Learning Conditional Error Model for Simulated Time-Series Data

Ashish Shrivastava
Apple Inc
ashish.s@apple.com

Oncel Tuzel
Apple Inc
otuzel@apple.com

Abstract

Applications such as autonomous navigation [1], human-robot interaction [2], game-playing robots [8], etc., use simulation to minimize the cost of testing in real world. Furthermore, some machine learning algorithms, like reinforcement learning, use simulation for training a model. To test reliably in simulation or deploy a model in the real world that is trained with simulated data, the simulator should be representative of the real environment. Usually, the simulator is based on manually designed rules and ignores the stochastic behavior of measurements. In particular, we would like to learn a model that captures uncertainties of the sensing algorithms (e.g. neural networks used to detect objects) in real world and add them in simulation. We model the distribution of residuals between the ground truth states of the objects and their perceived states by the sensing algorithm. This error distribution depends both on the current state of the object (e.g. distance from the sensor) and its past residuals. We assume the error distribution is conditionally Gaussian, and we use a deep neural neural network (DNN) to map the object states and past residuals to the distribution parameters (mean and variance). Our conditional model perturbs the dynamic objects' states (position, velocities, orientations, and shape) and produces smoother trajectories which look similar to the real data.

1. Introduction

An algorithm usually requires a thorough evaluation before it is deployed in a real world environment. The comprehensive evaluation of the algorithm becomes challenging as the complexity of the environment grows, often because testing becomes prohibitively slow. In such cases, to reduce the cost and the time, simulation is used to augment real world testing and has become an important component in the development of complex algorithms [9, 10, 3, 14, 12, 5, 4]. For example, an autonomous navigation algorithm is tested in simulation to quickly prototype and thoroughly evaluate performance before testing in the real world. Furthermore, simulators can be used to generate data to train machine learning (ML) models [13]. Some ML algorithms (e.g. reinforcement learning) use simulation to learn a policy that can be used in the real world [15]. For these models to generalize well in the real world, the simulator needs to be realistic and should model the stochasticity of the real environments.

Simulators are usually designed with hand-crafted rules and ignore the stochastic behavior of the real world environment. In a robot navigation application, a simulator consists of dynamic and static objects, where the dynamic objects are sensed without consideration of measurement error and they move in a deterministic manner. For instance, dynamic objects move in a straight line and have fixed pose and shape. This rule-based design ignores measurement error from the sensors and may fail to capture scenarios the robot encounters in the real world.

One way to model the measurement error would be to use an independent and identically distributed (i.i.d.) model such as a Gaussian Mixture Model (GMM). However, for many applications, the i.i.d. assumption does not hold because the measurement error often depends on the state of the world. For example, objects that are far from the robot may have larger error due to sensor limitations. The error in the sensing module of an autonomous system can depend on how fast the vehicles are moving, the distance between the autonomous agent and other vehicles, their relative orientations, etc. Such errors cannot be modeled by an i.i.d. GMM. Furthermore, the i.i.d. model does not consider the time correlation of the errors for time-series data and results in jittery predictions across time.

In this work, we learn a conditional model that perturbs the perceived states of dynamic objects in a simulator to model the uncertainties of the sensor measurements (Figure 1). Unlike previous work on uncertainty estimation (such as [6, 11]), we model the uncertainty of a time series data to improve the realism of a simulator.

2. Conditional Error Model

We model the measurement error with a conditional Gaussian distribution whose mean and variance depend on features that describe the state of the environment. In addition, to learn the correlation of errors across time, we also con-

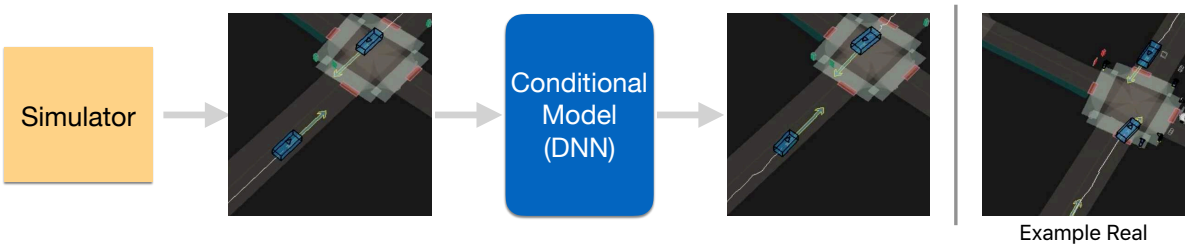


Figure 1: Overview of the system using the proposed conditional error model. We train a DNN model that outputs an error distribution for each state of the object, conditioned on the state of the world in the simulator. We sample error from the predicted distribution and add it to the object's state to make the simulator more realistic.

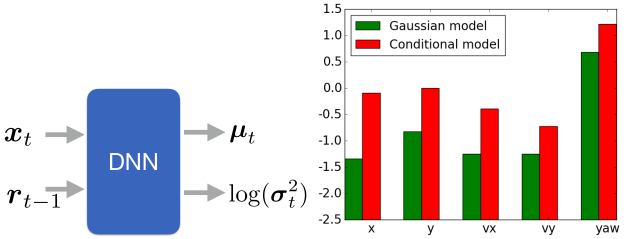
dition the error distribution on previous error samples. To estimate the error distribution, we use a Deep Neural Network (DNN) that takes a feature vector that describes the state of the world and the previous sample of the error as inputs and outputs mean and variance of the error distribution (Figure 2a). We sample from this distribution and add it to the current state of the object in the simulator to improve its realism. We learn this distribution using pairs of the ground truth states of objects, $\hat{\boldsymbol{s}}_t$, and the corresponding measurements from the sensing algorithms, $\boldsymbol{s}_t$, where t is the time index. We denote the errors by $\hat{\boldsymbol{r}}_t \triangleq \boldsymbol{s}_t - \hat{\boldsymbol{s}}_t$. For each sample $\hat{\boldsymbol{s}}_t$, we compute a feature vector $\boldsymbol{x}_t$ which may include properties of the environment (e.g. local map) in addition to the object states. We parameterize the Gaussian distribution with mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$. Here $\boldsymbol{\theta}$ are the parameters of the DNN (Figure 2a). The log likelihood $\mathcal{L}(\boldsymbol{\theta})$ of the error samples is given by:

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_t, \hat{\boldsymbol{r}}_t) = \sum_t (\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \hat{\boldsymbol{r}}_t)^2 \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)^{-2} + 2 \log \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) + \text{const.}$$

To learn the correlation across time, we also input the previous error sample $\boldsymbol{r}_{t-1}$ to the DNN. With slight abuse of notation, the likelihood function can be written as $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_t, \hat{\boldsymbol{r}}_t, \boldsymbol{r}_{t-1})$. Here $\boldsymbol{r}_t$ is a sample from the predicted error model: $\boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)$ while $\hat{\boldsymbol{r}}_t$'s are the training data (error samples). To model the likelihood of the sequence, the following multi-step prediction problem can be solved:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_t, \hat{\boldsymbol{r}}_t, \boldsymbol{r}_{t-1}) + \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_{t+1}, \hat{\boldsymbol{r}}_{t+1}, \boldsymbol{r}_t) + \ldots + \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_{t+T}, \hat{\boldsymbol{r}}_{t+T}, \boldsymbol{r}_{t+T-1}), \quad (1)$$

where T is the time horizon. Traditional sequence modeling methods (e.g. Wavenet [16]) model the sequence likelihood with a "teacher enforcing" approach during training, where the ground-truth output at time $t-1$ is fed as an input at time t. However, in our case, we found the teacher enforcing to overfit the training data and caused the objects to drift at inference time. Using the predicted samples as input requires sampling from the predicted model which is



(a) DNN model (b) Quantitative comparison

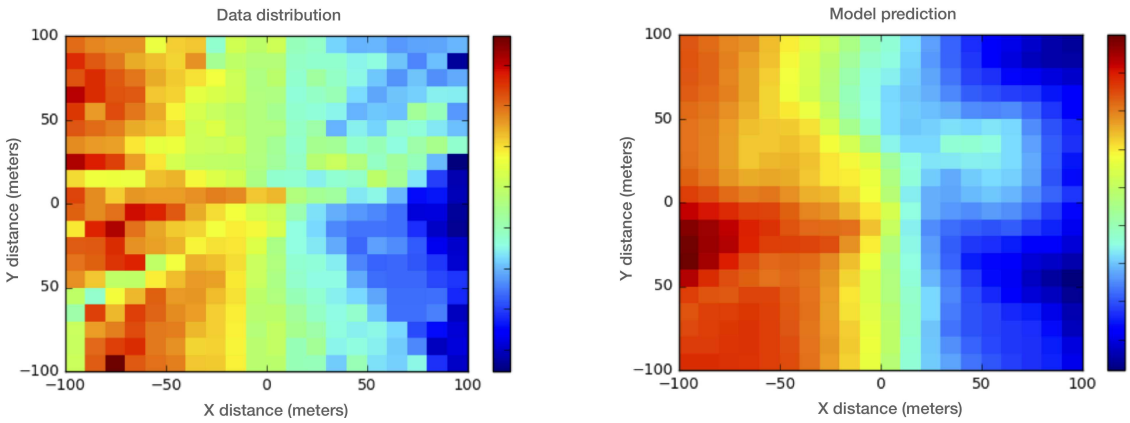
Figure 2: (a) The inputs to DNN model are the features $\boldsymbol{x}_t$ and the previous error samples $\boldsymbol{r}_{t-1}$. (b) Quantitative comparison of the conditional model with Gaussian baseline on a held-out test data. The likelihood of the i.i.d. Gaussian model is smaller because it cannot model the conditional nature of the error, nor correlation across time.

a non-differentiable operation. To avoid this problem, we use a well known re-parameterization trick of variational autoencoders [7] where, to compute $\boldsymbol{r}_t$, we sample from a zero mean and unit variance Gaussian, then scale by $\boldsymbol{\sigma}_t$ and add $\boldsymbol{\mu}_t$. Our model is a simple form of recurrent neural network (RNN) where we map the previous sampled error (r_{t-1}) and the feature vector encoding the object and environment (x_t) to error distribution parameters via three fully connected layers. Then we sample from the error distribution and repeat the process in the next time instance.

At inference time, we predict one sample at a time with a forward pass through the network.

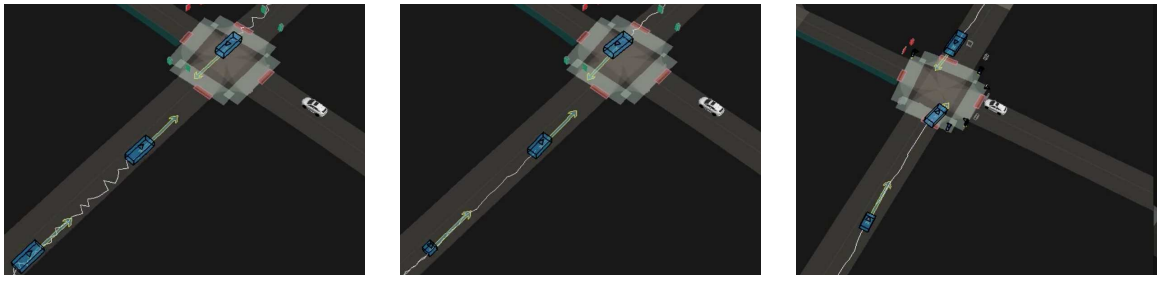
3. Experiments

We apply the proposed approach to model the sensing measurement errors in a robot navigation application. The states of the objects in the environment are positions (x, y), velocities (vx, vy), orientations (yaw), and the shape (length, width, height). To obtain the ground truth data, we instrument the objects in the real world with high precision location sensors. Our training data consists of the residual between the measured values with the sensing module and the ground-truth from the high precision sensor. We train



(a) example error distribution (b) sampled error distribution from the learned model

Figure 3: Example of the conditional measurement error in the position (along the x-axis) of the objects perceived by the robot as a function of the location of the object (a). This example demonstrates that the error in data in truly conditional and is modeled by our conditional model in (b).



(a) i.i.d. GMM model (b) proposed conditional model (c) example real frame

Figure 4: Qualitative comaprison of GMM (a) and the proposed conditional model (b) where we model the measurement error for the pose and velocities of the objects in the environment. In the real data (c), although the error is large, the trajectories are smooth because the error is correlated. The i.i.d. fails to model the correlation, producing a jittery trajectory. As we can see, qualitatively the conditional model is closer to the real data (trajectory is smoother despite the amount of error being large). The object in the bottom left corner is made smaller because the shape error of the object increases with distance from the sensor (shown in white) which is also the case for the real data.

the DNN by minimizing the loss in (1) using the RMSprop optimizer for 10000 steps with initial learning rate of 0.01 that was decayed by a factor of 0.9 after every 500 steps. We compute log likelihood, which quantifies how well the model explains the errors (on a held-out dataset) using the trained model. This measure takes into account both the prediction error as well as the uncertainty. As a baseline to compare our results against, we use an i.i.d. Gaussian model with a fixed mean and standard deviation computed from the training data. As can be seen from Figure 2b, our conditional model outperforms the Gaussian model for all the modeled states. To visualize the conditional nature of the errors, we plot errors as a function of location of the objects from the sensor in Figure 3. Although the error depends on many features, to illustrate the conditional nature of the error we plot the error in x-direction as a function of the object locations. To obtain these errors, we moved an object with a high-precision sensor around the robot that sensed the position of the object with the sensing module. The residual between the sensed position and the ground truth position is averaged at each location. As we can see in Figure 3, the error depends on the object's location and our model is able to capture the conditional nature of the error. We also qualitatively verify our results in Figure 4 by observing that the i.i.d. fails to model the correlation, producing a jittery trajectory. Conversely, the conditional model is closer to the real data (trajectory is smoother despite the error being large). These results show that the proposed method produces smooth trajectories similar to the real data, models the conditional nature of the measurement errors, and outperforms the baseline.

References

[1] DARPA grand challenge. http://archive.darpa.mil/grandchallenge/, 2004. 1

[2] Kerstin Dautenhahn. Socially intelligent robots: dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 2007. 1

[3] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular open robots simulation engine: MORSE. In *IEEE International Conference on Robotics and Automation*, 2011. 1

[4] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. *arXiv preprint arXiv:1605.06457*, 2016. 1

[5] Serena Ivaldi, Vincent Padois, and Francesco Nori. Tools for dynamics simulation of robots: a survey based on user feedback. *arXiv preprint arXiv:1402.7050*, 2014. 1

[6] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Proc. NIPS*, 2017. 1

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014. 2

[8] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, 1997. 1

[9] Leon lajpah. Simulation in robotics. *Math. Comput. Simul.*, 2008. 1

[10] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, and Rüdiger Dillmann. OpenGRASP: A toolkit for robot grasping simulation. In *Simulation, Modeling, and Programming for Autonomous Robots*, 2010. 1

[11] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. *IEEE International Conference On Neural Networks*, 1994. 1

[12] E. Rohmer, S. P. N. Singh, and M. Freese. V-REP: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013. 1

[13] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CVPR*, 2017. 1

[14] Aaron Staranowicz and Gian Luca Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*, 2011. 1

[15] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *http://arxiv.org/abs/1804.10332*, 2018. 1

[16] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *http://arxiv.org/abs/1609.03499*, 2016. 2