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 3D face analysis, 3D human pose estimation, and 3D object reconstruction – 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 [1–4]. Generative approaches hold out the promise of analyzing complex scenes more richly and flexibly [5–8], 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. Available Markov Chain Monte Carlo (MCMC) 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 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. The vast majority of faces, the 3D pose of articulated human bodies, and the 3D shape of medially-symmetric objects can be compared using a likelihood function or a distance metric $\lambda$ (as in Approximate Bayesian Computation [7]).

(b) Inference Engine: Automatically produces a variety of proposals and iteratively evolves the scene hypothesis $S$ to reach a high probability state given $I_D$. Every $\text{Picture}$ program has the following components. Scene Language: Describes 2D/3D scenes and generates particular scene-related trace variables $S^p \in \rho$ during execution. Approximate Renderer: Produces graphics rendering $I_R$ given $S^p$ and latents $X^p$ 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, contours and pixels). Comparator: During inference, $I_R$ and $I_D$ can be compared using a likelihood function or a distance metric $\lambda$ (as in Approximate Bayesian Computation [7]).

(d) Illustrative results: We demonstrate Picture on a variety of 3D computer vision problems and check their validity with respect to ground truth annotations and task-specific baselines.

References


