Picture: A Probabilistic Programming Language for Scene Perception

Tejas D Kulkarni
MIT
tejask@mit.edu

Pushmeet Kohli
Microsoft Research
pkohli@microsoft.com

Joshua B Tenenbaum
MIT
jbt@mit.edu

Vikash Mansinghka
MIT
vkm@mit.edu

Abstract

Recent progress on probabilistic modeling and statistical learning, coupled with the availability of large training datasets, has led to remarkable progress in computer vision. Generative probabilistic models, or "analysis-by-synthesis" approaches, can capture rich scene structure but have been less widely applied than their discriminative counterparts, as they often require considerable problem-specific engineering in modeling and inference, and inference is typically seen as requiring slow, hypothesize-and-test Monte Carlo methods. Here we present Picture*, a probabilistic programming language for scene understanding that allows researchers to express complex generative vision models, while automatically solving them using fast general-purpose inference machinery.* Picture *provides a stochastic scene language that can express generative models for arbitrary 2D/3D scenes, as well as a hierarchy of representation layers for comparing scene hypotheses with observed images by matching not simply pixels, but also more abstract features (e.g., contours, deep neural network activations). Inference can flexibly integrate advanced Monte Carlo strategies with fast bottom-up data-driven methods. Thus both representations and inference strategies can build directly on progress in discriminatively trained systems to make generative vision more robust and efficient. We use* Picture *to write programs for generative 3D face analysis, 3D human pose estimation, and 3D object reconstruction – each in under 50 lines of probabilistic code, and each competitive with specially engineered baselines.*

1. Introduction

Probabilistic scene understanding systems aim to produce high-probability descriptions of scenes conditioned on observed images or videos, typically either via discriminatively trained models or generative models in an "analysis by synthesis" framework. Discriminative approaches lend themselves to fast, bottom-up inference methods and relatively knowledge-free, data-intensive training regimes, and have been remarkably successful on many recognition problems [9, 24, 27, 31]. Generative approaches hold out the promise of analyzing complex scenes more richly and flexibly [11, 12, 51, 7, 19, 30, 32, 16, 21], but have been less widely embraced for two main reasons: Inference typically depends on slower forms of approximate inference, and both model-building and inference can involve considerable problem-specific engineering to obtain robust and reliable results. These factors make it difficult to develop simple variations on state-of-the-art models, to thoroughly explore the many possible combinations of modeling, representation, and inference strategies, or to richly integrate complementary discriminative and generative modeling approaches to the same problem. More generally, to handle increasingly realistic scenes, generative approaches will have to scale not just with respect to data size but also with respect to model and scene complexity. This scaling will arguably require general-purpose frameworks to compose, extend and automatically perform inference in complex structured generative models – tools that for the most part do not yet exist.

Here we present *Picture*, a probabilistic programming language that aims to provide a common representation language and inference engine suitable for a broad class of generative scene perception problems. We see probabilistic programming as key to realizing the promise of "vision as inverse graphics". Generative models can be represented via stochastic code that samples hypothesized scenes and generates images given those scenes. Rich deterministic and stochastic data structures can express complex 3D scenes that are difficult to manually specify. Multiple representation and inference strategies are specifically designed to address the main perceived limitations of generative approaches to vision. Instead of requiring photo-realistic generative models with pixel-level matching to images, we can compare hypothesized scenes to observations using a hierarchy of more abstract image representations such as contours, discriminatively trained part-based skeletons, or deep neural network features. Top-down Monte Carlo inference algorithms include not only traditional Metropolis-Hastings, but also more advanced techniques for inference in high-dimensional continuous spaces, such as elliptical slice sampling, and Hamiltonian Monte Carlo which can exploit the gradients of automatically differentiable renderers. These top-down inference approaches are integrated with bottom-up and automatically constructed data-driven proposals, which can

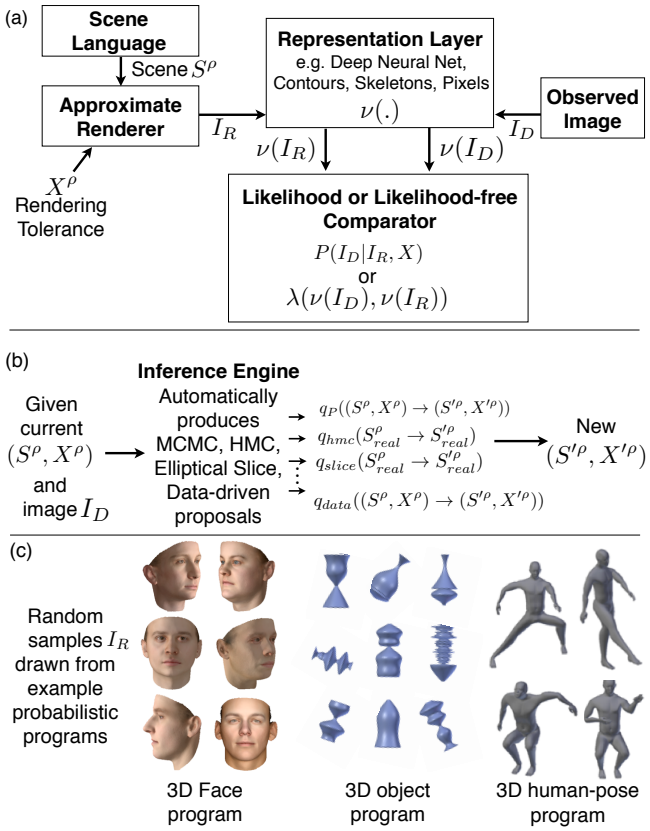


Figure 1: **Overview: (a)** All models share a common template; only the scene description S and image I_D changes across problems. Every probabilistic program f defines a stochastic procedure that generates both a scene description and all the other information needed to render an approximation I_R of any given observed scene image I_D. The program f induces a joint probability distribution on the program trace ρ. *Picture* has the following key components – **Scene Language:** Describes 2D/3D scenes and generates particular scene related trace variables $S^\rho \in \rho$ during execution. **Approximate Renderer:** Produces graphics rendering I_R given S^ρ and latents X^ρ for controlling the fidelity or tolerance of rendering. **Representation Layer:** Transforms I_D or I_R into a hierarchy of coarse-to-fine image representations $\nu(I_D)$ and $\nu(I_R)$ (deep neural networks [26, 24], contours [8] and pixels). **Comparator:** During inference, I_R and I_D can be compared using a likelihood function or a distance metric (approximate bayesian computation [46]). **(b) Inference Engine:** Automatically produces a variety of proposals and iteratively evolves the scene to reach a high probability state given I_D. **(c) Illustrative Samples:** Random draws of the scene from a variety of probabilistic programs.

dramatically accelerate inference by eliminating most of the "burn in" time of traditional samplers and enabling rapid mode-switching.

We demonstrate *Picture* on three challenging vision problems: inferring the 3D shape and detailed appearance of faces, the 3D pose of articulated human bodies, and the 3D shape of medially-symmetric objects. Perhaps surprisingly each problem requires less than 50 lines of problem-specific probabilistic code; the vast majority of code for image modeling and inference is reusable across these and many other tasks. We shows that *Picture* yields competitive performance with optimized baselines on each of these benchmark tasks.

```
function PROGRAM(MU, PC, EV, VERTEX_ORDER)
  # Scene Language: Stochastic Scene Gen
  face=Dict();shape = []; texture = [];
  for S in ["shape", "texture"]
   for p in ["nose", "eyes", "outline", "lips"]
    coeff = MvNormal(0,1,1,99)
    face[S][p] = MU[S][p]+PC[S][p].*(coeff.*EV[S][p])
   end
  end
  shape=face["shape"][:]; tex=face["texture"][:];
  camera = Uniform(-1,1,1,2); light = Uniform(-1,1,1,2)

  # Approximate Renderer
  rendered_img= MeshRenderer(shape,tex,light,camera)

  # Representation Layer
  ren_ftrs = getFeatures("CNN_Conv6", rendered_img)

  # Comparator
  #Using Pixel as Summary Statistics
  observe(MvNormal(0,0.01), rendered_img-obs_img)
  #Using CNN last conv layer as Summary Statistics
  observe(MvNormal(0,10), ren_ftrs-obs_cnn)
end

global obs_img = imread("test.png")
global obs_cnn = getFeatures("CNN_Conv6", img)
#Load args from file
TR = trace(PROGRAM,args=[MU,PC,EV,VERTEX_ORDER])
# Data-Driven Learning
learn_datadriven_proposals(TR,100000,"CNN_Conv6")
load_proposals(TR)
# Inference
infer(TR,CB,20,["DATA-DRIVEN"])
infer(TR,CB,200,["ELLIPTICAL"])
```

Figure 2: *Picture* code illustration for 3D face analysis: Modules from figure 1a,b are highlighted in **bold**. Running the program unconditionally (by removing *observe's* in code) produces random faces as shown in Figure 1c. Running the program conditionally (keeping *observe's*) on I_D results in posterior inference as shown in Figure 3. The variables **MU**, **PC**, **EV** correspond to the mean shape/texture face, principal components, and eigen vectors respectively (see [38] for details). These arguments parametrize the prior on the learnt shape and appearance of 3D faces. The argument **VERTEX_ORDER** denotes the ordered list of vertices to render triangle based meshes. The *observe* directive constrains the program execution based on both the pixel data and CNN features. The *infer* directive starts the inference engine with the specified set of inference schemes (takes the program trace, a callback function CB for debugging, number of iterations and inference schemes). In this example, data-driven proposals are ran for a few iterations to initialize the sampler, followed by slice sampling moves to further refine the high dimensional scene latents.

2. *Picture* Language

Picture descends from our earlier work on generative probabilistic graphics programming (GPGP) [32], and also incorporates insights for inference from the Helmholtz machine [17, 6] and recent work on differentiable renderers [30]

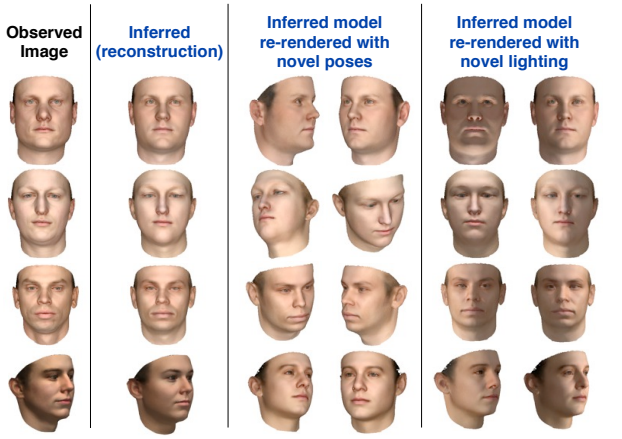


Figure 3: **Inference on representative faces using *Picture*:** We tested our approach on a held-out dataset of 2D image projections of laser-scanned data from [38]. Our short probabilistic program is applicable to non-frontal faces and provides reasonable parses as illustrated above just using general-purpose inference machinery. For quantitative metrics, refer to section 4.1.

and informed samplers [19]. GPGP aimed to address the main challenges of generative vision by representing visual scenes as short probabilistic programs with random variables, and using a generic Monte Carlo (single-site Metropolis-Hastings) method for inference. However, due to modeling limitations of earlier probabilistic programming languages, and the inefficiency of the Metropolis-Hastings sampler, GPGP was limited to working with low-dimensional scenes, restricted shapes, and low levels of appearance variability. Moreover, it did not support the integration of bottom-up discriminative models such as deep neural networks [24, 26] for data-driven proposal learning. Our current work extends the GPGP framework in all of these directions, letting us tackle a richer set of real-world 3D vision problems.

Picture is an imperative programming language, where expressions could take on either deterministic or stochastic values. A generative program implicitly induces a joint probability distribution. A *Picture* program f defines a stochastic procedure that generates both a scene description and all the other information needed to render an approximation I_R given observed scene image I_D. The program f induces a joint probability distribution on the program trace $\rho = \{\rho_i\}$, where each random choice ρ_i can belong to parametric or non-parametric family of distributions such as:{*Multinomial, MvNormal, DiscreteUniform, Uniform, Poisson, Gaussian_Process, etc.*}. We use the transformational compilation technique [47] to implement *Picture*, which is a general method of transforming arbitrary programming languages into probabilistic programming languages. Compared to earlier formulations of GPGP, *Picture* is dynamically compiled at run-time (JIT-compilation) instead of interpreting, making program execution fast.

Consider running the program in figure 2 unconditionally (without input data): as different ρ_i's are encountered (for eg. *coeff*), random values are sampled and cached in ρ_i with respect to its underlying probability distribution. As program execution is finished, the output is an image of a face with random shape, texture, camera and lighting parameters. Given image data I_D, inference in *Picture* programs amounts to iteratively sampling or evolving program trace ρ to a high probability state while respecting constraints imposed by the data. This constrained simulation can be achieved by using the $observe$ language construct (see figure 2 for code and 3 for results) first proposed in Venture [33] and also used in [37, 48].

2.1. Architecture

In this section, we will explain the essential architectural components highlighted in figure 1 (see figure 4 for summary of notation used).

Scene Language: The scene language is used to describe 2D/3D visual scenes as probabilistic code. Visual scenes can be built out of several graphics primitives such as: description of 3D objects in the scene (e.g. mesh, z-map, volumetric), one or more lights, textures, and the camera information. It is important to note that scenes expressed as probabilistic code are more general than parametric prior density functions as is typical in generative vision models. The probabilistic programs we demonstrate in this paper embed ideas from computer-aided design (CAD) and non-parametric Bayesian statistics[39] to express variability in 3D shapes.

Approximate Renderer (AR): *Picture*'s AR layer takes in a scene representation trace S^ρ and tolerance variables X^ρ, and uses general-purpose graphics simulators (Blender[5] and OpenGL) to express 3D scenes. The rendering tolerance X^ρ essentially defines a structured noise process over the rendering and is useful for the following purposes: (a) analogous to simulated annealing, adding X^ρ can sometimes make automatic inference tractable or more robust (e.g. global or local blur variables in GPGP [32]), and (b) to soak up model mismatch between the true scene rendering I_D and the hypothesized rendering I_R. Moreover, inspired from Loper et al.[30], *Picture* also support expressing AR's entire graphics pipeline as *Picture* code, enabling the language to express end-to-end differentiable generative models.

Representation Layer (RL): To avoid having to reason about the rich appearance variability of complex 3D scenes in terms of photo-realistic renderings in raw pixel space, which can be slow and sometimes may lead to sampler intractability, RL is designed to render the scene in abstract feature space. RL can be defined as a function ν which produces summary statistics given I_D or I_R, and may also have internal parameters θ_ν (e.g. weights of a deep neural net). For notational convenience, we denote $\nu(I_D; \theta_\nu)$ and $\nu(I_D; \theta_\nu)$ to be $\nu(I_D)$ and $\nu(I_R)$ respectively. RL produces summary statistics (features) that are used in two scenarios:

Modules	Functional Description
Scene Representation S:	

```
light_source { <0, 199, 20>
               color rgb<1.5,1.5,1.5> }
camera { location <30,48,-10> angle 40
         look_at <30,44,50> }

object{leg-right vertices ...
       trans  <32.7,43.6,9>}
object{arm-left vertices scale 0.2
       ... rotate x*0}
...
object{arm-left texture}
```

Program trace:	$\rho = \{\rho_i\}$
Rendering tolerance:	$X^\rho \in \rho$
Stochastic Scene:	$S^\rho \in \rho$
Approximate Rendering:	I_R
Approximate Renderer:	*render* : $(S, X) \rightarrow I_R$
Image data:	I_D
Data-driven Proposals:	$(f, T, \nu_{dd}, \theta_{\nu_{dd}}) \rightarrow q_{data}(.)$
Data representations:	$\nu(I_D)$ and $\nu(I_R)$
Comparator:	$\lambda : (\nu(I_D), \nu(I_R)) \rightarrow \mathbb{R}$ $P(\nu(I_D)\|\nu(I_R), X)$
Rendering Differentiator:	$\nabla_{S^\rho_{real}} : \rho \rightarrow$ $grad\_model\_density(S_{real}; I_D)$

Figure 4: **Formal Summary:** The scene S can be conceptualized as a program that describes the structure of known or unknown number of objects, texture-maps, lighting and other scene variables. The symbol T denotes the number of times the program f is executed to generate data-driven proposals (see section 3.2 for details). The rendering differentiator produces gradients of the program density with respect to continuous variables S_{real} in the program.

(a) to compare the hypothesis I_R with observed image I_D during inference (RL denoted by ν in this setting), and (b) as a dimensionality reduction technique for data-driven proposal learning (RL denoted by ν_{dd} and its parameters $\theta_{\nu_{dd}}$). *Picture* supports a variety of summary statistic functions including raw pixels, contours [8] and supervised/unsupervised convolutional neural network (CNN) architectures[24, 26].

Likelihood and Likelihood-free Comparator: *Picture* supports likelihood $P(I_D|I_R)$ inference in a bayesian setting. However, in the presence of black-box rendering simulators, the likelihood function is often unavailable in closed form. Given an arbitrary distance function $\lambda(\nu(I_D), \nu(I_R))$ (e.g. L1 error), approximate bayesian computation (ABC) [46] can be used to perform likelihood-free inference.

3. Inference

Automatic inference of *Picture* programs is especially hard due to the following reasons: highly coupled variables, mix of discrete and continuous variables, many local minima's due to clutter, occlusion and noisy observations. We can formulate the task of 3D image interpretation as approximately sampling mostly likely values of S^ρ given observed image I_D (LL stands for $P(I_D|I_R, X^\rho)$):

$$P(S^\rho|I_D) \propto \int P(S^\rho)P(X^\rho)\delta_{render(S^\rho,X^\rho)}(I_R)\, LL\, dX^\rho$$

Given a program trace ρ, probabilistic inference amounts to updating (S^ρ, X^ρ) to (S'^ρ, X'^ρ) until convergence via proposal kernels $q((S^\rho, X^\rho) \rightarrow q(S'^\rho, X'^\rho))$. Let $K = |\{S^\rho\}| + |\{X^\rho\}|$ and $K' = |\{S'^\rho\}| + |\{X'^\rho\}|$ be the total number of random choices in the execution before and after applying the proposal kernels $q(.)$. Let the log-likelihoods of old and new trace be $LL = P(I_D|I_R, X)$ and $LL' = P(I_D|I'_R, X')$ respectively. Let us denote the probabilities of deleted and newly created random choices created in S^ρ to be $P(S^\rho_{del})$ and $P(S^\rho_{new})$ respectively. Let $q_{(S',X')\rightarrow(S,X)} := q((S'^\rho, X'^\rho) \rightarrow (S^\rho, X^\rho))$ and $q_{(S,X)\rightarrow(S',X')} := q((S^\rho, X^\rho) \rightarrow (S'^\rho, X'^\rho))$. The new trace (S'^ρ, X'^ρ) can now be accepted or rejected using the acceptance ratio:

$$min\Big(1, \frac{LL'\ P(S'^\rho)P(X'^\rho)\ q_{(S',X')\rightarrow(S,X)}\ K\ P(S^\rho_{del})}{LL\ P(S^\rho)P(X^\rho)\ q_{(S,X)\rightarrow(S',X')}\ K'\ P(S^\rho_{new})}\Big). \quad (1)$$

3.1. Distance Metrics and Likelihood-free Inference

The likelihood function in closed form is often unavailable when integrating top-down automatic inference with bottom-up computational elements. Moreover, this issue is exacerbated when programs use black-box rendering simulators. Approximate bayesian computation (ABC) allows bayesian inference in likelihood-free settings, where the basic idea is to use a summary statistic function $\nu(.)$, distance metric $\lambda(\nu(I_D), \nu(I_R))$ and tolerance variable X^ρ to approximately sample the posterior distribution [46].

Inference in likelihood-free settings can also be interpreted as a variant of the probabilistic approximate MCMC algorithm [46], which is similar to MCMC but with an additional tolerance parameter $\epsilon \in X^\rho$ on the observation model. We can interpret our approach as systematically reasoning about the model error arising due to the difference of generative model's "cartoon" view of the world with reality. Let Θ to be space of all possible renderings I_R that could be hypothesized and P_ϵ be the error model (such as gaussian). The target stationary distribution that we wish to sample can be expressed as:

$$P(S^\rho|I_D) \propto \int_\Theta P(S^\rho)P_\epsilon(\nu(I_D)-\nu(I_R))P(\nu(I_R)|S^\rho)dI_R.$$

During inference, the updated scene S'^ρ (assuming random choices remain unchanged, otherwise add terms relating to addition/deletion of random variables as in equation 1) can then be accepted with probability:

$$min\left(1, \frac{P_\epsilon(\nu(I_D) - \nu(I'_R))P(S'^\rho)P(X'^\rho)\ q_{(S',X')\rightarrow(S,X)}}{P_\epsilon(\nu(I_D) - \nu(I_R))P(S^\rho)P(X^\rho)\ q_{(S,X)\rightarrow(S',X')}}\right).$$

3.2. Proposal Kernels

In this section, we will propose a variety of proposal kernels for scaling up *Picture* to complex 3D scenes.

Local and Blocked Proposals from Prior: Single site metropolis hastings moves on continuous variables and Gibbs moves on discrete variables can be useful in many cases. However, 3D affine latent variables such as the rotation and translation are almost always coupled. Therefore our inference library also support users to easily define arbitrary blocked proposals: $q_P((S^\rho, X^\rho) \rightarrow (S'^\rho, X'^\rho)) = \prod_{\rho'_i \in (S^\rho, X^\rho)} P(\rho'_i)$

Gradient Proposal: *Picture* inference supports automatic differentiation for a restricted class of programs (each expression should be able to provide output and gradients w.r.t input). Therefore it is fairly straight forward to obtain $\nabla_{S_{real}} \rho$ using reverse mode automatic differentiation, where $S_{real} \in S^\rho$ denotes all continuous variables. This enables us to automatically construct Hamiltonian Monte Carlo proposals[35] $q_{hmc}(S^\rho_{real} \rightarrow S'^\rho_{real})$ (see supplementary material for a toy example).

Slice Proposals: Slice sampling can be efficiently used to propose continuous random variables in a trace. To propose a large set of latent variables at once, our inference library supports elliptical moves with or without adaptive step sizes (see figure 2 for an example)[4, 34]. Assuming $S_{real} \sim \mathcal{N}(0, \Sigma)$. We can generate a new sub-trace S'_{real} efficiently as follows: $S'_{real} = \sqrt{1-\alpha^2} S_{real} + \alpha\theta$, where $\theta \sim \mathcal{N}(0, \Sigma)$ and $\alpha \sim Uniform(-1, 1)$. Furthermore, adaptive tuning of the step-size α is implemented using an elliptical slice sampler [34] on the trace space.

Data-driven Proposals: Top-down inference in generative probabilistic programs has been thought to be slow due to poor mixing and long "burn-in" time. However, these issues can be mitigated by integrating top-down inference with bottom-up methods to automatically construct data-driven proposals. Such techniques fall under the broader idea of amortizing or caching inference to improve its speed and accuracy [42, 19, 45]. Our approach for learning data-driven proposals is inspired by and similar to the *informed sampler* [19] idea, which produces discriminative data-driven proposals given generative models. In general, there are several ways to obtain data-driven proposals for probabilistic programs: (a) training or using discriminative approaches to learn bottom-up regressors or proposals to predict all or a subset of latent variables [45, 42], (b) a broader approach which we term as *Helmholtz proposals* that encapsulates frameworks such as the *informed sampler* and *Helmholtz machines* [17, 6].

Our main focus is to explore the idea of *Helmholtz proposals*, which can be summarized as followed: (a) hallucinate samples from an internal generative model, (b) store the hallucinated latents and corresponding data in memory, (c) refine the bottom-up predictor model, and finally (d) produce data-driven proposals using the bottom-up trained model. Under this framework, the data-driven proposals q_{data} can be calculated as follows:

(1) Specify the number of times T to forward simulate (unconditional runs) the given program f, which induces a distribution over program trace ρ. Create a database $\mathcal{C}$.
(2) During training, this module then forward executes f for T times to create program traces ρ^t and approximate rendering I^t_R, where $\{1 \leq t \leq T\}$.
(3) Specify a summary statistic function ν_{dd} with model parameters $\theta_{\nu_{dd}}$. The desiderata for a "good" ν_{dd} can be summarized as follows: (a) function should produce compressed representations of I_D and I_R with feature dimensionality being significantly smaller than input size to enable proposal learning on massive datasets, (b) for robustness the function should preserve as many transformations in the data as possible (not invariant) and (c) the function should project I_R and I_D close to each other in the feature space if they have similar semantic content.
(4) Fine-tune parameters $\theta_{\nu_{dd}}$ of the representation layer ν_{dd} using supervised learning by passing $\{I^t_R\}^T_{t=1}$ and predicting $\{\rho^t\}^T_{t=1}$. If labeled data is available for full or partial trace $\{S^\rho_p\}$ and corresponding observed image $\{I_D\}$, parameters $\theta_{\nu_{dd}}$ can be fine-tuned using supervised learning to further improve ν_{dd}.
(5) Define a hash function $H : \nu(I^t_R) \rightarrow h^t$, where h^t denotes the hash value for $\nu(I^t_R)$. For instance, H can be defined as K-nearest neighbor or Dirichlet-Process mixture model. During training, we store triplets $\{\rho^t, \nu(I^t_R), h^t\}$ in $\mathcal{C}$.
(6) Let test image I_D have hash value equal to h_D. We can extract all triplets $\{\rho^j, \nu(I^j_R), h^j\}^N_{j=1}$ that have hash value equal to h_D. We can then estimate the data-driven proposal as:

$$q_{data}(S^\rho \rightarrow S'^\rho | \mathcal{C}, I_D) = P_{density}(\{\rho^j\}^N_{j=1}),$$

where $P_{density}$ is a density estimator (e.g. multi-variate gaussian kernel density estimator as in [19]).

4. *Picture* Programs

Our emphasis is to evaluate a general-purpose framework on a wide variety of 2D and 3D computer vision problems. Although additional tricks could be employed to improve results for all the tasks we consider and there may exist better fine-tuned baselines, our main emphasis is to solve a broad class of problems by minimizing problem-specific engineering. To this end, we developed and evaluated *Picture* programs for various tasks including: 3D body pose estimation, 3D reconstruction of objects and generative face analysis.

4.1. 3D Analysis of Faces

We obtained a 3D deformable face model trained on laser scanned faces from Paysan *et al* [38]. After training with this data, the model generates a mean shape mesh and mean texture, along with principal components and eigen vectors. A new face can be rendered by randomly choosing coefficients for the 3D model and running the program shown in figure 2. The representation layer ν in this program used the top-layer CNN convolutional features from the ImageNet model[20] and raw pixels. However, upon further investigation we found that using the deep convolutional inverse graphics network [26] instead of the pre-trained ImageNet CNN gave better results. We evaluated the program on a held-out test set of 2D projected images of 3D laser scanned data (dataset from [38]). We additionally produced a dataset of about 30 images from the held-out set with different viewpoints and lighting conditions. In figure 3, we show qualitative results of inference runs on the dataset. Many other academic researchers have used 3D deformable face models in an analysis-by-synthesis based approach [29, 22, 1]. However, *Picture* is the only system to solve this as well as many other unrelated computer vision problems using a general-purpose system and in under 50 lines of probabilistic code. Moreover, the data-driven proposals and abstract summary statistics (CNN activations) allow us to tackle the problem without explicitly using 2D face landmarks as compared to traditional approaches.

During experimentation, we also discovered that since the number of latent variables in this program are large (8 sets of 100 dimensional continuous coupled variables), elliptical slice moves are significantly more efficient than metropolis hastings proposals (see supplementary figure 2 for quantitative results). The data-driven proposals module was ran with the following settings:

Constructing data-driven proposals for faster and more accurate inference: We trained the data-driven proposals from about 100k program traces drawn from unconditional runs (i.e. no *observe* statement, see figure 2a for syntax). The summary statistic function ν_{dd} used were the pre-built top-layer CNN convolutional features from the ImageNet model[20]. The conditional proposal density $P_{density}$ was a multivariate kernel density function over cached latents with a Gaussian Kernel (0.01 bandwidth). As shown in figure 5, learning proposals significantly outperforms the baseline in terms of both speed and accuracy. The final proposal used during inference was a mixture kernel of the data-driven proposal (0.1 probability) and elliptical slice (0.9 probability).

4.2. 3D Human Pose Estimation

We developed a *Picture* program for parsing 3D pose of articulated humans from single images. There has been notable work in model-based approaches [13, 28] for 3d human pose estimation, which served as an inspiration for the human-pose program we describe in this section. However, in contrast to *Picture*, existing approaches require custom purpose inference strategies and are specifically designed for the task. As shown in supplementary figure 4, the probabilistic code consists of latent variables denoting bone and joints of an articulated 3D base mesh of a body. In our probabilistic code, we use an existing base mesh of a human body, defined priors over bone location and joints, and enable the armature skin-modifier API via *Picture*'s Blender engine API. The latent scene S^{ρ} in this program can be visualized as a tree with the root node around the center of the mesh, and consists of bone location variables, bone rotation variables and camera parameters. The representation layer ν in this program used fine-grained image contours [8] and the comparator is expressed as the probabilistic chamfer distance [43].

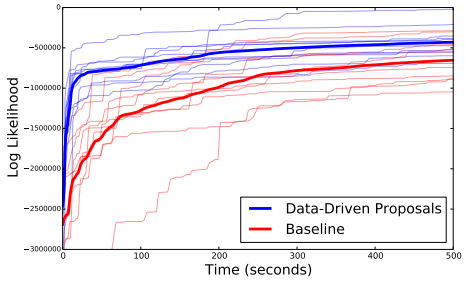


Figure 5: **Illustration of data-driven proposal learning for 3D face program:** We ran 50 independent chains for both approaches and show a few sample trajectories as well as the mean trajectories (in bold). Automatically learning data-driven proposals significantly improves the speed-accuracy trade-off. The data-driven approach used bottom-up proposals with a probability 0.1 and elliptical slice moves with a probability 0.9. The baseline only used elliptical slice sampling moves on all coupled continuous latent variables.

We evaluated our program on a dataset of humans performing a variety of poses, which was aggregated from: KTH [41] and LabelMe [40] images with significant occlusion in the "person sitting"(around 50 total images). Our choice of dataset was made in the context of the following considerations: (a) existing forward graphics simulators of articulated bodies are limited with respect to appearance but can represent arbitrarily complex articulated body configurations, and (b) fast methods for fine-grained bottom-up contour detection is an open problem in computer vision and does not work under heavy clutter. Therefore, our stimuli is constructed mainly to evaluate the model's capacity to reason about highly articulated bodies in low clutter environments.

We compared our methods with Deformable Parts Model (DPM) for pose estimation [49] (referred as DPM-pose), which is notably a 2D pose model. In order to quantitatively compare results, we project the 3D pose obtained from our model to 2D key-points. As shown in figure 6a, our system performs better than the baseline on this dataset. However, we advocate the integration of discriminative approaches like

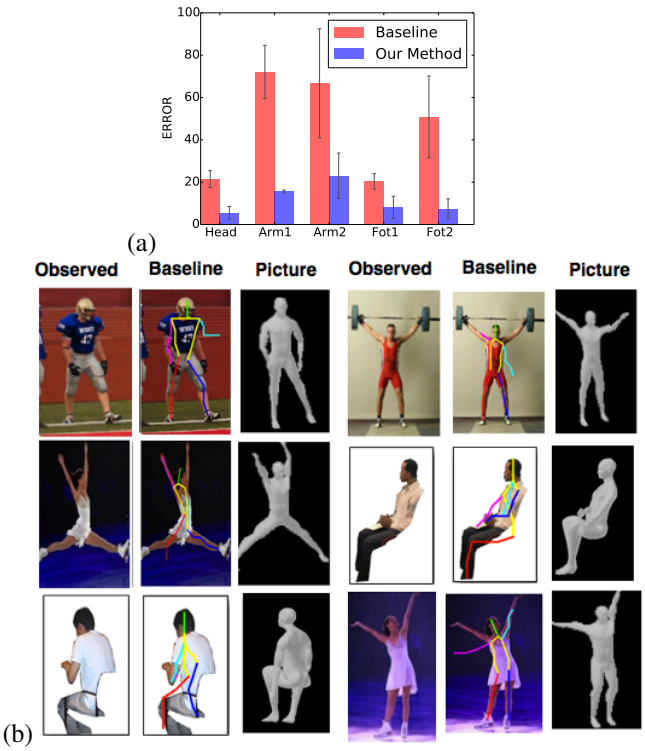


Figure 6: **Quantitative and qualitative results for 3D human pose program:** Refer to supplementary figure 4 for the probabilistic program. We quantitatively evaluate the pose program on a dataset collected from various sources such as KTH [41], LabelMe [40] images with significant occlusion in the "person sitting" category and the Internet. On the given dataset, as shown in the error histogram in **(a)**, our model is more accurate on average than just using the DPM based human pose detector [49]. The histogram shows average error for all methods considered over the entire dataset separated over each body part.

DPM-pose with probabilistic programming, which might turn out to be crucial for scaling such approaches to richer scene parsing problems. Moreover, as shown in figure 6b, images with people sitting and heavy occlusion are very hard for the discriminative model to get right – mainly due to "missing" observation signal – making a strong case for combining them with *model* based approaches which seems to give reasonable parses if we constrain the knee parameters to bend outwards in the prior. Most of our model's failure cases as shown in figure 6b, are in inferring the arm position; this is typically due to noisy and low quality feature maps around the arm area due to its small size.

Constructing data-driven proposals for faster and more accurate inference: Similar to the face example 4.1, we highlight the effect of data-driven proposal learning on the pose program in figure 7. We generated 500k program traces by unconditionally running the program. We used a pre-trained DPM pose model [49] as the function ν_{dd} and used similar program settings in case of the face program. As shown in figure 7, empirical results consistently favors inference runs with data-driven proposal learning both in terms

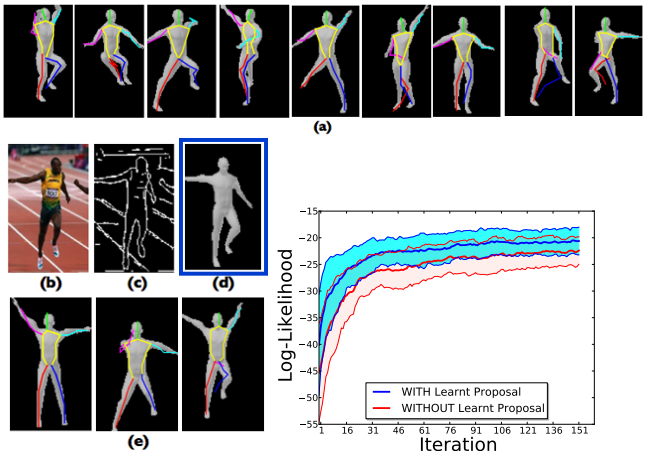
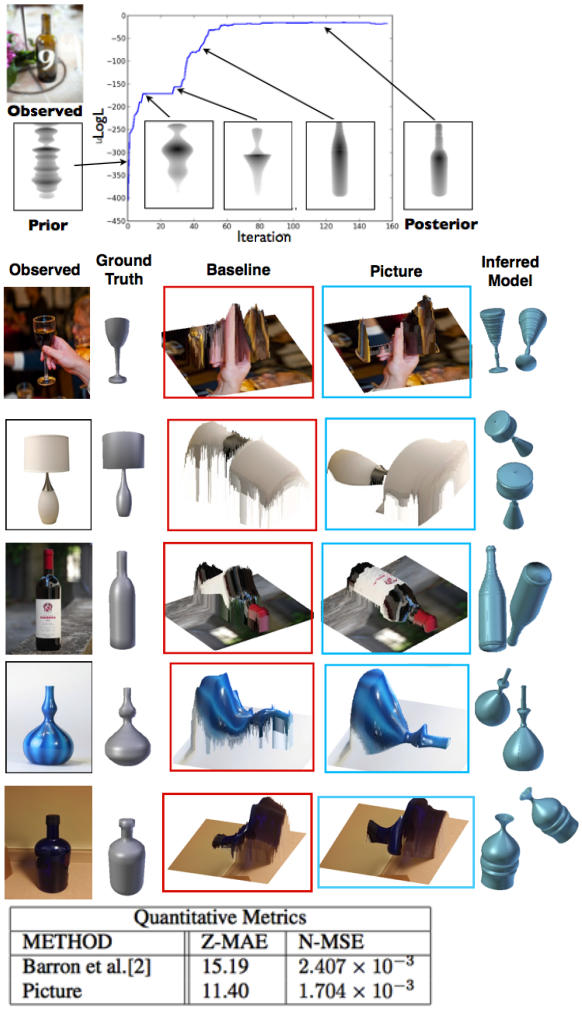


Figure 7: **Illustration of data-driven proposal learning for 3D human-pose program: (a)** Random program traces sampled from the prior during training. The colored stick figures are the results of applying DPM pose model on the *hallucinated* data from the program. **(b)** Representative test image. **(c)** Visualization of the representation layer $\nu(I_D)$. **(d)** Result after inference. **(e)** Samples drawn from the learnt proposal conditioned on test image are semantically close to the test image and results are fine-tuned by top-down inference to close the gap. As shown on the log-l plot, we run about 100 independent chains with and without the learned proposal. Inference with learned proposal (+ MCMC with some probability) consistently outperforms baseline in terms of both speed and accuracy.

of accuracy and speed. The final proposal used during inference was a mixture kernel of the data-driven proposal (0.1 probability) and MCMC (0.9 probability).

4.3. 3D Shape Program

Lathing and casting is a useful representation to express CAD models and is the main inspiration for our approach to modeling medially-symmetric 3D objects. It is fairly easy to express such a forward generation process of CAD objects by writing down a probabilistic program as shown in supplementary figure 3. However, the distribution induced by the program has a fairly complex form. Given object boundaries in $\mathcal{B} \in \mathcal{R}^2$ space, we can lathe an object by taking a cross section of points (fixed for this program), defining a medial axis for the cross section and sweeping the cross section across the medial axis by continuously perturbing with respect to $\mathcal{B}$. The space over $\mathcal{B}$ is extremely large due to variability in 3D shapes. However, *Picture* allows us to easily define a flexible non-parametric prior over object profiles (hereby referring to $\mathcal{B}$ by $f_{\mathcal{GP}}(x)$) for lathing using Gaussian Processes [39] (GPs). The probabilistic shape program produces an intermediate mesh of whole or part of the 3D object (soft-constrained to be in the middle of the screen), which then gets rendered to an image I_R by a deterministic camera re-projection function. The representation layer and the comparator used in this program were same as the one's



Quantitative Metrics		
METHOD	Z-MAE	N-MSE
Barron et al.[2]	15.19	2.407×10^{-3}
Picture	11.40	1.704×10^{-3}

Figure 8: **Qualitative and quantitative results of 3D object reconstruction program**: Refer to supplementary figure 3 for the probabilistic program. **Top:** We illustrate a typical inference trajectory of the sampler from prior to the posterior on a representative real world image. **Middle:** Qualitative results on representative images. **Bottom:** Quantitative results in comparison to [3]. For details about the scoring metrics, refer to section 4.3.

used for the 3D human pose example in section 4.2. The proposal kernel we used during inference consisted of blocked MCMC proposals on all the coupled continuous variables as described in the supplementary material. For a detailed summary and probabilistic program, refer to supplementary section 1.

We evaluate this program on a RGB image dataset of 3D objects with large shape variability. We asked CAD experts to manually generate CAD model fits to these images in blender and evaluated our approach in comparison to state-of-art 3D reconstruction algorithm from [3](SIRFS). It is important to note that SIRFS predominantly utilizes only low level shape priors such as piece-wise smoothness and is solving a much harder problem of inferring intrinsic images along with depth-maps. Also, our model has the added advantage of the assumption of more high-level shape priors. In the future, in order to scale-up single image reconstruction, it will be critical to combine low level shape/reflectance/lighting priors like in SIRFS[3] with generative shape programs. For quantitative performance, we calculated two metrics: (a) Z-MAE – Shift-invariant surface mean-squared error and (b) N-MSE – mean-squared error over normals[3]. As shown in figure 8, the shape program has a lower Z-MAE and N-MSE score than SIRFS [3]. As shown in figure 8, we also obtain qualitatively better results as compared to the baseline.

5. Discussion

There are several directions for future research including introducing a notion of dependency tracking mechanism to exploit conditional independences for parallel inference. It will be necessary to exploit automatic particle filtering based inference schemes [48, 25] for image sequences. Additionally, for scaling up our approach, better illumination [52], texture and shading models will be needed. Procedural computer graphics [2, 10] is also an appealing research avenue to scale up *Picture* for 3D full-scene understanding problems [50, 14, 7, 15]. Most importantly, flexible and expressive scene generator libraries for a wide class of objects/scenes are needed. We are also interested in extending *Picture* by taking insights from learning based "analysis-by-synthesis" approaches such as transforming auto-encoders [18], capsule networks [44] and deep convolutional inverse graphics network [26]. These models learn an implicit graphics engine in an encoder-decoder style architecture. With probabilistic programming, the space of decoders need not be restricted to neural networks and could consist of arbitrary probabilistic graphics programs with internal parameters.

The recent renewal of interest in inverse graphics approaches to vision has motivated the development of a number of general-purpose modeling and inference frameworks. Each addresses a different facet of the general problem. Differentiable renderers make it easier to fine-tune the numerical parameters of high-dimensional scene models [30]. Data-driven proposal schemes, including the informed sampler, suggest a way to rapidly identify plausible scene elements [19]. Earlier formulations of generative probabilistic graphics programming showed that probabilistic programs could effectively represent scene models and provided a flexible template for automatic inference [32]. However, none of these approaches have been individually sufficient for solving a broad class of complex scene perception problems. This paper shows that by integrating each of these approaches into a single probabilistic language and inference engine, it may be feasible to begin scaling up inverse graphics.

6. Acknowledgements

We thank Thomas Vetter for giving us access to the Basel face model. T. Kulkarni was graciously supported by the Leventhal Fellowship. This research was supported by ONR award N000141310333, ARO MURI W911NF-13-1-2012 and the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. We would like to thank Ilker Yildrim, Yura Perov, Karthik Rajagopal, Alexey Radul, Peter Battaglia, Alan Rusnak, and three anonymous reviewers for helpful feedback and discussions.

References

[1] O. Aldrian and W. A. Smith. Inverse rendering of faces with a 3d morphable model. *PAMI*, 2013.

[2] M. Averkiou, V. G. Kim, Y. Zheng, and N. J. Mitra. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In *Computer Graphics Forum*, volume 33, pages 125–134. Wiley Online Library, 2014.

[3] J. Barron and J. Malik. Shape, illumination, and reflectance from shading. Technical report, Berkeley Tech Report, 2013.

[4] J. Bernardo, J. Berger, A. Dawid, A. Smith, et al. Regression and classification using gaussian process priors. In *Bayesian Statistics 6: Proceedings of the sixth Valencia international meeting*, volume 6, page 475, 1998.

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam,

[6] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[7] L. Del Pero, J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard. Understanding bayesian rooms using composite 3d object models. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 153–160. IEEE, 2013.

[8] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. 2013.

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

[10] N. Fish, M. Averkiou, O. Van Kaick, O. Sorkine-Hornung, D. Cohen-Or, and N. J. Mitra. Meta-representation of shape families. In *Computer Graphics Forum*, volume 32, pages 189–200, 2013.

[11] U. Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

[12] U. Grenander, Y.-s. Chow, and D. M. Keenan. *Hands: A pattern theoretic study of biological shapes*. Springer-Verlag New York, Inc., 1991.

[13] P. Guan, A. Weiss, A. O. Balan, and M. J. Black. Estimating human shape and pose from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1381–1388. IEEE, 2009.

[14] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Computer Vision–ECCV 2010*, pages 482–496. Springer, 2010.

[15] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *Computer Vision–ECCV 2010*, pages 224–237. Springer, 2010.

[16] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[17] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[18] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 44–51. Springer, 2011.

[19] V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler. The informed sampler: A discriminative approach to bayesian inference in generative computer vision models. *arXiv preprint arXiv:1402.0859*, 2014.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[21] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2145–2152. IEEE, 2006.

[22] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. *PAMI*, 2011.

[23] R. Knothe. *A Global-to-local model for the representation of human faces*. PhD thesis, University of Basel, 2009.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[25] T. D. Kulkarni, A. Saeedi, and S. Gershman. Variational particle approximations. *arXiv preprint arXiv:1402.5715*, 2014.

[26] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. *arXiv preprint arXiv:1503.03167*, 2015.

[27] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

[28] M. W. Lee and I. Cohen. A model-based approach for estimating human 3d poses in static images. *PAMI*, 2006.

[29] M. D. Levine and Y. Yu. State-of-the-art of 3d facial reconstruction methods for face recognition based on a single 2d training image per person. *Pattern Recognition Letters*, 30(10):908–913, 2009.

[30] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *ECCV 2014*. 2014.

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

[32] V. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using gen-

erative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.

[33] V. Mansinghka, D. Selsam, and Y. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

[34] I. Murray, R. P. Adams, and D. J. MacKay. Elliptical slice sampling. *arXiv preprint arXiv:1001.0175*, 2009.

[35] R. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[36] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[37] B. Paige and F. Wood. A compilation target for probabilistic programming languages. *arXiv preprint arXiv:1403.0504*, 2014.

[38] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. Genova, Italy, 2009. IEEE.

[39] C. E. Rasmussen. Gaussian processes for machine learning. 2006.

[40] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2008.

[41] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[42] A. Stuhlmüller, J. Taylor, and N. Goodman. Learning stochastic inverses. In *Advances in neural information processing systems*, pages 3048–3056, 2013.

[43] A. Thayananthan, B. Stenger, P. H. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *CVPR*, 2003.

[44] T. Tieleman. *Optimizing Neural Networks that Generate Images*. PhD thesis, University of Toronto, 2014.

[45] Z. Tu and S.-C. Zhu. Image segmentation by data-driven markov chain monte carlo. *PAMI*, 24(5):657–673, 2002.

[46] R. D. Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12(2):129–141, 2013.

[47] D. Wingate, A. Stuhlmueller, and N. D. Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *International Conference on Artificial Intelligence and Statistics*, pages 770–778, 2011.

[48] F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, 2014.

[49] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011.

[50] Y. Zhang, S. Song, P. Tan, and J. Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Computer Vision–ECCV 2014*, pages 668–686. Springer, 2014.

[51] Y. Zhao and S.-C. Zhu. Image parsing via stochastic scene grammar. In *NIPS*, 2011.

[52] J. Zivanov, A. Forster, S. Schonborn, and T. Vetter. Human face shape analysis under spherical harmonics illumination considering self occlusion. In *Biometrics (ICB), 2013 International Conference on*, pages 1–8. IEEE, 2013.