and position of each object, we use Hamiltonian dynamics sampling [16]. The proposals from these two samplings strategies are often referred to as "jump" and "diffusion" moves [22].

The 3D structure of the model introduces several constraints and dependencies among parameters, which must be carefully taken into account during inference. Specifically, objects cannot overlap in 3D, and they have to be entirely inside the room box. Further, the camera must be inside the room box, and not within the volume occupied by any of the objects. Enforcing these constraints during inference can introduce several ambiguities, as illustrated in Fig. 6(g), using a birdview of

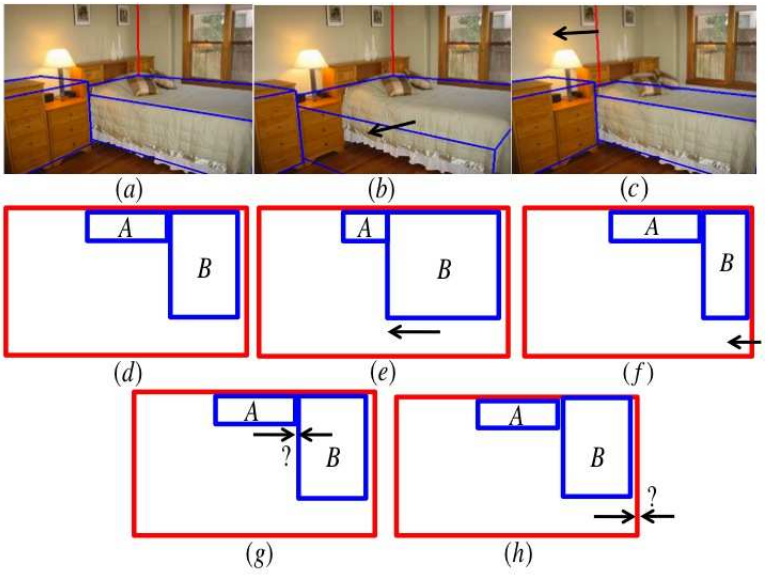

Fig. 6 Sampling over subsets of parameters handles containment constraints (top two rows) and avoids ambiguities (bottom row). Consider sampling over the parameters of a single object at a time. As the object expands, other objects have to shrink to avoid overlap. For example, **A** has to shrink to 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**) and (**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 would create similar conflicts (**h**)

the room box. Consider sampling over objects A and B simultaneously. This could result in a conflict where object B tries to expand towards the left, and A towards the right, as illustrated by the arrows. A similar situation can happen when the room box is trying to shrink and object B is trying to expand (h).

To avoid these ambiguities, we use three different types of continuous moves over subsets of the scene parameters, and define rules for adapting the model so that no constraint is violated. In the first one, 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. 6, a-b). The second move samples over the parameters of the room box only, and adapt the objects to respect the constraints. As the room box becomes smaller, we shrink objects in case they end outside the room (Fig. 6, c). Third, we jointly sample over camera and room box parameters, by enforcing constraints as in the previous move. Further, for all the three moves just discussed, we enforce that the camera is inside the room box, and 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.

Diffusion moves and Jump moves are alternated throughout inference. We summarize the entire process here, and describe each component in detail in the following Sections:

1. Initialize the parameters of the camera and of the room box
2. Repeat K times
   a. Generate a new sample with one of the following moves, chosen randomly
      - Jump 1: add an object to the scene (furniture of frame)
      - Jump 2: remove an object from the scene
      - Jump 3: pick a random object, and change 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).
3. Return the sample with the highest posterior

We rely on a multi-threaded strategy to efficiently explore more of the space on modern multi-core workstations. Each thread executes the procedure above, and, at the end, threads are allowed to exchange information, as some of the objects might be found by a thread and missed by the others, and vice versa (§3.4). This exchanging procedure is followed by additional sampling, and we output the best sample found. In our experiments we used 20 threads, and the whole inference process takes on average ten minutes per image.

In what follows, we first detail with the diffusion moves in Sect. §3.1, where we develop concepts needed to better discuss the Jump moves §3.2, and how we initialize the room box and the camera §3.3. Last, we explain how to exchange objects among threads §3.4.

3.1 Diffusion moves

As illustrated in Fig. 6, we sample over subsets of the scene parameters and specify rules to make the model comply with all constraints. We use Hamiltonian dynamics sampling, which were used by Schlecht et al. [18] for learning the continuous parameters of geometric models for furniture with a similar parametrization.

We use Neal's formulation of Hamiltonian dynamics [16] to sample over phase space, where the energy function is defined in terms of the joint distribution of the parameters and the image data $p(\theta, D)$

$$E(\theta) = -log(p(D|\theta)) - log(p(\theta)) \quad . \tag{18}$$

We follow the dynamics using leapfrog discretization [16], and compute the derivative of the potential energy with numerical approximation, which is the current bottleneck for computation.

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

When sampling 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, and further adjust the position of the other objects (§3.1.2).

To reduce the sampling time, we designed efficient ways for sampling object parameters. For example, consider the window in Fig. 7 (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. 7 (b). This technique is effective in our framework, since typically one edge of the object is correctly "latched" to an image edge, given our proposal mechanism from image corners discussed in §3.2.1, and we thus 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. 7 (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.

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

When sampling the parameters of the room box r, we adapt all the objects so that no constraints are violated. When we detect that an object is partly outside the box, we shrink it, and delete it if it is completely outside. As for the object parameters, we sample along one direction of the room by keeping one edge "latched" (Fig. 7, third row). Six sampling directions are available (fourth row), and we use 20 leapfrog steps per direction. At each leapfrog step we enforce the containment constraint, and we further apply a transformation to the objects in the room, to preserve their projection on the image plane. Since room objects parameters are relative to the room box coordinate system, changing the 3D position of the room box would not preserve the projection of the room objects. An example is shown in Fig. 7 (g-h), where a small change in the room center causes the object to move as well.

We address this as follows. We define the room 3D center $(r_x + \delta_{rx}, r_y + \delta_{ry}, r_z + \delta_{rz})$ at leapfrog step t, where all the parameters are relative to the world coordinate system, and (r_x, r_y, r_z) is the room center at leapfrog step $(t-1)$. Let us also define (x_i, y_i, z_i) as the center of object o_i at $(t-1)$, this time in room coordinates. For each object o_i, we compute the position of its center at step t as $(x_i + \delta^r_{rx}, y_i + \delta^r_{ry}, z_i + \delta^r_{rz})$, where $(\delta^r_{rx}, \delta^r_{ry}, \delta^r_{rz}) = r_{coord}(\delta_{rx}, \delta_{ry}, \delta_{rz})$. The result of applying this transformation

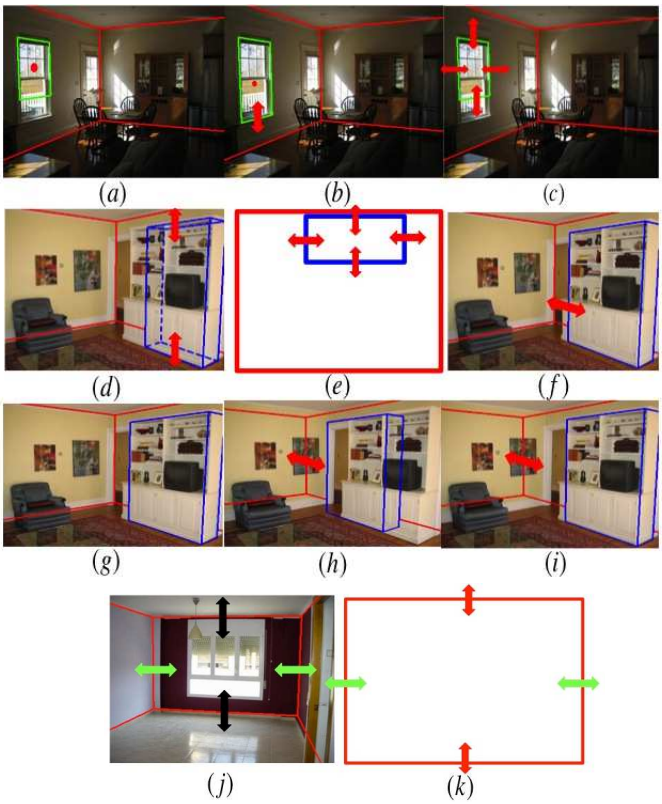

Fig. 7 Efficient strategies for sampling over continuous parameters. The upper edge of the window in (**a**) is positioned correctly, and the correct fit for the window can be obtained efficiently by dragging down the lower edge while keeping the upper edge fixed (**b**). We designed moves that achieve 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. The same strategy is also used when sampling over the room box (third row), where we have six possible directions, illustrated in (**j**) and (**k**) (two of those are shown in both, denoted by the green arrows). Since object coordinates are relative to the room box, changes in the latter have the undesirable effect of changing the projection of objects already in the room. For example, we only changed the 3D center of the room box from (**g**) to (**h**), and 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 (**i**)

is shown in Fig. 7 (i), where the room box has changed, but the object block kept its position on the image plane.

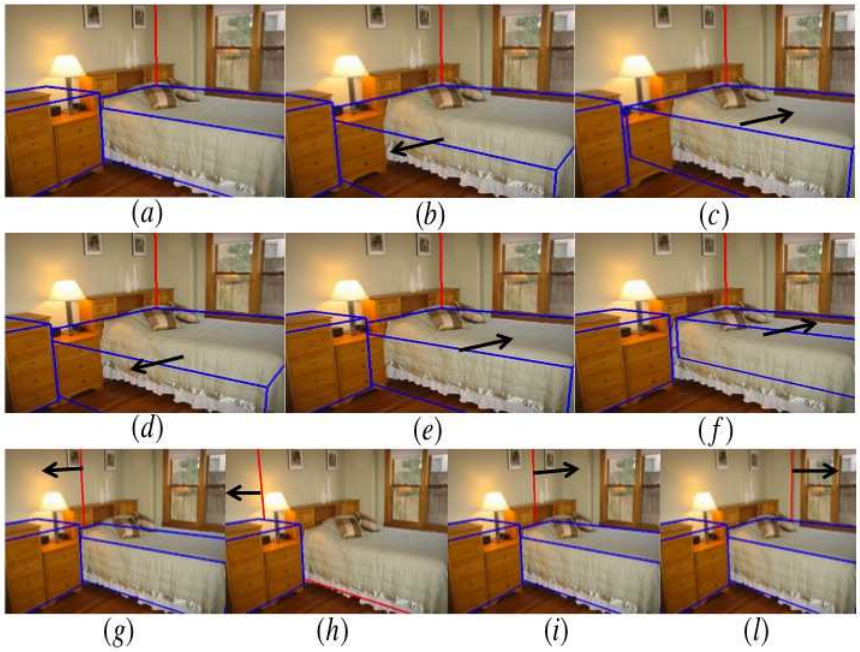

Fig. 8 Sensible sampling with interacting boxes. Suppose sampling over the block projected over the bed (**a**), along the direction denoted by the arrow (**b**) for N leapfrog steps. For a few steps, the bed tries to expand, and the containment constraints force the block on the left to shrink (**b**). 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 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**)

3.1.3 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 iterations. At each step, we enforce the same constraints as in the case where we sample over the room box only.

3.1.4 Sensible dynamics with multiple boxes

The constraints in the model give rise to an additional problem illustrated in Fig. 8. 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 show 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). The procedure implementing this strategy for the sampling of a set of continuous parameters is as follows (θ_{in} is the initial sample, and θ_{out} is the sample after N leapfrog steps).

1. Set $\theta_0 = \theta_{in}$
2. For every step $i = 1, ..., N$
   a. Compute θ_i from θ_{i-1} by following the Hamiltonian dynamics.
   b. Whenever the posterior needs to be evaluated, compute it on a copy of θ_{i-1} where we apply the containment constraints. Never apply the constraints on either θ_i or θ_{i-1}
3. Set $\theta_{out} = \theta_N$
4. Apply the containment constraints on θ_{out}

This procedure implements the desired behavior illustrated in Fig. 8, by keeping track of the initial positions and sizes of all objects in the model, and applying the containment constraints only when the posterior needs to be evaluated. Finally, these constraints are applied to the last sample, which will be the starting point for the next sampling move.

3.2 Jump moves

One of the main challenges in the inference process is designing jump moves to efficiently add objects to the scene. Since the sampling space is so large, naive jump proposals, such as samples from a prior distribution, are unlikely to be accepted, and this leads to unacceptably long running times. To face this challenge, we introduced a data-driven strategy [22, 26] to condition the sampling on the data, by proposing samples from image evidence in a bottom-up fashion. Further, we exploit the Manhattan world assumption [1] that most surfaces in the world are aligned with one of three orthogonal directions, as this provides strong constraints on the parameter space.

In Fig. 9, we illustrate 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*, 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 propose furniture objects (Fig. 9, 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 [17]. 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 (§3.2.1), how to detect image corners (§3.2.2), and how to use them to propose both objects in the scene (§3.2.3) and room box candidates (§3.3).

3.2.1 Vanishing point estimation

We first detect straight edges and fit line segments to them using the straight connected edge detector by Hoiem et al [11]. Then, we detect vanishing points following the RANSAC procedure proposed by Lee et al [14]. 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. We then estimate the intrinsic camera matrix K from the Choleski decomposition of the absolute conic matrix [25], 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).

For each valid triplet $V_t(v_1, v_2, v_3)$, we compute the objective function $f(V_t)$ as follows. First, we compute the angular distance $\alpha(s_i, v_k)$ between each line segment s_i and each vanishing point v_k (Fig. 10). A segment s_i is labeled as an outlier if $\min_{k=1,2,3} \alpha(s_i, v_k) > 0.06$. If n is the total number of outliers, we have

$$f(V_t) = \left(\sum_{i=1}^{n} \frac{\min_{k=1,2,3} \alpha(s_i, v_k)}{l_i} \right) \Big/ n \tag{19}$$

where n is the number of inlier line segments in the RANSAC procedure. l_i is the length of segment s_i, and is used to assign 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 first only triplets supported by a large number of segments, and, if none is available, we allow a larger number of outliers.

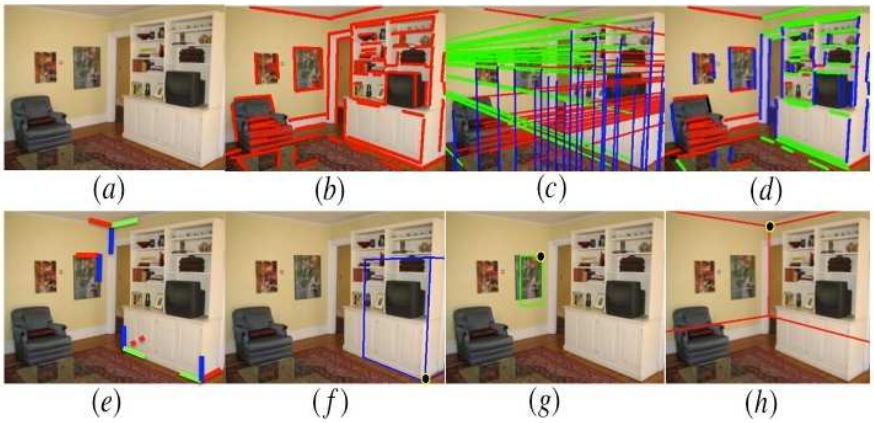

Fig. 9 Detecting vanishing points (top) to find orthogonal corners (bottom). Most surfaces in indoor scenes are aligned with three principal orthogonal directions. This defines a triplet of orthogonal vanishing points in the image plane, which we find in a RANSAC fashion (**c**) from straight detected 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**)

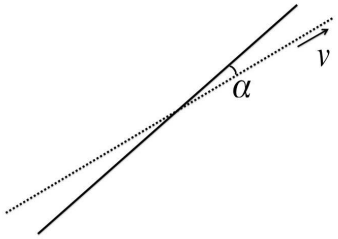

Fig. 10 The angular distance α between a segment s and its vanishing point v. α measures the angle between s and the line through the mid point of s and v

3.2.2 Image corners

Given the estimated triplet of vanishing points, each line segment is assigned to the vanishing point minimizing the angular distance α [8], 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. 9 (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 such segments, which we call image corner, is thus likely to be generated by a 3D orthogonal corner, such as

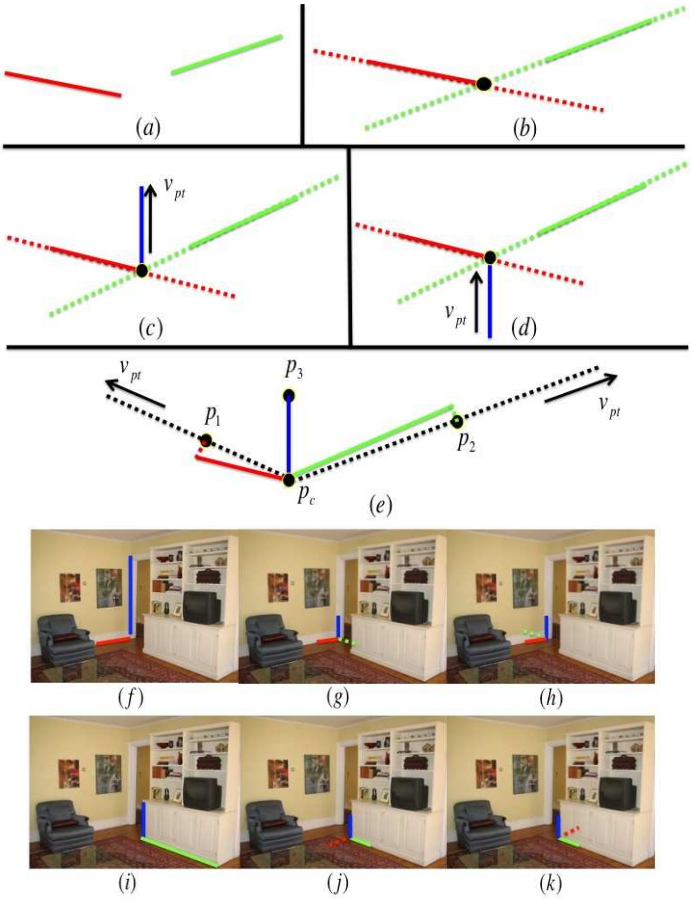

Fig. 11 Creating orthogonal corners from line segments. Given a pair of segments converging to different vanishing points (**a**), we first find their intersection (**b**). We do not create corners if the distance between the intersection and each segment is larger than 30 pixels. We position the third 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 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 line through the corner center and the vanishing points (**e**), which are guaranteed to be orthogonal. The final corner is centered in p_c, and defined by segments p_1p_c, p_2p_c and p_3p_c. Notice that p_3p_c was hallucinated by using a vanishing point, and does not need to be "rectified". In the last two rows, we show examples of corners created from pairs of segments

the corners of the cabinet or the inner corner of the room shown in Fig. 9 (e). Given an image corner and an estimate for the camera pose, we can propose 3D objects in likely positions. However, this requires knowing the three segments forming the im-

age corner, but typically only two are visible due to occlusions, like the bottom left corner of the cabinet in Fig. 9 (e). We address this problem by "hallucinating" the third segment when it is not visible. In the case of the bottom left cabinet corner, we consider the intersection between the two visible segments, which converge to two different vanishing point, and hypothesize that the third segment converges to the remaining vanishing point in the triplet. This is discussed in detail in Fig. 11, where we also explain how to use the Manhattan constraints to ensure the orthogonality of the corner. Last, we stress that, since the main purpose of corners is to propose the correct position and orientation of objects in 3D, and not their size, the length of corner segments in 2D does not provide any valuable information, as long as their direction is correct.

3.2.3 Adding objects in the scene using image corners

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 [5].

First, the image corner is used to estimate the position of the furniture object in 3D (Fig. 12 and 13). The proposal is conditioned on the current estimate of both camera pose and room box. Second, we randomly select the type of the object, such as bed, and use the priors for that category to propose the size of the object. Third, to further increase the acceptance ratio of jump moves, we use a delayed acceptance mechanism. In fact, corners help propose objects at the right position, but the size sampled from the prior is often inaccurate. 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.

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

1. Randomly choose an image corner, and determine whether it is pointing up or down (Fig. 12)
2. Determine the object category τ by randomly choosing from the four available classes (bed, cabinet, couch, table), where 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. 12, otherwise find the position of the 3D corner as explained in Fig. 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 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.
10. Briefly sample over the object continuous parameters using Diffusion move 3 (delayed acceptance)
11. 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. Notice that the move adding an object chooses whether to add a furniture object or a frame with 0.5 probability.

1. Randomly choose an image corner
2. Determine the object category τ by randomly choosing from the three available classes (door, picture frame, window), where 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. 14.
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. When w_i is negligible, set $l_i = \frac{h_i}{r_{i1}}$
6. Shrink any frame in the scene colliding with the proposed one
7. If the frame does not fit on the wall, expand it.
8. Briefly sample over the object continuous parameters using Diffusion move 3 (delayed acceptance)
9. Accept or reject the proposed frame by consulting the MH acceptance formula

3.2.4 Other jump moves

The remaining two jump moves respectively remove an object from the room, or change the type of one of the objects. The former simply deletes a randomly selected object, and the proposed change is accepted or rejected using the MH acceptance formula. Instead, changing the type of an object involves only changes in the prior. First, an object is randomly selected, and we propose changing its current type τ to a different category, also randomly selected. For example, changing the label of an object from “bed” to “couch” results in using the distribution on size and position for couches when evaluating the object prior probability, while before the prior for

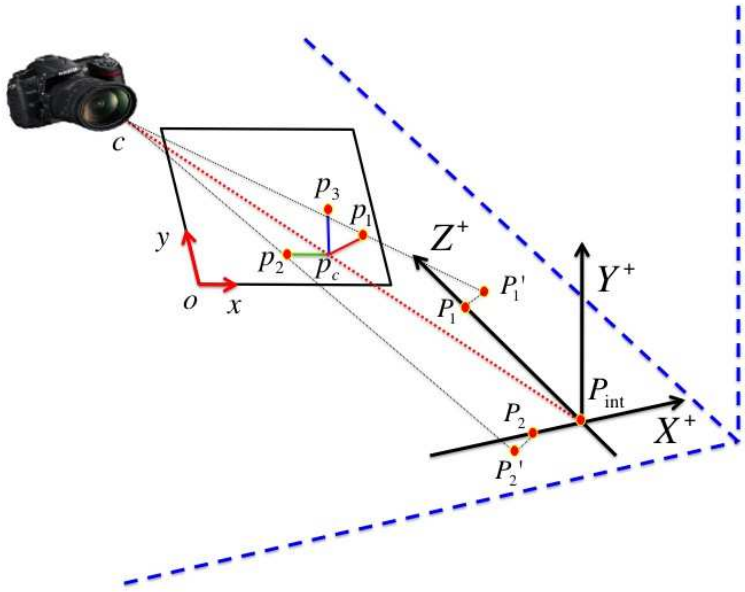

Fig. 12 Finding the 3D corner of a furniture object from an image corner, conditioned on the current room box. $p_c = (p_{cx}, p_{cy})$ denotes the position of the corner on the image plane, and $P_c = (p_{cx}, p_{cy}, -f)$ is the corner in the 3D camera reference frame, where f is the focal length. We first consider the segment $p_3 p_c$: if $p_{3y} > p_{cy}$ in the image reference frame, the image corner is pointing upwards, and we assume that it was generated by an object corner on the room floor, delimited by the dashed lines (for the complimentary case of a downwards corner, see Fig. 13). 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), which is aligned with the room walls and centered in P_{int}, the XZ plane coinciding with the room floor. We know the 3D corner will expand along the positive Y axis, and we 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 $p_1 c$ 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

beds was used. Again, the proposed change is evaluated using the MH acceptance formula. Last, we do not propose changing furniture objects into frames or vice versa. This means that, for example, a couch can only be changed into a table, a bed or a cabinet, while a door only into a picture frame or a window.

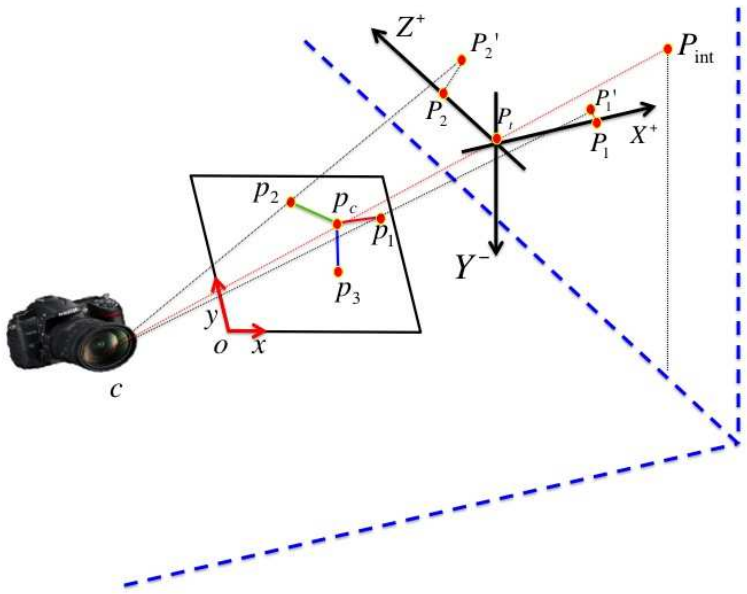

Fig. 13 Finding the 3D corner of a furniture object from a downwards image corner. We assume that a downwards image corner is generated by a corner on the top of a 3D object (For the complimentary case of an upwards corner, see Fig. 12). We cast a ray through the camera center and the corner position 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 choose 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. 12, 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

3.3 Initializing the room box and the camera parameters

We initialize the parameters of the camera and of the room box by proposing candidates from the orthogonal corners detected on the image plane. Each corner is used to generate a candidate, and we use the N room box candidates with the highest posterior to initialize the N threads used for inference. While we do not solely commit to these proposals, since the inference process will modify room box and camera parameters, we found that a good initial estimate of the room box parameters makes the inference more efficient. We now discuss how a corner is used to generate a candidate.

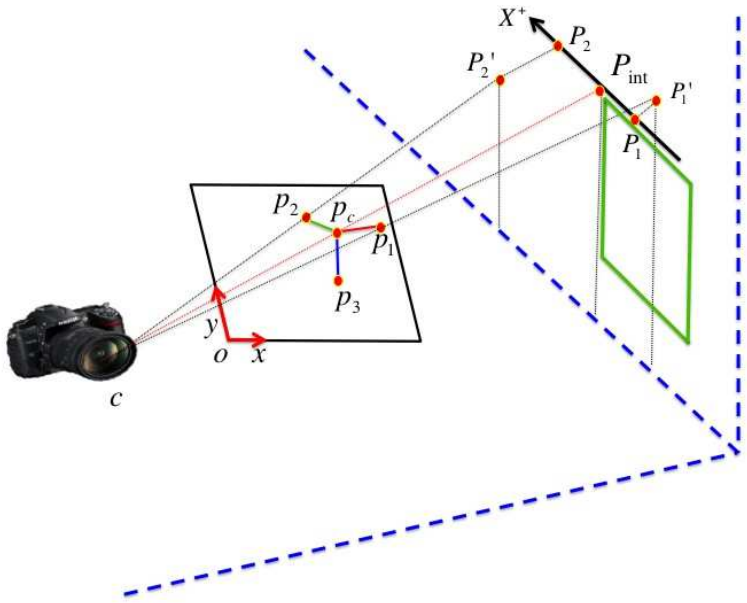

Fig. 14 Finding the 3D corner of a frame from an image corner. We cast a ray between the camera center c and P_c, and find the intersection P_{int} with the closest wall, which determines the position of the corner in 3D. We know that the frame will expand downwards on the wall, as the corner on the 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', and 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

Shi et al. [20] showed how to estimate the camera pose from the projection of an orthogonal corner and the known focal length. We follow their procedure to estimate the pitch ϕ, the roll ψ of the camera and the yaw γ_r of the room, which in our framework define the camera pose. Notice that the focal length is available from the estimated triplet of orthogonal vanishing points.

The method by Shi et al. [20] also recovers the 3D directions of the lines forming the corner. However, we still need to determine the 3D position of the corner, which can lie anywhere on the line defined by the camera center and the corner position on the image plane, as illustrated in Fig. 15. 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.

This still leaves the room dimensions (h_r, w_r, l_r) as free parameters. Since absolute sizes cannot be used when reconstructing from a single image, we set the room height such that the ratio between the height of the camera from the floor h_c and h_r takes plausible values (Fig. 15). Specifically, we sample uniformly from the interval

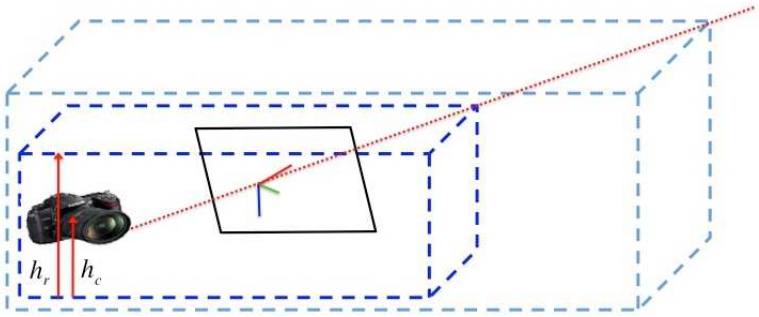

Fig. 15 Finding the position of a 3D room corner from an image corner. We first cast a ray between the corner in the image and the camera center. We then position the 3D room corner along this line. Different positions generate different room boxes, as illustrated by the two dashed examples above. Once 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 the range of plausible values (see text). Further, we have to enforce that the room box is big enough to contain the camera

$[\mu_{ch} - 2\sigma_{ch}, \mu_{ch} + 2\sigma_{ch}]$, with step $\frac{\sigma_{ch}}{5.0}$. For each different height, we set w_r and l_r such that $\frac{w_r}{h_r} = \mu_{r2}$ and $\frac{l_r}{h_r} = \mu_{r2}$, and if the room box is not big enough to contain the camera, we expand it. We then briefly sample over room box and camera parameters by alternating Diffusion moves 1 and 2, and keep the sample with the highest posterior.

3.4 Exchanging information among threads

At the end of the inference process, each thread outputs the sample with the highest posterior. In most cases, we found that some objects are found only by some of the threads, as illustrated in Fig. 16. 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 is defined relatively to the room, we need an exchanging mechanism taking into account that different threads have different estimates of the room box. 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 (Fig. 17).

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 one that provides the best 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

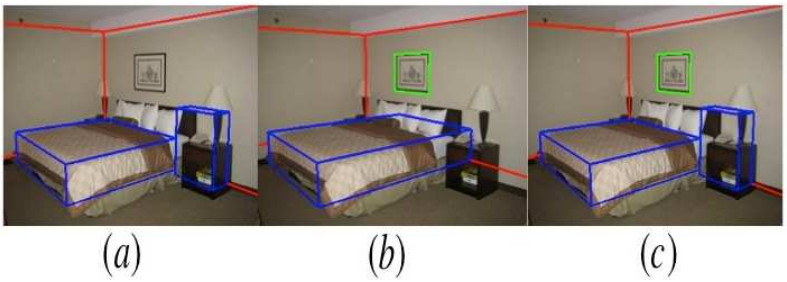

Fig. 16 The result (**c**) of exchanging objects between the samples found by two different threads (**a** and **b**).

in practice. The complete procedure for exchanging objects among N threads is as follows:

1. For each thread i, save in θ_i^0 the best sample found by thread i
2. For each θ_i^0, and for $k = 1, \ldots, K$
   a. Set $\theta_i^k = \theta_i^{k-1}$. Compute the posterior p_i^k of θ_i^k. Get all the objects found by other threads $O = \cup_{j=1}^{N} O_j$ with $j \neq i$, where $O_j = (o_{j1}, \ldots, o_{jn})$ is the list of objects in θ_j^0.
   b. for each object o_w in O, create $\theta_i^{k\_w}$ by adding o_w to θ_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 $\theta_i^K = \theta_i^k$.
3. Return θ_i^K with the largest posterior.

4 Results

We start by evaluating the performances of the various components of our algorithm in terms of the error on the room layout estimation [8, 15, 23], which is a standard measure in this field. This error 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.

In Table 1, we evaluate the impact of the different components of our approach on the test portion of the Hedau dataset [8], consisting of 104 color images. In the left column we show the contribution of the priors in the scenario where the likelihood function uses edges and orientation maps, as in our 2012 CVPR paper [4]. We see that the room prior reduces the layout error, and so do priors on objects. This

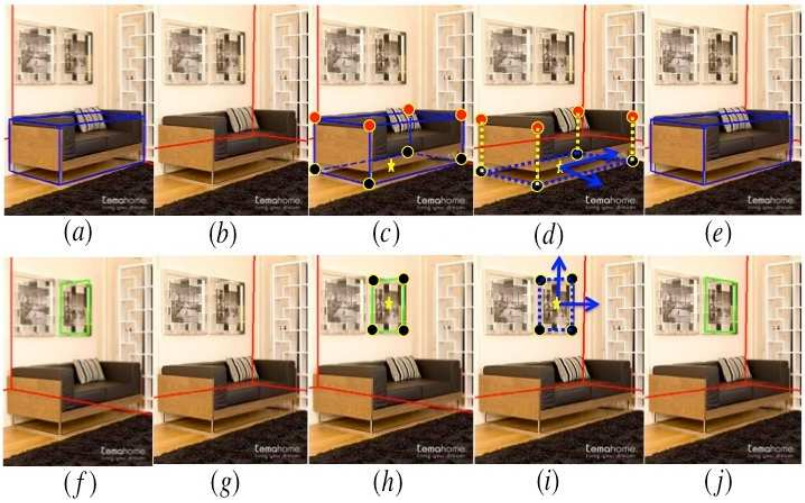

Fig. 17 Exchanging furniture objects (top row) and frames (bottom row) between two samples. Consider adding the object in (**a**) to (**b**): the camera parameters and the room boxes are different, and this is a problem since object coordinates are relative to the room reference frame. We address this by making sure that the projection of the transferred object (**e**) matches that of the original one (**a**). First, we use the original camera to project the object corners that touch the floor, shown in black in (**c**). The star indicates the projection of the center of the object "floor". We then cast a ray between the destination camera and the star, and intersect it with the floor of the destination room. This gives us the object "floor" center in the coordinate system defined by the destination room box. The transferred object will be aligned with the walls of the destination room, denoted by the arrows in (**d**). We determine the object width and length by intersecting the arrows with the dashed lines (**d**), which might not be aligned with the destination walls due to the differing camera parameters. The object height is found by projecting the corners on the top of the object under the original camera (**c**). For each corner c, we consider ray r_1 through its projection and the destination camera, and ray r_2 orthogonal to the room floor and through the corresponding corner of the object on the floor. The 3D position of c in the destination room is given by the point on r_2 closest to r_1. The 3D length of the vertical dashed lines determines the height of the transferred object. This length might not be the same for all four lines, and we set the height of the object by averaging them. The final transferred object is shown in (**e**). The second row illustrates the equivalent procedure for frames, where we consider the projection of the frame's corners on the closest wall

suggests that adding realistic objects in the room drives the inference towards better spatial configurations.

We then evaluate the benefits of the different components of the likelihood, considering three cases: 1) we do not allow any object in the room, 2) we allow objects in the scene, and 3) we allow exchanging information among threads. Interestingly, exchanging objects allows choosing a better room box, since more objects provide more evidence on the correct room layout. There is no improvement when only edges are used, and we believe this happens because the likelihood is not robust enough in this case. Qualitative examples of these improvements are shown in Fig. 18.

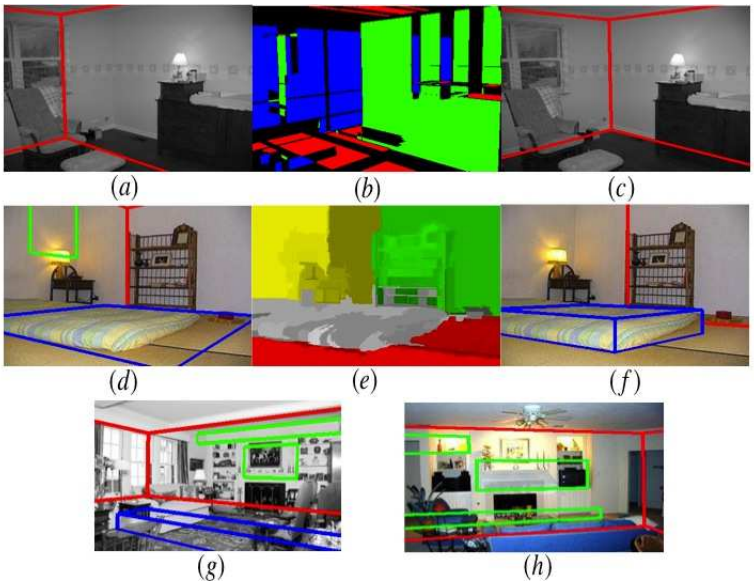

Fig. 18 Effects of the individual components of our model. Using only edges we get a poor estimate of the room box, due to the edge detector missing the wall edges (**a**). By adding orientation maps to the likelihood (**b**), where we draw in green and blue the pixels assigned to the two orthogonal vertical surfaces and in red the horizontal pixels, we obtain a better room box (**c**). Also adding geometric context helps. In (**d**), the algorithm is confused by several mistakes made in estimating the orientation maps, but geometric context (**e**) helps the reconstruction process (**f**). In (**e**), we used red for pixels labeled as floor, yellow for left wall, green for middle wall, and gray for objects. Last, using priors on object 3D position and size avoids proposing objects with unrealistic size, which would otherwise often "latch" to image features (**g**-**h**)

Table 2 shows that our method is comparable to the state-of-the-art on the Hedau dataset, and also report results on the UCB dataset [24], consisting of 340 black and white images. When comparing with state-of-the-art, we consider our full algorithm, including all the likelihood components, and with the object exchange enabled.

Then, we evaluate on object recognition. We ground truthed the UCB [24] and Hedau dataset [8] by manually identifying the 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. 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.

We report in Table 3 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 subcategory accuracy cannot be evaluated. Table 4 shows that geometric context improves all recognition scores, except for frame subcategory classification on the Hedau dataset. 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 in general geometric context only introduces small improvements in the subcategory classification, if we exclude furniture on the UCB dataset. This is because adding a new feature allows to obtain more accurate object fits, but the "burden" of classification is still entirely on the prior on size and position. We posit that having different geometric models for object classes, such as tables with legs or couches with backrests, as well as class-dependent appearance models, would improve this score.

The effects of exchanging objects among threads are available in Table 5. 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. 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, 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. 19, where we show the scene reconstructions provided by the full algorithm. The most typical failures are shown in Fig. 20.

Table 1 Analysis of the components of our approach based on room layout error, evaluated on the Hedau test set [8]

	Edges + OM		Edges	Edges + OM	Edges + OM + GC
No Prior	20.4%	No objects	24.1%	**21.3%**	21.8%
Room prior	19.7%	Objects	20.7%	17.8%	**17.2%**
Room + obj prior	**17.8%**	Exchange objects	14.2%	14.6%	**13.6%**

Table 2 Comparison with state-of-the-art on room layout error

Dataset	Hedau [8]	Lee [15]	Schwing [19]	Our full approach
Hedau [8]	21.2%	16.2%	**12.8%**	13.6%
UCB [24]	NA	NA	NA	**14.2%**

Table 3 Benefits of object priors evaluated on UCB and Hedau datasets. P, R, and S denote Precision, Recall and Subcategory Accuracy

UCB [24]	P	R	S	Hedau [8]	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 4 Benefits of geometric context evaluated on UCB and Hedau datasets

UCB [24]	P	R	S	Hedau [8]	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%

Table 5 Effects of exchanging objects among threads on UCB and Hedau datasets

UCB [24]	P	R	S	Hedau [8]	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 6 Confusion matrices on Hedau test set [8]

	Bed	Cabinet	Couch	Table		Door	Picture	Window
Bed	14	1	7	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				

Fig. 19 Scene reconstructions provided by our full approach. We show the room box in red, furniture in blue and frames in green

5 Conclusions

Our top-down Bayesian approach for understanding indoor scenes is competitive with state-of-the-art approaches on the task of recovering the room box. Interestingly, priors on 3D geometry both improved object recognition scores, and provided better room box estimates. We believe this is an interesting finding towards providing top-down scene interpretations that are globally consistent in terms of geometry and semantics. Another strength of this method is that it does not commit to partial configurations, and can recover from initial errors, one example being the improvements on the initial estimate of the room box.

We posit that our method will prove more powerful as it integrates more sophisticated object models, including non convex-approximations such as tables with legs. In fact, more detailed geometry should help distinguish between classes that are similar in position and size. Also, recent work by Hedau et al. [10] showed that scene interpretation is enhanced by more accurate geometric models, such as blocks with backrests. More sophisticated models introduce additional complex constraints on the structure of objects (for example, chairs with four symmetric legs), which our Bayesian framework could accommodate relatively easily. Further, the Bayesian formulation also allows integrating additional image evidence that would make the model more robust. For example, visual inspection of the results suggested that adding category dependent appearance models is a promising direction of research,

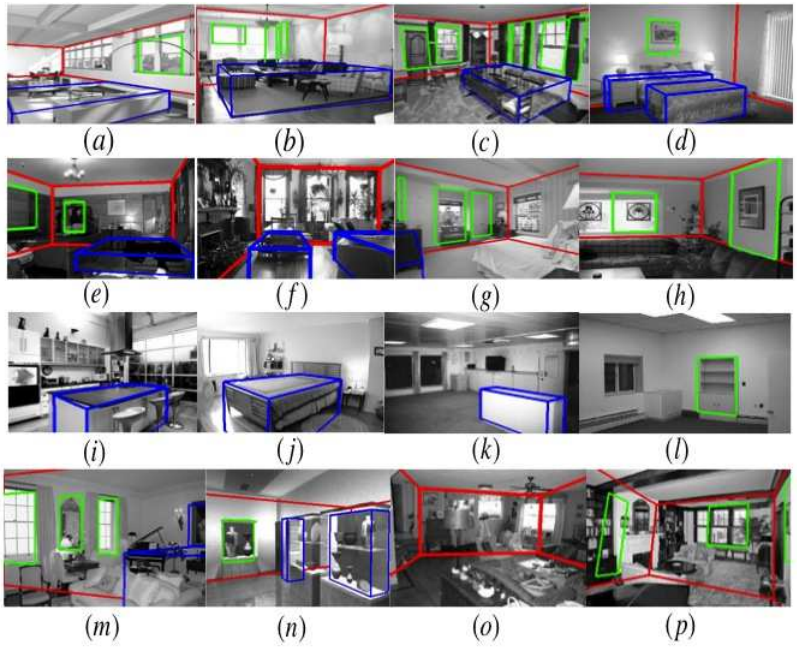

Fig. 20 Some typical failures. Due to clutter we often hallucinate objects (**a**-**d**). For example, in (**b**) we propose a bed whose edges "latch" to those of the carpet, and to the top of the armchairs. However, in this case our approach still provides a reasonable approximation of the space occupancy of the scene. In other situations (**e**-**f**), we hallucinate boxes in the gap among objects, thus providing a wrong estimate of the occupied space, or completely miss objects (**g**-**h**). This mostly happens when faint edges are missed by the edge detector, such as those of the bed in (**g**). Further, we often confuse object categories similar in size and position. For example, the table in (**i**) is wrongly labeled as a bed, and the opposite happens in (**j**). The wall in (**k**) is labeled as a cabinet, and we sometimes confuse an object that is directly facing the camera for a frame (**l**). We also show some cases where the algorithm estimated the wrong room box (**m**-**n**). In (**m**), this is caused by the edge detector missing the edge between the room walls and the ceiling completely. Other more catastrophic failures are due to bad initial estimates of the camera, from which the algorithm could not recover (**o**-**p**)

as well as enforcing that projections of objects are uniform in terms of color and texture.

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant No. 0747511. We thank Joseph Schlecht for his contributions and suggestions in designing the code base. We also acknowledge the valuable help of Joshua Bowdish, Ernesto Brau, Andrew Emmott, Daniel Fried, Jinyan Guan, Emily Hartley, Bonnie Kermgard, and Philip Lee.

References

1. Coughlan, J. M., Yuille, A. L., *Manhattan World: Compass Direction from a Single Image by Bayesian Inference*, ICCV (1999)
2. Delage, E., Lee, H. L, Ng, A. Y., *Automatic single-image 3d reconstructions of indoor manhattan world scenes*, ISRR (2005)
3. Del Pero, L., Guan, J., Brau, E., Schlecht, J., Barnard, K., *Sampling bedrooms*, CVPR (2011)
4. Del Pero, L., Bowdish, J., Fried, D., Kermgard, B., Hartley, E., Barnard, K., *Bayesian geometric modeling of indoor scenes*, CVPR (2012)
5. Green, P. J., *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*, Biometrika **82** (1995)
6. Green, P. J., *Trans-dimensional markov chain monte carlo*, Highly Structured Stochastic Systems (2003)
7. Gupta, A., Satkin, S., Efros, A. A., Hebert, M., *From 3D Scene Geometry to Human Workspace*, CVPR (2011)
8. Hedau, V., Hoiem, D., Forsyth, D., *Recovering the Spatial Layout of Cluttered Rooms*, ICCV (2009)
9. Hedau, V., Hoiem, D., Forsyth, D., *Thinking Inside the Box: Using Appearance Models and Context Based on Room Geometry*, ECCV (2010).
10. Hedau, V., Hoiem, D., Forsyth, D., *Recovering free space of indoor scenes from a single image*, CVPR (2012)
11. Hoiem, D., Efros, A. A., Hebert, M., *Geometric Context from a Single Image*, ICCV (2005)
12. Hoiem, D., Efros, A. A., Hebert, M., *Putting Objects in Perspective*, CVPR (2006)
13. Karsch, K., Hedau, V., Forsyth, D., Hoiem, D., *Rendering synthetic objects into legacy photographs*, SIGGRAPH Asia (2011)
14. Lee, D.C., Hebert, M., Kanade, T., *Geometric reasoning for single image structure recovery*, CVPR (2009)
15. Lee, D.C., Gupta, A., Hebert, M., Kanade, T., *Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces*, NIPS (2010)
16. Neal, R. M., *Probabilistic inference using Markov chain Monte Carlo methods*, Technical Report (1993)
17. Rother, C., *A new approach to vanishing point detection in architectural*, IVC **20** (2002)
18. Schlecht, J., Barnard, K.: *Learning models of object structure*, NIPS (2009)
19. Schwing, A., Hazan, T., Pollefeys, M., Urtasun R., *Efficient Structure Prediction with Latent Variables for General Graphics Models*, CVPR (2012)
20. Shi, F., Zhang X., Liu Y., *A new method of camera pose estimation using 2D-3D corner correspondence*, Pattern Recognition Letters (2004)
21. Tsai, G., Xu, C., Liu, J., Kuipers, B., *Real-time indoor scene understanding using Bayesian filtering with motion cues*, ICCV (2011)
22. Tu, Z., Zhu, S., *Image segmentation by data-driven markov chain monte-carlo*, PAMI (2002)
23. Wang., H., Gould S., Koller, D., *Discriminative Learning with Latent Variables for Cluttered Indoor Scene Understanding*, ECCV (2010)
24. Yu, S. X., Zhang, H., Malik, J., *Inferring Spatial Layout from A Single Image via Depth-Ordered Grouping*, POCV (2008)
25. Hartley, R. I., Zisserman, A., *Multiple View Geometry in Computer Vision*, Cambridge University Press (2004)
26. Zhu, S.-C., Zhang, R., Tu, Z., *Integrating topdown/bottom-up for object recognition by data driven Markov chain Monte Carlo*, CVPR (2000)