Real-time Approximate Inference for Scene Understanding with Generative Models

Javier Felip[*] Intel Labs

Nilesh Ahuja Intel Labs

David Gómez-Gutiérrez Intel Labs

Omesh Tickoo Intel Labs

Vikash Mansinghka MIT

Abstract

Consider scene understanding problems such as predicting where a person is probably reaching, or inferring the pose of 3D objects from depth images, or inferring the probable street crossings of pedestrians. This paper shows how to solve these problems using inference in generative models, by introducing new techniques for real-time inference. The underlying generative models are built from realistic simulation software, wrapped in a Bayesian error model for the gap between simulation outputs and real data. This paper shows that it is possible to perform approximately Bayesian inference in these models, obtaining high-quality approximations to the full posterior over scenes, without using bottom-up neural networks. Instead, this paper introduces two general real-time inference techniques. The first is to train neural surrogates of the simulators. The second is to adaptively discretize the latent variables using a Tree-Pyramid (TP) approach. This paper also shows that by combining these techniques, it is possible to perform accurate, approximately Bayesian inference in realistic generative models, in real-time.

1 Introduction

Visual scene understanding involves inferring scene descriptions (like object category, location, etc.) from observed images and videos. Bayesian approaches to such problems are gaining importance, as these provide rigorous estimates of uncertainty. In some problems, such as inferring the intents of people, accurate characterization of uncertainty is fundamental to the problem. However, Bayesian inference is widely viewed as too slow for real-time applications. This paper introduces two techniques for real-time approximate inference in generative models that can be applied to a broad class of scene understanding problems. It also shows empirically that the techniques can be combined to solve real-world instances in real-time.

The generative models in this paper are built from realistic off-the-shelf simulation software. On top of these simulators, the generative models include a simple Bayesian error model for the gap between simulation outputs and real data. This formulation can be thought of as a variant of Approximate Bayesian Computation (ABC) [1, 2, 3, 4], in which the Bayesian error model plays the role of the ABC distance metric. The ABC approach has previously been successfully applied to solve 2D and 3D computer vision problems [5, 6, 7]. However, these architectures are not fast or accurate enough for real-time applications. The contributions of this paper are (i) new techniques for real-time inference in generative models, that could apply to a broad class of perception problems, and (ii) an empirical demonstration of efficacy on scene understanding problems.

2 Real-time Inference Techniques

The problem of Bayesian inference in simulator-based models has been extensively studied in the Approximate Bayesian Computation (ABC) community, and has proved computationally challeng-

[*] javier.felip.leon@intel.com, nilesh.ahuja@intel.com, david.gomez.gutierrez@intel.com, omesh.tickoo@intel.com, vkm@mit.edu

Workshop on Perception as Generative Reasoning, NeurIPS 2019, Vancouver, Canada.

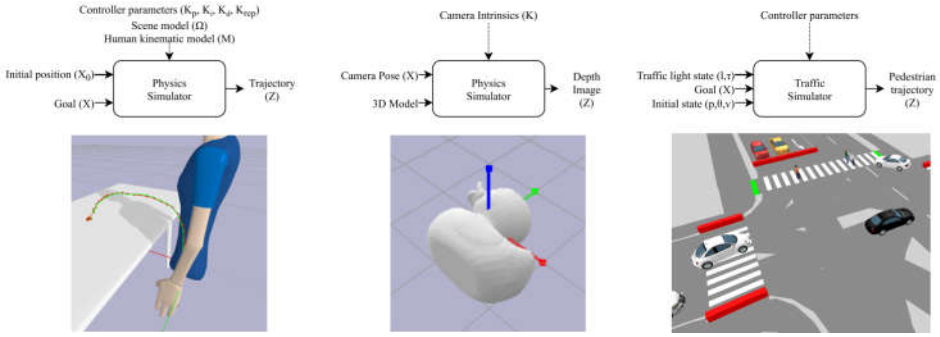

Figure 1: **Simulators used to define generative models:** Reach-intent estimation (left), 3D object pose estimation (center), and pedestrian intent estimation (right). Top row shows inputs, outputs, and known parameters for each simulator. Bottom row shows a sample output from each simulator. The task is to infer probable inputs given real-world data treated as a noise-corrupted simulator output.

ing [8]. Monte Carlo approximations require expensive likelihood evaluations, and the posterior distributions can be complex and multimodal. This paper introduces two techniques, that together make it possible to apply ABC with complex simulators in real-time applications. The first technique is to train neural surrogates for the simulators. The neural surrogates allow for fast approximate likelihood evaluations, which can be many orders of magnitude faster than the original simulators. Unlike the neural surrogates that are more frequently used in ABC, these surrogates are trained on simulated data that is intentionally corrupted with additive Gaussian noise, following the Bayesian error model. This helps to account for the gap between the simulator and reality. It can be viewed as a simple form of domain randomization [9], an increasingly popular technique for building deep learning systems that are trained on synthetic data from complex simulators but deployed on data from the real world [10]. However, even with neural surrogates, our experiments confirm that Monte Carlo approximations to these kinds of complex posteriors can still be unnecessarily expensive. Thus, the second technique is an adaptive discretization algorithm for approximating the posterior distribution based on Tree-Pyramids (TPs), a hierarchical data structure from computer graphics [11]. Experiments show that this TP-based approximate inference algorithm can represent complex posterior distributions induced by neural surrogates more accurately than Monte Carlo approximations, given the same computational budget.

2.1 Generative modeling via realistic simulators

It is desired to estimate a latent variable, X, from an observation, D. Assume that we have access to a simulator that models the transformation from the input space of X to the observation space. The distribution on X is represented by a scene prior, $p(X)$. Given an input X, the simulator produces a simulated output, Z, which has an associated conditional distribution $p(Z|X)$. The relation between D and Z is modeled as:

$$p(D|Z,\epsilon) \sim Normal(Z,\epsilon),\ p(\epsilon) \sim Gamma(1,1), \tag{1}$$

where, ϵ is a slack variable representing the gap between simulation and reality. Scene understanding then becomes the problem of performing posterior inference jointly over ϵ and X:

$$p(X,\epsilon|D) \propto p(X)p(\epsilon)p(D|X,\epsilon). \tag{2}$$

As is typical in ABC problems, the likelihood distribution induced by the simulators in this paper, $p(D|X,\epsilon) = \int p(D|Z,\epsilon)P(Z|X)dZ$, can be difficult to calculate. Note that we are interested in posterior inferences about both the scene X and the parameter ϵ that governs the error. This is because we may wish to filter out scenes where

$$\epsilon^* = \operatorname{argmax} p(\epsilon|D) \tag{3}$$

is high. The dynamics of joint real-time inference over X and ϵ implement a form of real-time model criticism [12]. When the data is unlikely under the simulator, high values of ϵ are favored in the posterior. In addition to improving robustness to simulator-data mismatch, this extension makes it possible to filter datasets to detect samples that violate the assumptions in a given simulator model. In the additional material, we show how this kind of filtering can find anomalies such as pedestrians who break traffic rules. The following two subsections describe the techniques used to achieve real-time performance.

2.2 Training neural surrogates with domain randomization

The first technique used in this paper to make real-time ABC feasible is to train neural surrogates $f_\phi(X)$ for the simulator outputs. These are deterministic neural networks that only approximate the true underlying simulator $p(Z|X)$ using their parameters ϕ: $f_\phi(X) \approx \text{argmax}_Z\, p(Z|X)$.

However, unlike many applications of ABC in science and engineering, our scene understanding involve real-world data that may not be accurately modeled by the underlying simulator. We employ a model-based variation on domain randomization, a widespread technique used for applying deep learning to synthetic data [9]. Specifically, we train a randomized surrogate f^*_ϕ to replicate the most probable outcome of a noisy version of the simulator including an error model, with high ϵ^* of Gaussian error added to the simulator outputs: $f^*_\phi(X) \approx \text{argmax}_D\, p(D|X, \epsilon^*)$.

In multiple experiments, this randomized training approach yielded surrogates that performed more robustly on real-world data. We note that there are intriguing parallels between this domain randomization and the error model; future work exploring this relationship may be relevant.

Given this neural surrogate, the inference problem we solve is defined by the following approximation to the true posterior (following eq. (2)):

$$p(X, \epsilon|D) \propto p(X)p(\epsilon)Normal(f^*_\phi(X), \epsilon). \tag{4}$$

Training data for the surrogates are generated by their respective simulators. The networks are trained to minimize the following loss function on synthetic data elements $(\hat{X}, \hat{Z})$ sampled from the simulator:

$$L = \|f_\phi(\hat{X}) - Normal(\hat{Z}|\hat{X}, \epsilon^*)\|^2. \tag{5}$$

2.3 Adaptive discretization via Tree Pyramids

A second technique we use to speed up inference is to adaptively discretize the latent variables, using techniques from computer graphics. Specifically, we propose a K-dimensional tree-pyramid (KD-TP) based posterior sampling algorithm. A KD-TP is a full tree where each node has either zero or 2^K children and represents a hierarchical subdivision of K-dimensional space into convex subspaces. By hierarchically sampling the posterior in each of the subspaces, we show an efficient sampling approach. See an example of the creation sampling procedure in Figure 2.

Algorithm 1 shows our adaptive discretization strategy. The required parameters consist of: (i) k, dimensionality of the space; (ii) τ, the likelihood expansion threshold; (iii) ρ, the resolution limit; (iv) D, the observation; (v) $\Sigma = \{\epsilon_0, ..., \epsilon_n\}$, the discretized slack terms; (vi) c, the sample space center/root node location; and (vii) r, the sample space span.

First, the root node is added to the expansion set χ. While the expansion set is not empty, the nodes are expanded (find details of node expansion in additional material) and their children are added to the evaluation set E. Using the observed data D and the discretized slack terms Σ, the likelihood for each node in E is computed (sample values are obtained at the center of each node). Thanks to the use of neural emulators, the likelihood computations of all nodes $n \in E$ with all $\epsilon \in \Sigma$ are easily vectorized, dramatically reducing the computation time. In lines 8 to 11, the samples from E with a radius bigger than the limit radius ρ and with a likelihood over the threshold τ are added to the set χ. The likelihood of each sample is the maximum of the likelihoods obtained with each of the slack values $\epsilon \in \Sigma$. The result of the inference is the center of the leaf node with maximum likelihood. Figure 2 shows an example of executing the proposed posterior sampling algorithm in a 2D-TP for the reaching intent prediction problem.

3 Results

To demonstrate the effectiveness of the proposed method, we focus on three scene understanding applications: (a) predicting what physical object a person is reaching for, (b) inferring the probable street crossing of pedestrians at a traffic intersection, and (c) estimating pose of 3D objects from depth images. For a detailed description of the three scenarios, see the additional material.

We have evaluated the neural surrogates and tree pyramid approaches in accuracy and runtime including a comparison to baseline particle filter [13], MCMC Metropolis-Hastings [14], ABC-Reject [2] and ABC-SMC [1]. Results for the reaching intent inference problem are shown in Figure 3. The rest of details required for the experimental set-up are detailed in the additional

Algorithm 1 Inference algorithm based on adaptive discretization using posterior TP

function COMPUTETPPOSTERIORAPPROXIMATION($k, \tau, \rho, D, \Sigma, c, r$)
  $T \leftarrow TP(k, c, r)$ ▷ Initialize tree with the dimensions, center and radius.
  $\chi \leftarrow \{getRoot(T)\}$ ▷ Add the tree root node to the expansion set χ.
  while $\chi \neq \emptyset$ **do**
    $\{e_n\} \leftarrow genCandidateTPExpansions(T, \chi, k)$
    $\chi \leftarrow \emptyset$
    $L \leftarrow likelihood(e_i, D, \Sigma)$ ▷ Compute likelihoods for each candidate node.
    for $n \leftarrow 1...N_L$ **do**
      if $L_n > \tau$ **and** $getRadius(e_n) > \rho$ **then**
        $\chi = \chi \cup \{e_n\}$ ▷ Add nodes from the candidate set E that satisfy
  return T, μ ▷ the expansion criteria to the expansion set χ.

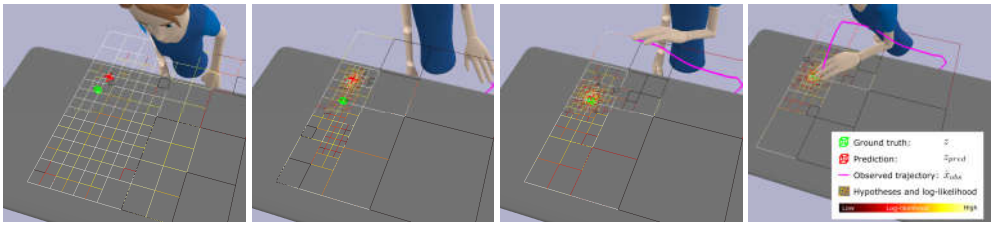

Figure 2: **Adaptive discretization of latent variables using Tree-Pyramids (TPs).** This figure shows an approximate posterior distribution for the reaching intent at four different moments in a single reaching movement. Note that the tree grows articulated in regions of high probability.

material. We have performed many simulations, and applied our real-time inference techniques to live data in a real scenario, find an example in the following video: https://bit.ly/2OSNhwA.

4 Discussion

This paper has shown that it is possible to use ABC for real-time scene understanding. Applications include inferring the goal to which a person is reaching; inferring the 3D pose of an object from depth images; and inferring the probable crossing time of a pedestrian at an intersection. The modeling approach is based on realistic software simulators that are used as the core of probabilistic generative models, with a simple Bayesian error model that links the simulator output to real data. This paper introduces two techniques for improving performance. The first is the use of neural surrogates, trained with simple domain randomization. This paper shows empirically that neural surrogates can be hundreds to thousands of times faster than the original simulators in our applications. The second is a technique for adaptive discretization of the latent variables, that improves inference accuracy and robustness as compared to Monte Carlo baselines. We hope the techniques introduced in this paper help researchers explore other real-time applications of modeling approaches from the ABC community, and to develop new real-time methods for approximate Bayesian inference.

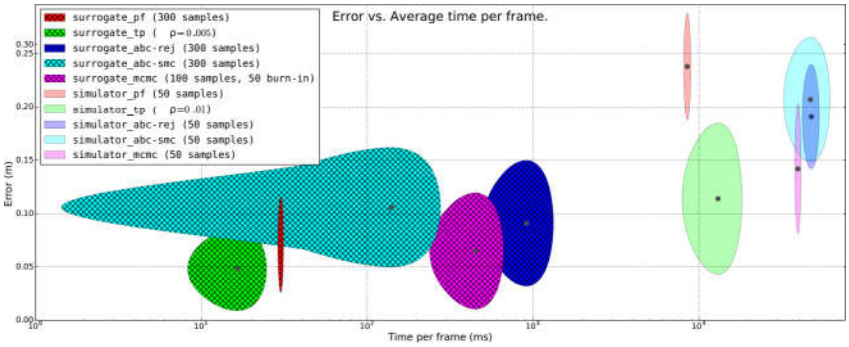

Figure 3: **Error vs. Average time per frame.** Pattern-filled ellipses represent inference methods using neural likelihood surrogate. Ellipses show 1-stdev boundary after averaging 30 experiments. The combination of TP and neural surrogates strikes the best accuracy-time.

References

[1] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

[2] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate bayesian computation. *PLoS computational biology*, 9(1), 2013.

[3] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[4] Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

[5] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015.

[6] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.

[7] Marco F Cusumano-Towner, Feras A Saad, Alexander K Lew, and Vikash K Mansinghka. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 221–236. ACM, 2019.

[8] Daniel Wegmann, Christoph Leuenberger, and Laurent Excoffier. Efficient approximate bayesian computation coupled with markov chain monte carlo without likelihood. *Genetics*, 2009.

[9] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[10] Shuang Li, Xiaojian Ma, Hongzhuo Liang, Michael Görner, Philipp Ruppel, Bing Fang, Fuchun Sun, and Jianwei Zhang. Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. *arXiv preprint arXiv:1809.06268*, 2018.

[11] Henry Fuchs, Zvi M Kedem, and Bruce F Naylor. On visible surface generation by a priori tree structures. In *ACM Siggraph Computer Graphics*, volume 14, pages 124–133. ACM, 1980.

[12] Oliver Ratmann, Christophe Andrieu, Carsten Wiuf, and Sylvia Richardson. Model criticism based on likelihood-free inference, with an application to protein network evolution. *Proceedings of the National Academy of Sciences*, 106(26):10576–10581, 2009.

[13] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 325(6):653–658, 1997.

[14] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[15] E Coumans. Bullet, game physics simulation. http://www.bulletphysics.org/.

[16] Elias Strigel, Daniel Meissner, Florian Seeliger, Benjamin Wilking, and Klaus Dietmayer. The KO-PER intersection laserscanner and video dataset. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1900–1901. IEEE, 2014.

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[19] DM Henderson. Euler angles, quaternions, and transformation matrices for space shuttle analysis. 1977.

[20] Markus Deserno. How to generate equidistributed points on the surface of a sphere. 2004.