Learning models of object structure

Joseph Schlecht Kobus Barnard

Computer Science Department

University of Arizona

{schlecht, kobus}@cs.arizona.edu

Abstract

We develop an approach to learn stochastic geometric models of object categories from single view images. We focus here on models expressible as a spatially contiguous assemblage of blocks. Model topologies are learned across groups of images, and one or more such topologies is linked to an object category (e.g. chairs). Fitting learned topologies to an image can be used to identify the object class, as well as detail its geometry. The latter goes beyond labeling objects, as it provides the geometric structure of particular instances.

We learn the models using joint statistical inference over structure parameters, camera parameters, and instance parameters. These produce an image likelihood through a statistical imaging model. For inference we use trans-dimensional sampling to explore topology hypotheses, and alternate between Metropolis-Hastings and stochastic dynamics to explore instance parameters at an effective rate. During inference we maintain convergence to the target distribution. In practice we are often able to find a good optimum in the samples.

We experiment with data sets consisting of images of standard furniture such as tables and chairs. We demonstrate that we can learn models that are able to infer category and geometry in this kind of data.

1. Introduction

In this paper we develop an approach to learn stochastic 3D geometric models of object categories from single view images. Exploiting such models for object recognition systems enables going beyond simple labeling. In particular, fitting such models opens up opportunities to reason about function or utility, how the particular object integrates into the scene (i.e., perhaps it is an obstacle), how the form of the particular instance is related to others in its category (i.e., perhaps it is a particularly tall and narrow one), and how categories themselves are related.

Capturing the wide variation in both topology and geometry within object categories, and in particular, finding good estimates for the underlying statistics, suggests a large scale learning approach. We propose exploiting the growing number of labeled single-view images to learn such models. While our approach is trivially extendable to exploit multiple views of the same object, large quantities of such data is rare. Further, the key issue is to learn about the variation of the category. Put differently, if we are limited to 100 images, we would prefer to have 100 images of different examples, rather than, say 10 views of 10 examples.

Representing, learning, and using object statistical geometric properties is potentially simpler in the context of 3D models. In contrast, statistical models that encode image-based appearance characteristics and/or part configuration statistics must deal with confounds due to the imaging process. For example, right angles in 3D can have a wide variety of angles in the image plan. But this requires using the same representations for variability for both structure variation and pose variation. This means that the represented geometry is less specific and less informative. By contrast, encoding the structure variation in 3D models is simpler and more informative because they are linked to the object alone.

To deal with the effect of an unknown camera, we estimate the camera parameters simultaneously while fitting the model hypothesis. A 3D model hypothesis is a relatively strong hint as to what the camera might be. Further, we make the observation that the variations due to standard camera projection are quite unlike typical category variation. Hence, in the context of a given object model hypothesis, the fact that the camera is not known is not a significant impediment, and much can be determined about the camera, statistically, under that hypothesis.

We explore our approach using object models that are expressible as a spatially contiguous assemblage of blocks. We also include in the model a prior on right angles between blocks. We experiment with images from three sub-categories of furniture: tables, chairs, and bookcases. We further simplify matters by considering images where there are minimal distracting features in the background. Within this domain, we are able to automatically learn topologies. The models can then be used to identify the object category using statistical inference. Recognition of objects in clutter is likely effective with this approach, but we have yet to integrate support for occlusion of object parts into our inference process.

We learn the parameters of each category model using Bayesian inference over multiple image examples for the category. Thus we have a number of parameters specifying the category topology that apply to all images of objects from the category. Further, as a side effect, the inference process finds instance parameters that apply specifically to each object. For example, all tables have legs and a top, but the proportions of the parts differ among the instances. In addition, the camera parameters for each image are determined, as these are simultaneously fit with the object models. The object and camera hypotheses are combined with an imaging model to provide the image likelihood that drives the inference process.

For inference we need to find parameters that give a high likelihood of the data from multiple images. To do so, we take a sampling approach. Because we are searching for model topologies, we need to search among models with varying dimension. For this we use the trans-dimensional sampling framework [8, 9]. We explore the posterior space within a given probability space of a particular dimension by combining standard Metropolis-Hastings [21, 10, 1, 18], with stochastic dynamics [23]. As developed further below, these two methods have complementary strengths for our problem. Importantly, we arrange the sampling so that the hybrid of samplers are guaranteed to converge to the posterior distribution. This ensures that the space will be completely explored, given enough time.

1.1. Related work

Most work on learning representations for object categories has focused on image-based appearance characteristics and/or part configuration statistics (e.g., [6, 5, 4, 16, 28, 17]). These approaches typically rely on effective descriptors that are somewhat resilient to pose change (e.g., [20]). A second force favoring learning 2D representations is the explosion of readily available images compared with that for 3D structure, and thus treating category learning as statistical pattern recognition is more convenient in the data domain (2D images). However, some researchers have started imposing more projective geometry into the spatial models. For example, Savarese and Fei-Fei [24, 25] build a model where arranged parts are linked by a fundamental matrix. Their training process is helped by multiple examples of the same objects, but notably they are able to use training data with clutter. Their approach is different than ours is that models are built more bottom up, and this process is somewhat reliant on the presence surface textures. Our work is driven by parametric parts that provide strong cues when they are appropriate. A different strategy proposed by Hoeim et al. [12] is to fit a deformable 3D blob to cars, driven largely by appearance cues mapped onto the model. Their choice modeling in 3D simplifies a number of issues, and provides for more natural integration with work in understanding scene geometry [11], as is the case for us. However, our modeling approach is different in that we focus in on learning topologies for assemblages of parameterized parts, instead of working with deformation of a single structure. Our interest in learning structure topologies relates to recent work in learning abstract topologies [29, 15], and more concretely, structure models for 2D images of objects (e.g., [30, 31]) constrained by grammar representations.

Our work is also related to a large body of older work on finding and recognizing objects in images, given a precises 3D model, such as one might have for machined parts in an industrial setting (e.g., [19, 13]). Finally, we also acknowledge work on fitting deformable models of known topology to 2D images in the case of human pose estimation (e.g, [22, 27, 26]. We are trying to learn such configurable part models for categories, and thus the inference procedures potentially have a lot in common.

2. Our approach

From an image collection of an object category, we learn a three-dimensional structure model that probabilistically describes the form and appearance of the category. We accomplish this by inferring instance parameters of object and camera models for each image, and jointly learning across these a category-level organization of object parts (topology) and their distributions. Since an object category typically has multiple, closely related structure topologies, e.g. chairs with and without armrests, we learn sub-categories of structure. This enables us to capture variation within the 3D structure of object categories and can be used to recognize or detect instances of our model in new images.

In our approach we present a generative model for the 3D structure of an object, the camera viewing it and the image captured. Our representation of an object comprises a set of 3D parts linked together by a learned topology. The parts are geometric primitives representing unit pieces of structure and are generated from distribution parameters specific to the object category. The topology characterizes the spatial relationship between parts, and together with the parts, forms the 3D object model. We learn several object sub-categories by clustering over part topologies and distributions; each object model instance is generated by one of the clusters. We further model the camera capturing the view of an object into an image, enabling an understanding of imaged objects under arbitrary views. Finally, conditioned on the object and camera models, we represent independently detected image features, such as edge and surface points, as generated by object parts projected under the camera model. By combining the object, camera and image models, we have a process to generate images of objects that we can use for model inference.

Following a Bayesian strategy, we reverse our forward model and, from detected features in an image, simultaneously fit the most likely 3D object model and camera to have generated them. Using the inferred object and camera models for a set of images in a category, we learn the form of the category topology and part distributions. In this way we have two types of parameters in our model: per image and per category (or sub-category in the case of multiple clusters). We infer both types of parameters together from a set of training images of an object category. For recognition or detection in a new image, we need only infer the instance parameters.

In describing our model and its process of inference, we first introduce some notation and parameter descriptions. For a single image, we label the corresponding set of structure parts in our object model $\mathbf{s}$ and the camera capturing it $\mathbf{c}$. The topology shared across multiple images generated by the same object category is given by $\mathbf{t}$. We label the similarly shared cluster and distribution parameters for structure sub-categories $\mathbf{r_s}$ and camera distribution $\mathbf{r_c}$. The number of parts in the object model is unknown a priori, making the model parameter set variably sized. For an object with M sub-categories, we denote the number of parts in each object model as $\mathbf{n} = n_1, \ldots, n_M$. Since the dimension of our model depends upon the number of parts, we denote the set of category model parameters for one image

$$\boldsymbol{\theta}^{(\mathbf{n})} = (\mathbf{c}, \mathbf{s}, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}, \mathbf{n}) \ . \tag{1}$$

The camera parameters for an image are shared across the sub-categories. We could learn sub-category camera parameters, but multiple structure motifs of an object typically are independent of how they are viewed, *e.g.* chairs with armrests are similarly viewed as those without. Hence, we label the parameters for a single image under the m^{th} sub-category as a subset of $\boldsymbol{\theta}^{(\mathbf{n})}$,

$$\boldsymbol{\theta}_m^{(n_m)} = (\mathbf{c}, \mathbf{s}_m, \mathbf{t}_m, \mathbf{r_c}, \mathbf{r}_{\mathbf{s}m}, n_m) \ , \tag{2}$$

which has a shared camera and generating distribution.

Given a set of D images containing examples of the an object category, our goal is to learn the model $\boldsymbol{\Theta}^{(\mathbf{n})}$ from their detected image features $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_D$. In addition to category-level parameters, $\boldsymbol{\Theta}^{(\mathbf{n})}$ comprises camera instances $\mathbf{C} = \mathbf{c}_1, \ldots, \mathbf{c}_D$ for each image and structure part parameters $\mathbf{S}_m = \mathbf{s}_{m1}, \ldots, \mathbf{s}_{mD}$ for the m^{th} sub-category and each image. Our posterior over the parameters then takes the form

$$p\left(\boldsymbol{\Theta}^{(\mathbf{n})} \,|\, \mathbf{X}\right) = p\left(\mathbf{X}, \boldsymbol{\Theta}^{(\mathbf{n})}\right) / \int p\left(\mathbf{X}, \boldsymbol{\Theta}^{(\mathbf{n})}\right) d\boldsymbol{\Theta}^{(\mathbf{n})} \, . \tag{3}$$

The integral behaves as a constant and is not computed; it is canceled out during our inference process, as we show. The joint density function over the features and parameters, however, is the core of our inference and requires further attention.

Since instance parameters are bound to the feature data in each image, we separate the joint density into a likelihood and prior as follows

$$
\begin{aligned}
p\left(\mathbf{X}, \mathbf{\Theta}^{(\mathbf{n})}\right) &= p^{(\mathbf{n})}(\mathbf{X}, \mathbf{C}, \mathbf{S}, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}, \mathbf{n}) &(4)\\
&= p^{(\mathbf{n})}(\mathbf{X}, \mathbf{C}, \mathbf{S} \,|\, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s})\, p^{(\mathbf{n})}(\mathbf{t}, \mathbf{r_c}, \mathbf{r_s}, \mathbf{n})\,, &(5)
\end{aligned}
$$

where we use the notation $p^{(\mathbf{n})}(\cdot)$ to denote the $\mathbf{n}^{\text{th}}$ sub-density function over a variably sized parameter set. Conditioned on the category parameters, our likelihood assume the D sets of image features and instance parameters are independent,

$$
p^{(\mathbf{n})}(\mathbf{X}, \mathbf{C}, \mathbf{S} \,|\, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}) = \prod_{d=1}^{D} p^{(\mathbf{n})}(\mathbf{x}_d, \mathbf{c}_d, \mathbf{s}_d \,|\, \mathbf{t}, \boldsymbol{r}_\mathbf{c}, \boldsymbol{r}_\mathbf{s})\,. \tag{6}
$$

This seems a fairly safe assumption; if two images contain examples of an object, then their particular appearances are typically independent. An exception to this assumption, however, includes inadequately modeling sub-category structure variation within a class of objects.

From the independent sets of features and instance parameters in (6), we develop a likelihood clustering model over sub-categories of object structure. The feature data and structure parameters are generated by a sub-category cluster with weights and distribution defined by $\mathbf{r_s} = (\boldsymbol{\pi}, \boldsymbol{\mu}_\mathbf{s}, \boldsymbol{\Sigma}_\mathbf{s})$. The camera is shared across clusters, however, and drawn from a distribution defined by $\mathbf{r_c} = (\boldsymbol{\mu}_\mathbf{c}, \boldsymbol{\Sigma}_\mathbf{c})$. We formalize the likelihood of an object, camera, and image features under M clusters as

$$
\begin{aligned}
&p^{(\mathbf{n})}(\mathbf{x}_d, \mathbf{c}_d, \mathbf{s}_d \,|\, \mathbf{t}, \boldsymbol{r}_\mathbf{c}, \boldsymbol{r}_\mathbf{s})\\
&\quad = \sum_{m=1}^{M} \pi_m \underbrace{p^{(n_m)}(\mathbf{x}_d \,|\, \mathbf{c}_d, \mathbf{s}_{md})}_{\text{Image}} \underbrace{p(\mathbf{c}_d \,|\, \boldsymbol{\mu}_\mathbf{c}, \boldsymbol{\Sigma}_\mathbf{c})}_{\text{Camera}} \underbrace{p^{(n_m)}(\mathbf{s}_{md} \,|\, \mathbf{t}_m, \boldsymbol{\mu}_{\mathbf{s}m}, \boldsymbol{\Sigma}_{\mathbf{s}m})}_{\text{Object}}\,.
\end{aligned}
\tag{7}
$$

We arrive at equation (7) by introducing a binary assignment vector $\mathbf{z}$ for each image feature set, such that $z_m = 1$ if the m^{th} cluster generated it and 0 otherwise. As the cluster assignments are not actually known, we marginalize the likelihood (6) over all $\mathbf{z}$, weighting each sub-category cluster according to

$$\pi_m = p(z_m = 1)\,. \tag{8}$$

By assuming the feature set and instance parameters to be conditionally independent given the object sub-category, we formally derive the likelihood clustering in (7) as follows

$$
\begin{aligned}
&p^{(\mathbf{n})}(\mathbf{x}, \mathbf{c}, \mathbf{s} \,|\, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}) \\
&\quad = \sum_{\mathbf{z}} p^{(\mathbf{n})}(\mathbf{x}, \mathbf{c}, \mathbf{s}, \mathbf{z} \,|\, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}) && (9)\\
&\quad = \sum_{\mathbf{z}} \prod_{m=1}^{M} p^{(\mathbf{n})}(\mathbf{x}, \mathbf{c}, \mathbf{s}_m, z_m \,|\, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s})^{z_m} && (10)\\
&\quad = \sum_{m=1}^{M} p\,(z_m = 1 \,|\, \pi_m)\;\; p^{(n_m)}(\mathbf{x} \,|\, \mathbf{c}, \mathbf{s}_m, \mathbf{t}_m)\; p(\mathbf{c} \,|\, \mathbf{r_c})\; p^{(n_m)}(\mathbf{s}_m \,|\, \mathbf{t}_m, \mathbf{r}_{\mathbf{s}m})\,. && (11)
\end{aligned}
$$

In this way, we define each independent component of our likelihood for a single image.

For the prior probability distribution over model parameters, we assume sub-category and parameter independence, where the topologies are conditionally independent given the number of parts in the model. The prior in our posterior (5) then expands to

$$p^{(\mathbf{n})}(\mathbf{t}, \mathbf{r_c}, \mathbf{r_s}, \mathbf{n}) = p(\mathbf{r_c}) \prod_{m=1}^{M} p^{(n_m)}(\mathbf{t}_m \,|\, n_m)\; p^{(n_m)}(\mathbf{r}_{\mathbf{s}m})\; p(n_m)\,. \tag{12}$$

The parameters for camera and structure instances, $\mathbf{r_c}$ and $\mathbf{r_s}$, are Gaussian distributed with hyperparameters empirically chosen from experiments using subsets of image categories with similar topologies. The number of parts in the object sub-categories, $\mathbf{n}$, is uniformly distributed. In the following section focused on our object model, we describe the prior probability of a topology.

The joint density over image features and model parameters created from the likelihood (6) and

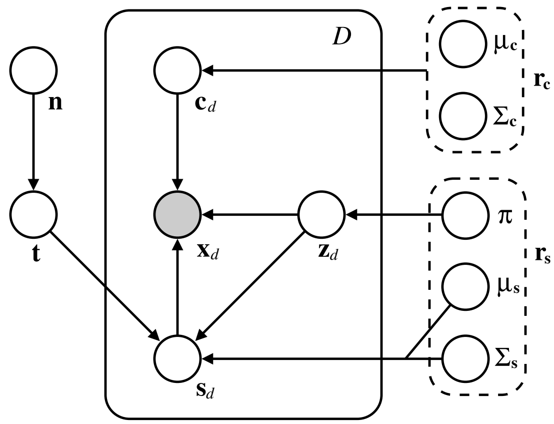


Figure 1: Graphical model for our posterior joint density (4) defined over D sets of image features $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_D$ and model parameters $\boldsymbol{\Theta}^{(\mathbf{n})} = (\mathbf{C}, \mathbf{S}, \mathbf{t}, \mathbf{r_c}, \mathbf{r_s}, \mathbf{n})$.

prior (12) describes our generative model and Bayesian approach for learning 3D object structure in images. Figure 1 shows the graphical version of this model and summarizes its parameter relationships. In the next few sections, we detail the object, camera, and image components of our likelihood and prior.

2.1. Object model

We have four basic types of structure parts: Individual blocks, symmetric pairs of blocks, symmetric quartets of blocks, and symmetric stacks of blocks.

The parameters of each block are assumed independent of the others conditioned on the subcategory topology and part parameters.

Each block has three internal parameters for width, height, and length. It also has potentially two external attachment parameters (u,v) for each face; we only allow one part to be attached to a face at a time. Note that the parts with multiple blocks are specified with essentially one attachment, since the other attachments are symmetric.

The blocks are positioned by in an object coordinate system defined by an origin o. An important note that the location of blocks within this coordinate system is entirely defined by o and the attachments points from one block to another. Thus, it does not seem necessary to have position in object space of each block as a parameter; the object model is assumed connected.

Figure 2: Object model

$$\mathbf{b}_{\textsf{ind}} = (w,\, h,\, l\,) \tag{13}$$

$$\mathbf{b}_{\textsf{set}} = (w,\, h,\, l,\, d_w,\, d_l,\, \tau) \tag{14}$$

We constrain the blocks to be connected at right angles and centrally aligned, so the orientation and position of the blocks, beyond attachment, are implicit determined from the object model. In essence, the model is a stack of block and symmetric block sets, with block size and attachment points being the main variables describing an instance. Figure 2 shows a couple example individual and block sets and potential configurations. Despite its simplicity, this model can approximate a surprising range of man made objects.

Combined with a y-axis rotation angle, φ, and position in $\mathbb{R}^3$, a collection of blocks and block sets comprise the structure of our object model

$$\mathbf{s}^{(n_1,n_2)} = \left(\varphi,\, x,\, y,\, z,\, \mathbf{b}_{\textsf{ind}}^{(n_1)},\, \mathbf{b}_{\textsf{set}}^{(n_2)}\right)\ , \tag{15}$$

with $\mathbf{b}^{(n_1)} = (\mathbf{b}_1, \ldots, \mathbf{b}_{n_1})$ defined similarly for both types of blocks.

We specify the object model topology as a set of dependencies. For two blocks $\mathbf{b}_i$ and $\mathbf{b}_j$ connected in the direction of their height, a topology parameter is defined as

$$\mathbf{t}_k = \left(i,\, j\, :\, \mathbf{b}_i \xleftrightarrow{h} \mathbf{b}_j\right)\ . \tag{16}$$

Let $\mathbf{t}^{(n_3)}$ be a collection of topology constraints. Our object model $\mathbf{o}^{(\mathbf{n})} = \left(\mathbf{s}^{(n_1,n_2)},\, \mathbf{t}^{(n_3)}\right)$ combines the structure elements for individual objects and the topology, which is shared across multiple instance of the the category. We denote the space over our object model as $\mathsf{S}^{(n_1,n_2)} \times \mathsf{T}^{(n_3)}$.

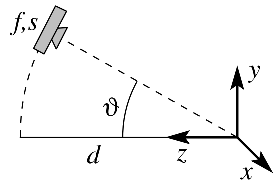


Figure 3: Camera parameters

2.2. Camera model

In the paradigm of learning about an object from a single view, the full specification of the camera and the object position and scale leads to a redundant set of parameters. We choose a minimal set for inference that retains full expressiveness as follows.

Without a priori information we are unable to distinguish the actual size of an object from its distance to the camera. For this reason we constrain the camera to be at a fixed distance from the world origin and accept knowing the size of an object up to a scaling factor. If at some point we learn what the scaling factor is, we would be able to plug this in and know actual sizes and positions of objects in the world.

We assume that objects of interest are variably positioned near the ground plane x, y and constrain the camera to be always looking at the world origin. Because we allow the object to rotate around its vertical axis, we only need to specify the camera altitude angle, ϑ. Thus we set the horizontal x-coordinate of the camera in the world to zero and allow ϑ to be the only variable extrinsic parameter. In other words, the position of the camera is constrained to a circular arc on the y, z-plane. See Figure 3 for an illustration.

We further model the amount of perspective in the image from the camera by parameterizing its focal length, f, and inferring it from the image. The focal length parameter strongly interacts with the scale, s, of the objects in the world. However, it affects the convergence of parallel lines and specifies a unique image. Our camera parameter vector is then given by

$$\mathbf{c} = (\vartheta,\ f,\ s)\ , \tag{17}$$

where $\vartheta \in [0, \pi/2]$, and $f, s > 0$. We refer to the space over our camera parameters as C.

2.3. Image model

We represent an image as a collection of detected feature sets that are statistically generated by an instance of our object and camera. We model each of the image feature sets as arising from a corresponding feature generator that depends on projected object information. For example, we generate edge points from projected object contours and image foreground from colored surface points. Figure 4 illustrates this representation of detected image features. Our likelihood over image feature sets, conditioned on an object and camera model, captures the process by which features are generated and measures how well a model explains their observations.

Given an object and camera, a feature generator stochastically produces the response of a detector at every pixel of an image. Thus each pixel has a non-zero probability of a feature being generated over it by the model, which we assume is independent from all other pixels' chances, conditioned on our model. This is a strong assumption, but the projected object model provides a good hint about what features we expect to see in the image per pixel, so it seems warranted. Our image model is then per pixel, and we compute the likelihood of a feature detector's response per pixel given object and camera information.

We formally define the likelihood of image feature sets as a product over per pixel observations. For an image with N_x pixels, we assume independence, as previously mentioned, between per pixel feature responses conditioned on the model. We further assume independence among the G different types of generated features detectable in the image. For the d^{th} image with feature sets $\mathbf{x}_d = \mathbf{x}_{d1}, \ldots, \mathbf{x}_{dG}$, we expand the image component of equation (7) to

$$p^{(n_m)}(\mathbf{x}_d \,|\, \mathbf{c}_d, \mathbf{s}_{md}, \mathbf{t}_m) = \prod_{g=1}^{G} \prod_{i=1}^{N_x} f_{\theta g}^{(n_m)}(x_{dgi})\,. \tag{18}$$

The function $f_{\theta g}^{(n_m)}(\cdot)$ measures the likelihood of a feature generator producing the response of a detector at each pixel using our object and camera models. The number of pixels per image, N_x, is equivalent for all images in a category; hence, the number of potentially detected features in each image is the same. This is a simplification of our model in order to compare the likelihoods of one image with another.

It may not necessarily be the case that the G detected feature sets are independent. As in the pixel independence assumption, however, conditioning on the model provides a way to lessen [this dependency] and simplify our model. We further observe that detected features of different types do not always have [strong] dependencies. This is particularly true for edge and surface points. Since edge points are located in surface regions of high color transition, and most surface color is not in these regions, it is unlikely a strong dependency exists between a particular surface point and an edge detection. For these reasons we believe it is reasonable to assume such independence.

Using our approach to the image likelihood, it is not difficult to model many different types detectable features. We currently have chosen to model edge points and image foreground since they are straightforward to extract from the image, provide a good representation of object part structure and location, and are readily modeled by our object representation. As Figure 4 shows, the projected object contours model detected edge points and colored surface points model the detected image foreground. Our foreground representation is essentially a binary color indicating whether a pixel contains a surface point in the foreground. We could easily extend this to account for more color in the foreground surface points and add another feature generator to the image model. We first describe how we model edge point generation followed by surface point generation.

Edge point generator

Image edge points occur at pixel locations where there is a large change in color relative to nearby pixels. We model edge points as generated from discretized points along the projected 3D contours of our object model. The object contours arise where two or more surfaces meet with different orientation, each having potentially different color or shading. The projected object contour points are positioned in a hypothesized model image and contain contour orientation information. This representation is consistent with edges detected with gradient-based methods that give edge point pixel locations and a gradient vector indicating edge orientation.

In addition to location and orientation, an edge detector indicates whether an image pixel contains

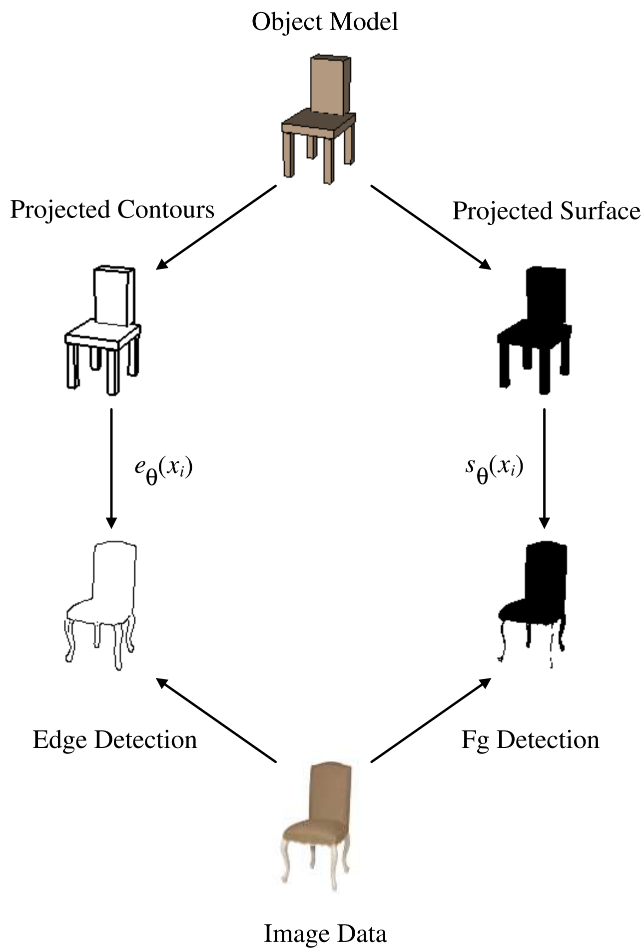


Figure 4: Example of the generative image model for detected features. The top of the figure gives a rendering of the object and camera models fit to the image at the bottom. The downward arrows show the process of statistical generation of image features. The upward arrows are feature detection in the image data.

an edge point. Since the feature generator likelihood in (18) is computed over all detection responses in an image, we define the edge generator likelihood as

$$\prod_{i=1}^{N_x} f_\theta(x_i) = \prod_{i=1}^{N_x} e_\theta(x_i)^{\mathcal{E}_i} \cdot e'_\theta(x_i)^{(1-\mathcal{E}_i)} \,, \tag{19}$$

where the probability density function $e_\theta(\cdot)$ gives the likelihood of detected edge point at the i^{th} pixel, and $e'_\theta(\cdot)$ is the density for pixel locations not containing an edge point. The two density functions are selected per pixel by an indicator $\mathcal{E}_i$, which is 1 if the pixel is an edge point and 0 otherwise. We have suppressed the generator index g and sub-density index $^{(n_m)}$ notation.

The edge point density $e_\theta(\cdot)$ is defined over detected edges that have been generated by projected contour points of the object model. For each edge point, suppose we know which projected model point generated it. Then we assume the i^{th} edge point generated from the j^{th} model point has some Gaussian distributed displacement d_{ij} in the perpendicular direction of the projected contour containing the model point. We further assume the gradient direction of the generated edge point has some Gaussian distributed angle difference ϕ_{ij} with the perpendicular direction of the projected contour. So we define the likelihood to be the product of two Gaussians, measuring the distance and angular error, assuming independence. Let m_j be the known model point to have generated x_i, then

$$e_\theta(x_i) = c_e \, \mathcal{N}\left(d_{ij}; 0, \sigma_d\right) \mathcal{N}\left(\phi_{ij}; 0, \sigma_\phi\right) \tag{20}$$

where the perpendicular distance between x_i and m_j and angular difference between edge point gradient $\mathbf{g}_i$ and model contour perpendicular $\mathbf{v}_j$ are defined

$$d_{ij} = \| \, x_i - m_j \, \| \tag{21}$$

$$\phi_{ij} = \cos^{-1}\left(\frac{\mathbf{g}_i^{\mathrm{T}} \, \mathbf{v}_j}{\|\mathbf{g}_i\| \, \|\mathbf{v}_j\|}\right). \tag{22}$$

The range of d_{ij} is ≥ 0, and the angle ϕ_{ij} is in $[0, 1]$; thus, the Gaussians are truncated and we normalize their product with the constant c_e.

Pixels not containing an edge point still give an edge detection response from a nearby projected model contour. Suppose we know the projected model point generating each of these non-edge responses. Then we define the probability of an edge detection response x_i that does not contain an edge point as

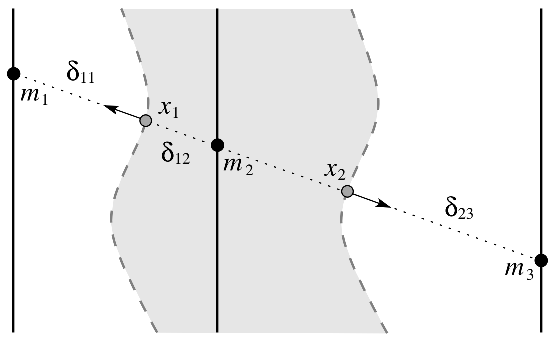
$$e'_\theta(x_i) = 1 - \int_{\mathbf{X}'_i} e_\theta(x)\, dx\,, \tag{23}$$

where $\mathbf{X}'_i$ is the space of all edge detection responses at the same pixel location as x_i, but that contain an edge point. That is, we define $e'_\theta(x_i)$ as the complement of the probability a model point generates any detection response containing an edge point at the i^{th} pixel.

Unfortunately, during model inference with actual detected edge points in an image, we do not know the correspondence between hypothesized model points and the edge detection responses x_i. We could search for the most likely correspondence linking edge detection responses and model points, but there are exponentially many of them. Instead, we build uncertainty into the point correspondences by redefining the edge point generator density over several model points for each edge point and develop an efficient approximation of its most likely correspondence.

We model an edge point with no correspondence information as generated by one of several candidate model points, and assume that each model point generates at most one edge point. If we detect an edge point at the i^{th} pixel of an image, it is modeled as being generated by one of K_i projected model contour points m_k that are nearby. We simplify computing nearby point correspondences by linking points on the hypothesized model contour to their closest image edge point in the direction of the edge gradient. Creating this linkage based on the detected edge gradient instead of the model contour perpendicular has some the practical advantages, including being able to quickly find the candidate model points. This is accomplished as follows.

For each image edge point, we compute the distance along the edge gradient to points on the projected model contours. Under the assumption that a model point generates at most one edge point, we link a model point to its closest edge point using the computed distances. Each edge point will then have a disjoint set of model points it is linked with. Figure 5 illustrates a simple example



(a)

	m_1	m_2	m_3
x_1	δ_{11}	δ_{12}	δ_{13}
x_2	δ_{21}	δ_{22}	δ_{23}

(b)

	m_1	m_2	m_3
x_1	1	1	0
x_2	0	0	1

(c)

x_1	x_2
m_1	m_3
m_2	

(d)

Figure 5: Example point correspondence resolution linking three projected model contours (solid) with two edges (dashed) of an image object (shaded). For each edge point, a set of nearby model points in the gradient direction is found and used in the edge density function (24). All points in (a) are co-linear and parallel to the image gradients at x_1 and x_2. Distances between edge points and model points along the gradient are computed in (b); the shaded distances, $\delta_{12} < \delta_{11} < \delta_{23}$, are the smallest and labeled in (a). Model to edge point linkages are then made in (c) based on the closest edge point. Final linking of nearby model point sets to each edge point is summarized in (d).

of this process. The model point set can be empty, however, due to no points along the edge gradient or the distance being greater than a threshold. In this case, the edge point is not linked to any model points and is considered noise.

Given a set of K_i linked model points, we redefine the density for an edge point x_i in our generative image model. Since we do not know which of the model points actually generated the edge point, we average across their Gaussian response of eq. (20) with equal weights. The edge density function then becomes

$$\widetilde{e}_\theta(x_i) = \frac{1}{K_i} \sum_{k=1}^{K_i} \mathcal{N}\left(\widetilde{d}_{ik}; 0, \sigma_d\right) \mathcal{N}\left(\phi_{ik}; 0, \sigma_\phi\right) , \tag{24}$$

where the perpendicular distance $\widetilde{d}_{ik}$ from a model point is also redefined as

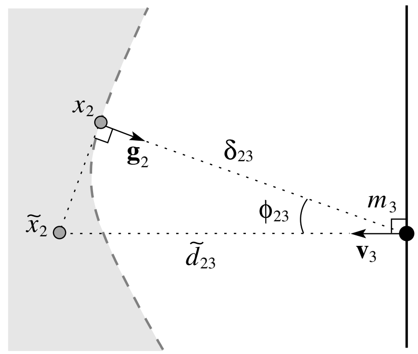


Figure 6: Distance and angle representation of $\widetilde{d}_{23}$ and ϕ_{23} for the edge point x_2 and model point m_3 in Figure 5. The point $\widetilde{x}_2$ is the generative approximation of x_2 that is perpendicular to both the projected model contour and gradient $\mathbf{g}_2$.

$$\widetilde{d}_{ik} \quad = \quad \| \, \widetilde{x}_i - m_k \, \| \, . \tag{25}$$

The point $\widetilde{x}_i$ is the generative approximation of x_i that is perpendicular to both the model contour at m_k and the gradient $\mathbf{g}_i$. If the edge point was found to be noise, however, due to no nearby model points, a constant minimum likelihood value, e_{noise}, is used instead. Figure 6 shows the details of this calculation for one of the example edge points in Figure 5.

In addition to redefining the edge point density using point correspondence estimation, we approximate the probability of not detecting an edge point at a particular pixel, eq. (23), with constants. We use a pair of probability constants for detection responses that are missing an edge point or are image background with no edge point expected. Pixels not containing an edge point, but that have the same image location as a projected model point, contribute a constant factor, e_{miss}, to the likelihood. However, only the $K_i - 1$ furthest model points from the i^{th} non-edge pixel contribute this constant; one of the model contours is assumed to have generated an actual edge point. Model points not linked to any edge point that have a detection response of no edge also contribute this constant. For all other pixel locations with no detected edge points, we factor in a constant background probability, e_{bg}.

We combine the approximations of the edge and no-edge density functions to redefine the likelihood for the edge point generator. Since detections not containing an edge point have constant probability, it is unnecessary to know which of the model points are missing an edge point. We only need to know how many there are, which we can easily compute by adding the number of model points not linked to an edge point with $\sum_{i=1}^{N_x}(K_i - 1)^{\mathcal{E}_i}$. This enables us to approximate the generator likelihood (19) with

$$\prod_{i=1}^{N_x} f_\theta(x_i) \approx \left\{ \prod_{i=1}^{N_x} \widetilde{e}_\theta(x_i)^{\mathcal{E}_i} \right\} \; e_{\text{bg}}^{N_{\text{bg}}} \; e_{\text{miss}}^{N_{\text{miss}}} \,. \tag{26}$$

where N_{bg} and N_{miss} are the number of background and missing detection responses in the image, and $N_{\text{bg}} + N_{\text{miss}} = \sum_{i=1}^{N_x} 1 - \mathcal{E}_i$.

Our approach has some similarities to standard edge matching (*e.g.*, [3, 14]), but we explain the edge points as the result of a generative statistical process that accounts for both distance and gradient direction. Using the Hausdorff distance for edges in our approach, for example, would preclude our ability to link edge points to projected model contours for likelihood computation, since no correspondences would be computed.

While the assumption that a model point can be assigned to at most one image edge point may seem arbitrary, we have experimented with other assignment alternatives and found it to give the best results. We have also experimented with different weightings in the average computed over model points in eq. (24) and found uniform weights to work the best in the most cases.

Implementation detail: Much of the model and edge point linkage is easily precomputed at program initialization for the input images. After detecting the edge points in each image, we create a correspondence grid of potential model point distances and gradient angles with the same dimensions and indexing as the image. The kth index in the grid stores the computed d_{ik} and discretized set of ϕ_{ik} for each of the edge points x_i whose gradient traces through the kth point in the grid. A discretized set of ϕ_{ik} is computed because the orientation of the model contour is not yet known. During learning and recognition, when the model contours are computed and projected, we look-up the distance and gradient angle for each of the model points in the precomputed table.

Surface point generator

Surface points are part of the projected object model and occur at image pixels that are part of the foreground. We model these detected foreground pixels as the projected points of viewable surfaces in our object model. Foreground pixels are detected in an image by applying the k-means algorithm on pixel intensities, where $k = 2$. In most cases this gives a good estimate of the foreground because the objects are already set on an almost white background. Figure 4 shows an example foreground detection for an image.

Similar to the edge point generator, the surface detector gives a response at each pixel location. We also have density functions for surface and non-surface points. Thus, we define the surface generator likelihood as

$$\prod_{i=1}^{N_x} f_{\theta g}^{(n)}(x_{gi}) = \prod_{i=1}^{N_x} s_\theta(x_i)^{\mathcal{S}_i} \cdot s'_\theta(x_i)^{(1-\mathcal{S}_i)}. \tag{27}$$

The per pixel indicator $\mathcal{S}_i$ is 1 if the pixel contains a detected surface point in the foreground, otherwise it is 0 and considered part of the background.

We define the density functions in terms of constant likelihoods for surface and non-surface points. The decision for what type of constant to use is based on comparing the surface point detection response at a pixel in the observed image and the corresponding projected object model surface point in the same pixel location of a hypothesized model image.

We define the density function for detected surface points with two constants for foreground and noise. If the pixel contains a detected surface point and shares a location with a projected model surface point, then we say it is part of the foreground and contributes s_{fg}. If the detected surface point has no projected model surface point over it, we label it as noise and factor in s_{noise}.

We define the density function for detected non-surface points also with two constants, but for background and missing points. If the pixel does not contain detected surface point and has no projected model surface points in the hypothesized image, then we say it is part of the background and contributes s_{bg}. If the pixel again does not contain a detected surface point, but has a projected model surface point over it in the hypothesized image, we label it as missing a surface point in the observed image with factor in s_{miss}. Thus, the surface point generator likelihood expands to

$$\prod_{i=1}^{N_x} f_\theta(x_i) = s_{\text{fg}}^{N_{\text{fg}}}\; s_{\text{bg}}^{N_{\text{bg}}}\; s_{\text{noise}}^{N_{\text{noise}}}\; s_{\text{miss}}^{N_{\text{miss}}}\,. \tag{28}$$

where $N_{\text{fg}} + N_{\text{bg}} + N_{\text{noise}} + N_{\text{miss}} = N_x$.

3. Inference

We sample the posterior to find the best set of parameters that fit a set of images. Given enough iterations, a good sampler converges to the target distribution and an optimal value would be readily discovered in the process. However, our posterior distribution is highly convoluted with many sharp, narrow ridges for close fits to the edge points and foreground. In our domain, as in many similar problems, standard sampling techniques tend to get trapped in these local extrema for long periods of time. Our strategy is to combine a mixture of sampling techniques with different strengths in exploring the posterior distribution.

Our sampling space is over all camera parameters, internal object parameters and object topologies

$$\Omega = \bigcup_{\mathbf{n} \in \mathbb{N}^3} \mathsf{C}^D \times \mathsf{S}^{(n_1,n_2)D} \times \mathsf{T}^{(n_3)}\,, \tag{29}$$

such that $\boldsymbol{\theta}^{(\mathbf{n})} \in \Omega$.

From a set of images, we formulate a posterior from the image likehood model and a prior probability distribution,

$$p\left(\boldsymbol{\theta}^{(\mathbf{n})} \,|\, \mathbf{X}\right) = k_p\, L\left(\mathbf{X} \,|\, \boldsymbol{\theta}^{(\mathbf{n})}\right)\, p\left(\boldsymbol{\theta}^{(\mathbf{n})}\right)\,. \tag{30}$$

Conditioned on the object topology, the structure and camera paramters are indepenent for each image,

$$p\left(\boldsymbol{\theta}^{(\mathbf{n})}\right) = \prod_{d=1}^{D} p(\mathbf{c}_d,\, \mathbf{s}_d \,|\mathbf{t})\, p(\mathbf{t})\,. \tag{31}$$

Each of the parameters within the camera and structure components are modeled according to a

Gaussian distribution with large variance. The topology pieces are uniformly distributed.

3.1. Trans-dimensional sampling

The Metroplis-Hastings (MH) algorithm is an MCMC sampling technique to generate unbiased and representative samples from a target distribution [21, 10, 23, 7, 2]. The central concept of the algorithm is to propose samples from a distribution $q(\theta' \mid \theta)$, which can be easily sampled, and accept or reject the samples with probability

$$\alpha\left(\tilde{\boldsymbol{\theta}}^{(\mathbf{n})}\right) = \min\left\{1,\ \frac{p(\tilde{\boldsymbol{\theta}}^{(\mathbf{n})} \mid \mathbf{X})\, q(\boldsymbol{\theta}^{(\mathbf{n})} \mid \tilde{\boldsymbol{\theta}}^{(\mathbf{n})})}{p(\boldsymbol{\theta}^{(\mathbf{n})} \mid \mathbf{X})\, q(\tilde{\boldsymbol{\theta}}^{(\mathbf{n})} \mid \boldsymbol{\theta}^{(\mathbf{n})})}\right\} \tag{32}$$

Trans-dimensional changes in topology

$$\alpha\left(\tilde{\boldsymbol{\theta}}^{(\mathbf{n+m})}\right) = \min\left\{1,\ \frac{p(\tilde{\boldsymbol{\theta}}^{(\mathbf{n+m})} \mid \mathbf{X})}{p(\boldsymbol{\theta}^{(\mathbf{n})} \mid \mathbf{X})\, q^{(\mathbf{m})}(\tilde{\mathbf{b}},\, \tilde{\mathbf{t}})} \frac{r_d}{r_b} \left|\frac{\partial(\tilde{\boldsymbol{\theta}}^{(\mathbf{n+m})})}{\partial(\boldsymbol{\theta}^{(\mathbf{n})},\, \tilde{\mathbf{b}},\, \tilde{\mathbf{t}})}\right|\right\} \tag{33}$$

Since the sampler follows a Markov chain, and it maintains the detailed balance condition
it is sufficient that the sampler will have as its invariant the posterior, assuming there are no zero probability transitions.

3.2. Stochastic Dynamics

We can use the negative log of the joint density over parameters and data to define a potential energy function, which will be convenient during inference,

$$E^{(\mathbf{n})}(\boldsymbol{\theta}) = -\log L\left(\mathbf{X} \mid \boldsymbol{\theta}^{(\mathbf{n})}\right) - \log p\left(\boldsymbol{\theta}^{(\mathbf{n})}\right) \tag{34}$$

Define a kinetic energy function over the introduced momentum $K^{(\mathbf{n})}(\mathbf{r}) = \frac{1}{2}\sum_{i=1}^{N_x^{(\mathbf{n})}} r_i^2$. The total energy in phase space is the Hamiltonian

$$H^{(\mathbf{n})}(\boldsymbol{\theta},\,\mathbf{r}) = E^{(\mathbf{n})}(\boldsymbol{\theta}) + K^{(\mathbf{n})}(\mathbf{r}) \tag{35}$$

The canonical distrubution over phase space is given by

$$p^{(\mathbf{n})}(\boldsymbol{\theta},\,\mathbf{r}) = \frac{1}{Z_H}\exp\left(-H(\boldsymbol{\theta},\,\mathbf{r})\right) \tag{36}$$

We'll sample from the marginal distribution by sampling from the canonical and just ignoring the momentum.

We'll sample by following the Hamiltonian dynamics over time τ

$$\frac{d\theta_i}{d\tau} = r_i\,, \quad \frac{dr_i}{d\tau} = -\frac{\partial E}{\partial \theta_i} \tag{37}$$

Total energy is conserved as we move through phase space. Dynamics additionally preserves volume in phase space (Liouville's theorem). Combining these two results, the invariant of the dynamics over time is the canonical distrubution.

We discretize the dynamics with the standard leapfrog algorithm, where we update the momenta and position each after every halfstep discretized time step. We transition from $\boldsymbol{\theta}, \mathbf{r}$ at time step τ to $\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{r}}$ at time step $\tau + \epsilon$ by doing a half update to the momenta $r_i(\tau + \epsilon/2)$, and full update to the position variable $\theta_i(\tau + \epsilon)$ and another half update of the momenta $r_i(\tau + \epsilon)$, where

$$\begin{aligned}\tilde{r}_i(\tau + \epsilon/2) &= r_i(\tau) - \frac{\epsilon}{2}\frac{\partial E(\theta_i(\tau))}{\partial \theta_i} && (38)\\ \tilde{\theta}_i(\tau + \epsilon) &= \theta_i(\tau) + \epsilon\,\tilde{r}_i(\tau + \epsilon/2) && (39)\end{aligned}$$

Introduce stochastic transitions to have the sampler ergodically sample from the canonical distrubution and transition to states of different total energy. We use the following stochastic transitions

$$\tilde{r}_i = \alpha r_i + (1 - \alpha^2)^{1/2}\,\eta_i \tag{40}$$

4. Results

We evaluated our sampling strategy and models by inferring them on a set of 32 table images. The edges in all the images were detected with the same parametrization, resulting in many edge points that could be considered noise, or in some cases, missing from major portions of table structure. The fitting process for each image was initialized from a different random state drawn from our fairly uninformative prior.

We cycled through each of the Langevin, covariance scaled Metropolis-Hastings and hyperdynamics samplers five times, with most obtaining a good fit after just a couple of cycles. The Langevin and Metropolis-Hastings samplers were run for 10K iterations during each cycle, this was followed by 50 iterations of hyperdynamics. The hyperdynamics sampler takes a significantly longer time for sample generation due to the complex numerical approximations that must be calculated for the bias function. However, 50 iterations was often enough to position the model parameters at a transition point so a new state could be reached.

As shown in Fig. 7, we accurately fit most of the table and camera models to the images. The same for Fig. 8. If we continue to run the sampler, we believe that good fit would eventually be found for all. The image in the top left corner of Fig. 7 is particularly interesting because of the poor edge detection that occurred. We observe from this fit that even though a substantial number of edge points are noise, nearly half the table surface edge points are missing, and the back leg has no detection at all, we are still able to make a somewhat accurate fit to this image.

As we stated in the introduction, our overall goal is to learn the form of general 3D structure models for objects. We believe that the novel inference process we have presented in this paper is a good first step on a trajectory to learning structure models. To some extent we can already reason about the structure of tables; we have fit a highly configurable model and can now consider the statistics of such a model after fitting it to a collection of images. We plan to do the same for other objects, in a less specified way, to learn how to discriminate between various classes of structures that share parts and appear similar.

Figure 7: Learning the topology of simple furniture objects. Sets of contiguous blocks were fit across two small data sets—one consisting of eight tables and one consisting of eight chairs. Model fitting is done jointly for the eight images of each set. The topology (how many blocks there are and how they connect) was constrained to be the same over all exemplars in a given set whereas the camera parameters and the instance parameters (block position and size) were modulated across the exemplars. The location of the edge points in the image is only softly fit to the model edges (shown in red) to account for the deformation from a parallelepiped (note the legs of the upper right table). While the quality of individual fits naturally varies across examples (these are relatively good results), the main point is that in both cases the system learned, from the same starting point, a serviceable topology for the category represented by the collections of instances.

Figure 8: Learned topology and instance parameters for a chair object.

5. Conclusion

The main goal of this work was to develop an approach for learning strong 3D models, with unknown topologies, from single 2D images. In particular, we are working with models that represent objects as 3D assemblages of sub-structures that we assume to be effective way to represent a range of objects for a variety of vision task. A key technical challenge for learning such models from single 2D views was streamlining the inference so that model hypotheses can be explored relatively quickly. A second key challenge addressed in this work was to arrange an image model that statistically explains every image pixel to effectively mitigate against biases for more or less complex models, and explaining more or less of the image. Dealing with these two challenges allowed viable topologies to emerge that are consistent with multiple images of objects from the same category.

We have developed the approach in a relatively simple domain, but the methods can be extended to more general configurations (relaxing the right angle assumption), and a larger palate of more deformable parts. Further it will be relatively easy to take structure learned for a category, and then split it into clusters, allowing different related topologies to be learned to encode different ways of being a particular object (e.g., chairs with and without backs). Finally, learning more about the statistics of the individual parts in a particular topology will be helpful, especially as we expand the number of categories.

References

[1] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003. 3

[2] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. 22

[3] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988. 19

[4] D. Crandall and D. Huttenlocher. Weakly-supervised learning of part-based spatial models for visual object recognition. In *9th European Conference on Computer Vision*, 2006. 4

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*, 2004. 4

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003. 4

[7] D. A. Forsyth, J. Haddon, and S. Ioffe. The joy of sampling. *International Journal of Computer Vision*, 41(1-2):109–134, January 2001. 22

[8] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995. 3

[9] P. J. Green. Trans-dimensional markov chain monte carlo. In *Highly Structured Stochastic Systems*. 2003. 3

[10] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970. 3, 22

[11] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006. 4

[12] D. Hoiem, C. Rother, and J. Winn. 3d layoutcrf for multi-view object class recognition and segmentation. In *CVPR*, 2007. 4

[13] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(2):195–212, 1990. 4

[14] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9), 1993. 19

[15] C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008. 4

[16] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. In *CVPR*, 2007. 4

[17] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007. 4

[18] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, 2001. 3

[19] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991. 4

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoint. *International Journal of Computer Vision*, 60(2):91–110, 2004. 4

[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953. 3, 22

[22] G. Mori and J. Malik. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 4

[23] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993. 3, 22

[24] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *IEEE Intern. Conf. in Computer Vision (ICCV)*, 2007. 4

[25] S. Savarese and L. Fei-Fei. View synthesis for recognizing unseen poses of object classes. In *European Conference on Computer Vision (ECCV)*, 2008. 4

[26] C. Sminchisescu. Kinematic jump processes for monocular 3d human tracking. In *Computer vision and pattern recognition*, 2003. 4

[27] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003. 4

[28] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, 2005. 4

[29] J. B. Tenenbaum, T. L. Griffiths, and C. Kemp. Theory-based bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318, 2006. 4

[30] L. Zhu, Y. Chen, and A. Yuille. Unsupervised learning of a probabilistic grammar for object detection and parsing. In *NIPS*, 2006. 4

[31] S. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 4(2):259–362, 2006. 4