Computer vision, perspectives from Machine Learning

"the pixels-to-torques problem where an RL agent learns a closed-loop control policy (torques) from pixel information only."

UPPSALA
UNIVERSITET

Thomas Schön
Division of Systems and Control
Department of Information Technology
Uppsala University.

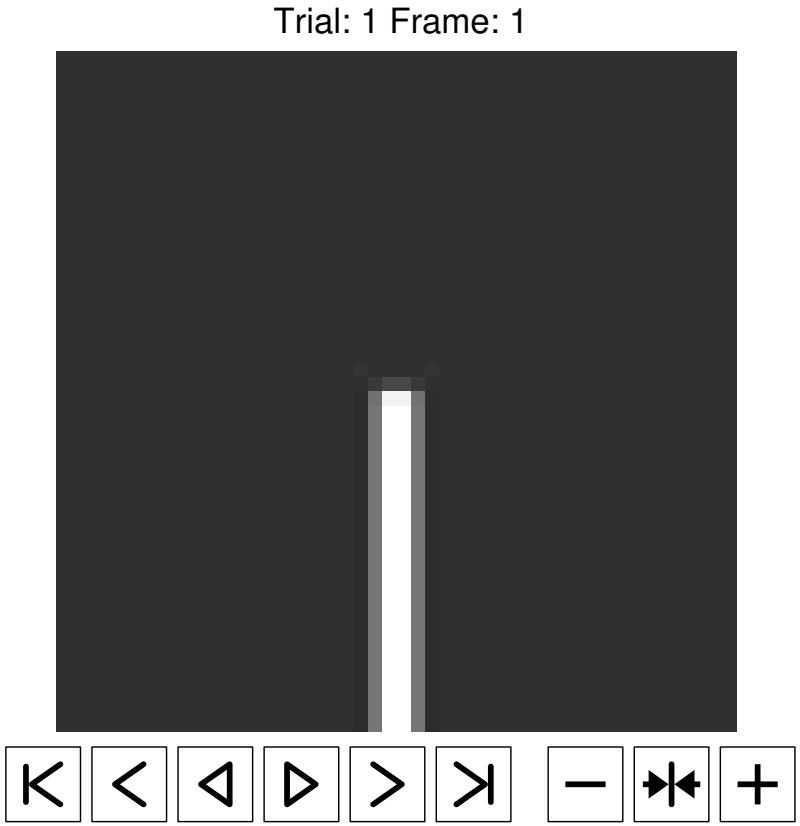
Email: thomas.schon@it.uu.se,
www: `user.it.uu.se/~thosc112`

Joint work with: **John-Alexander M. Assael** (University of Oxford), **Marc Deisenroth** (Imperial College), **Joel Kronander** (Linköping University), **Jonas Unger** (Linköping University), and **Niklas Wahlström** (Linköping University).

UPPSALA UNIVERSITET

A recent example

First steps towards an autonomous system that learns by itself from raw pixel data.



- Deep autoencoder network + nonlinear dynamical model
- Learn not only a model, but **also** how to collect new data to achieve a goal.
- Deepmind has very cool work in this direction.

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

UPPSALA UNIVERSITET

Another recent example

Automatically learn how to describe the contents of images.

Illustrates the **modularity** of the autoencoder, consisting of an **encoder** (vision deep CNN) and a **decoder** (language generating RNN).

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

Xu, K., Lei Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R. Richard S. Zemel, R. S., and Bengio, Y. **Show, attend and tell: neural image caption generation with visual attention**. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July, 2015.

UPPSALA
UNIVERSITET

A few examples where it failed

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Again, a very nice achievement and the fact that they show where it fails is honest.

UPPSALA UNIVERSITET

Outline

Introduction via two recent DL applications

1. **Deep** reinforcement **learning** via an example
   a) Pixels-to-torques problem
   b) Deep autoencoder
   c) Deep dynamical model
2. **Probabilistic programming** – opening up for generative CV?
3. **Rendering** – building a Markov chain in path space

Deep neural networks

A **neural network (NN)** is a nonlinear function $y = g_\theta(u)$ from an input variable u to an output variable y parameterized by θ.

A sequential construction of several generalized linear regressions,

$$z^{(l+1)} = f^{(l)}\left(W^{(l+1)} z^{(l)} + b^{(l+1)}\right), \quad z^{(0)} = u.$$

Think of this as **multiple levels of representation (features)**.

Key aspect: The layers are **not** designed by humans, they are learned from (typically lots of) data.

Key idea: (10 years old) Initialization by training each layer individually using an **unsupervised algorithm**.

Problem Formulation

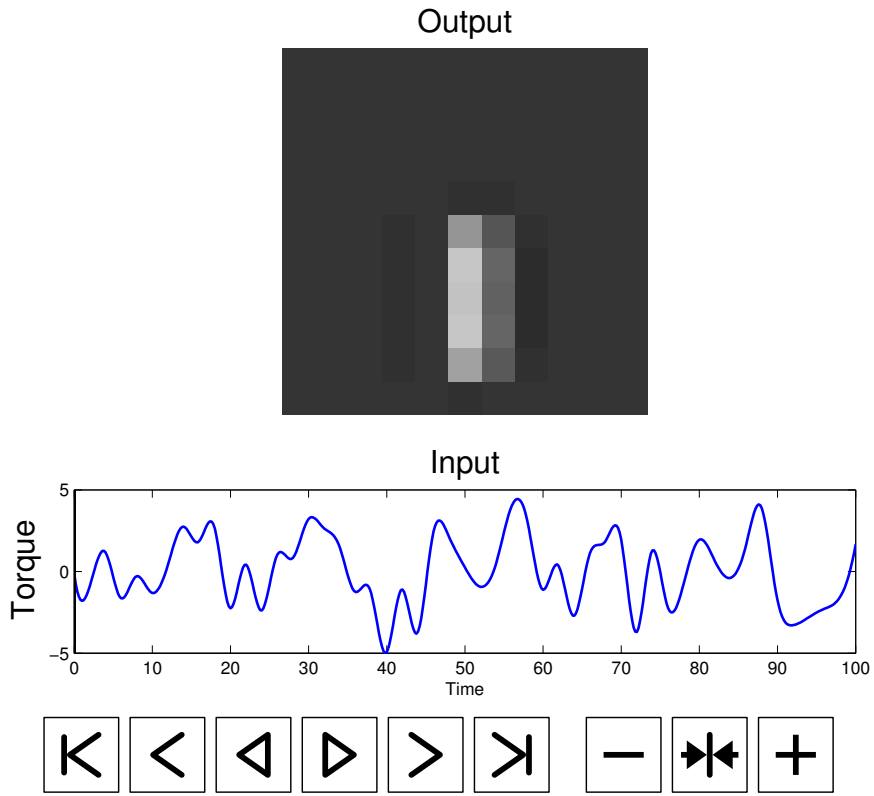
Vision: Systems learning by themselves from raw pixel data.

Problem formulation: Modeling of high-dim. pixel data

Strategy: Construct a **deep dynamical model** that contains a low-dimensional dynamical model.

Example: Video stream of a pendulum

- **Input:** Torque of a pendulum
- **Output:** Pixel values of an 11×11 image



Model component – deep autoencoder

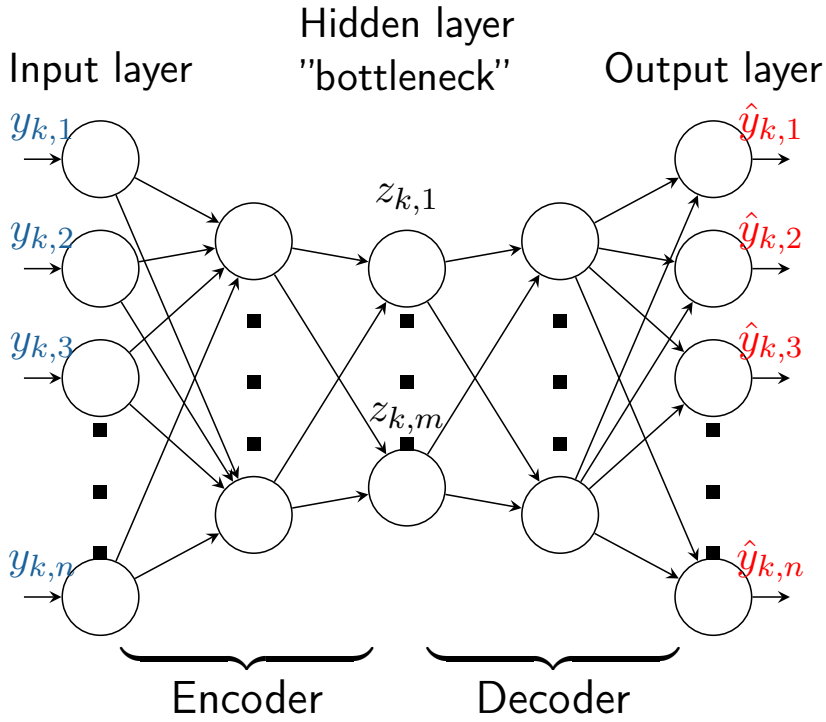
Unsupervised learning procedure for **dimensionality reduction**.

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\dim(\mathbf{y}_k) \gg \dim(\mathbf{z}_k)$

Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{g}^{-1}(\mathbf{y}_k; \boldsymbol{\theta}_{\mathsf{E}})$
2. Decoder: $\widehat{\mathbf{y}}_k^{\mathsf{R}} = \mathbf{g}(\mathbf{z}_k; \boldsymbol{\theta}_{\mathsf{D}})$



Reconstruction error:

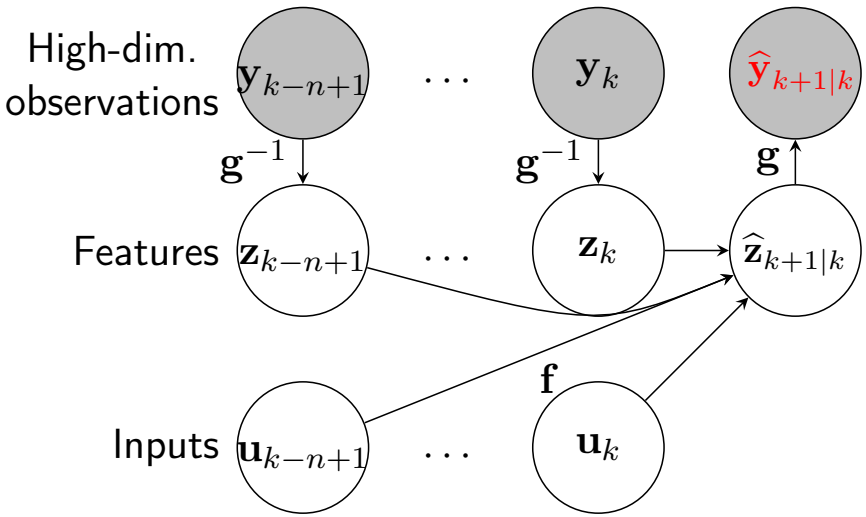
$$V_{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}) = \sum_{k=1}^{N} \|\mathbf{y}_k - \widehat{\mathbf{y}}_k^{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}})\|^2$$

UPPSALA UNIVERSITET

Deep Dynamical Model

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{g}^{-1}(\mathbf{y}_k; \boldsymbol{\theta}_{\mathsf{E}})$
2. Prediction model: $\widehat{\mathbf{z}}_{k+1|k} = \mathbf{f}(\mathbf{z}_k, \mathbf{u}_k, \ldots, \mathbf{z}_{k-n+1}, \mathbf{u}_{k-n+1}; \boldsymbol{\theta}_{\mathsf{P}})$
3. Decoder: $\widehat{\mathbf{y}}^{\mathsf{P}}_{k+1|k} = \mathbf{g}(\widehat{\mathbf{z}}_{k+1|k}; \boldsymbol{\theta}_{\mathsf{D}})$

Prediction error:

$$V_{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}}) = \sum_{k=n}^{N-1} \|\mathbf{y}_{k+1} - \widehat{\mathbf{y}}^{\mathsf{P}}_{k+1|k}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}})\|^2$$

Training

Key ingredient: The reconstruction error and the prediction error are minimized *simultaneously*!

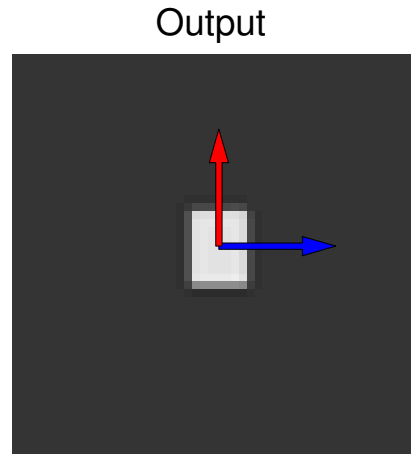
$$\left(\widehat{\boldsymbol{\theta}}_{\mathsf{E}}, \widehat{\boldsymbol{\theta}}_{\mathsf{D}}, \widehat{\boldsymbol{\theta}}_{\mathsf{P}}\right) = \underset{\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}}}{\arg\min}\ V_{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}) + V_{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}})$$

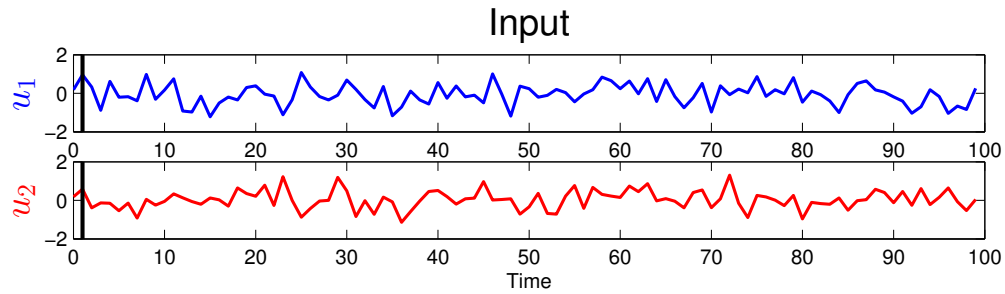
$$V_{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}) = \sum_{k=1}^{N} \|\mathbf{y}_k - \widehat{\mathbf{y}}_k^{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}})\|^2,$$

$$V_{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}}) = \sum_{k=n}^{N-1} \|\mathbf{y}_{k+1} - \widehat{\mathbf{y}}_{k+1|k}^{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}})\|^2.$$

Experiment: Agent in a Planar System

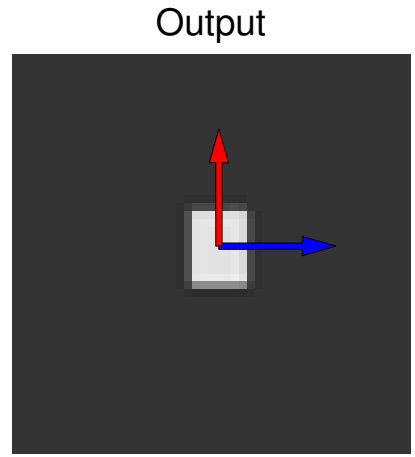
- **Input:** Offset in x−dir. (u_1) and y−dir. (u_2)
- **Output:** Pixel values of a 51×51 image
- **Latent dim.:** dim($\mathbf{z}$)=2

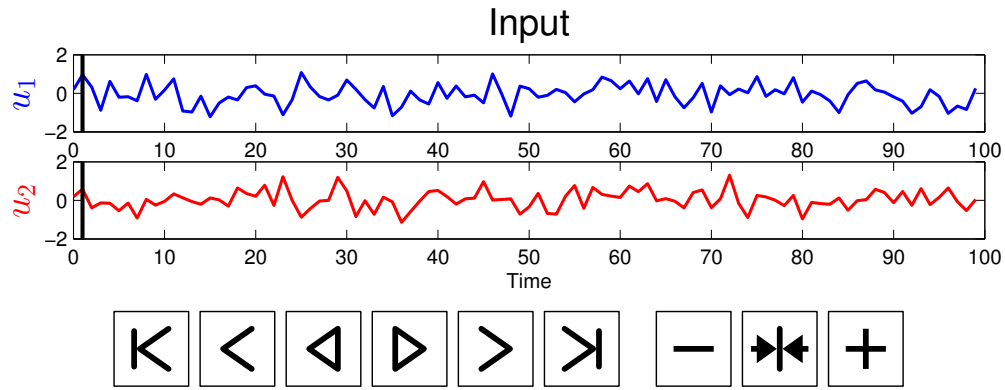




Experiment: Agent in a Planar System

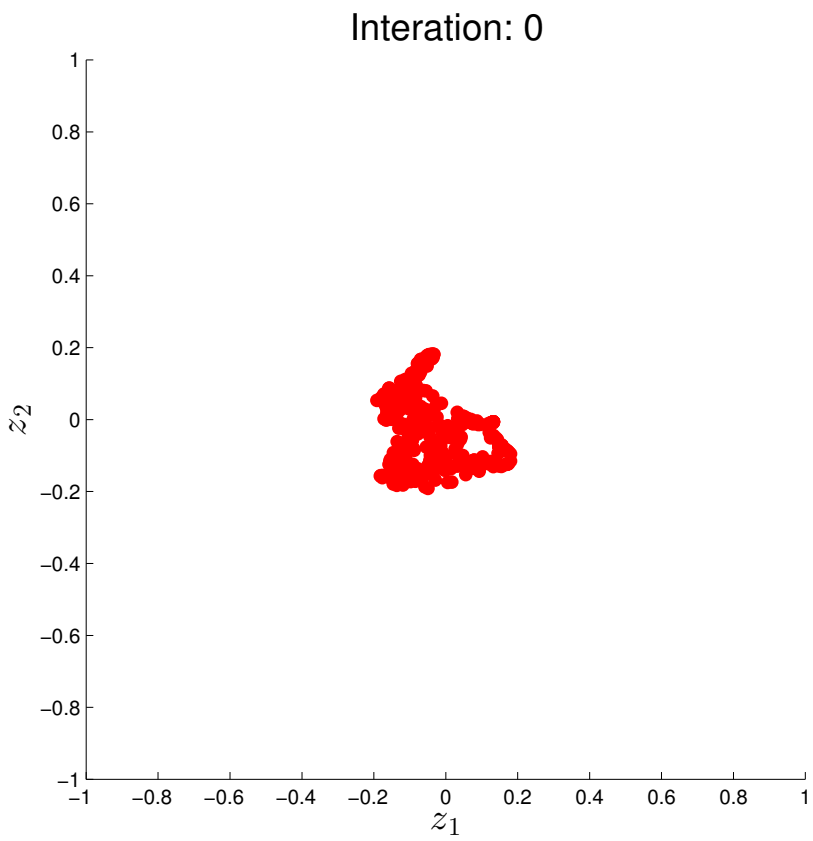
- **Input:** Offset in x−dir. (u_1) and y−dir. (u_2)
- **Output:** Pixel values of a 51×51 image
- **Latent dim.:** dim($\mathbf{z}$)=2



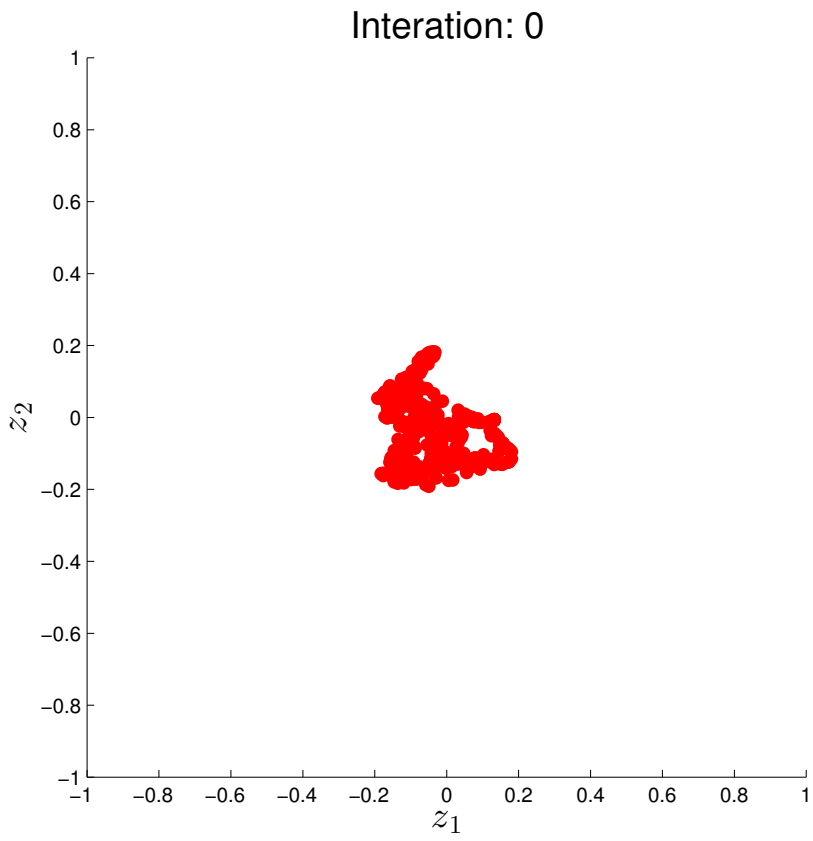


Experiment: Agent in a Planar System

Simultaneous Training

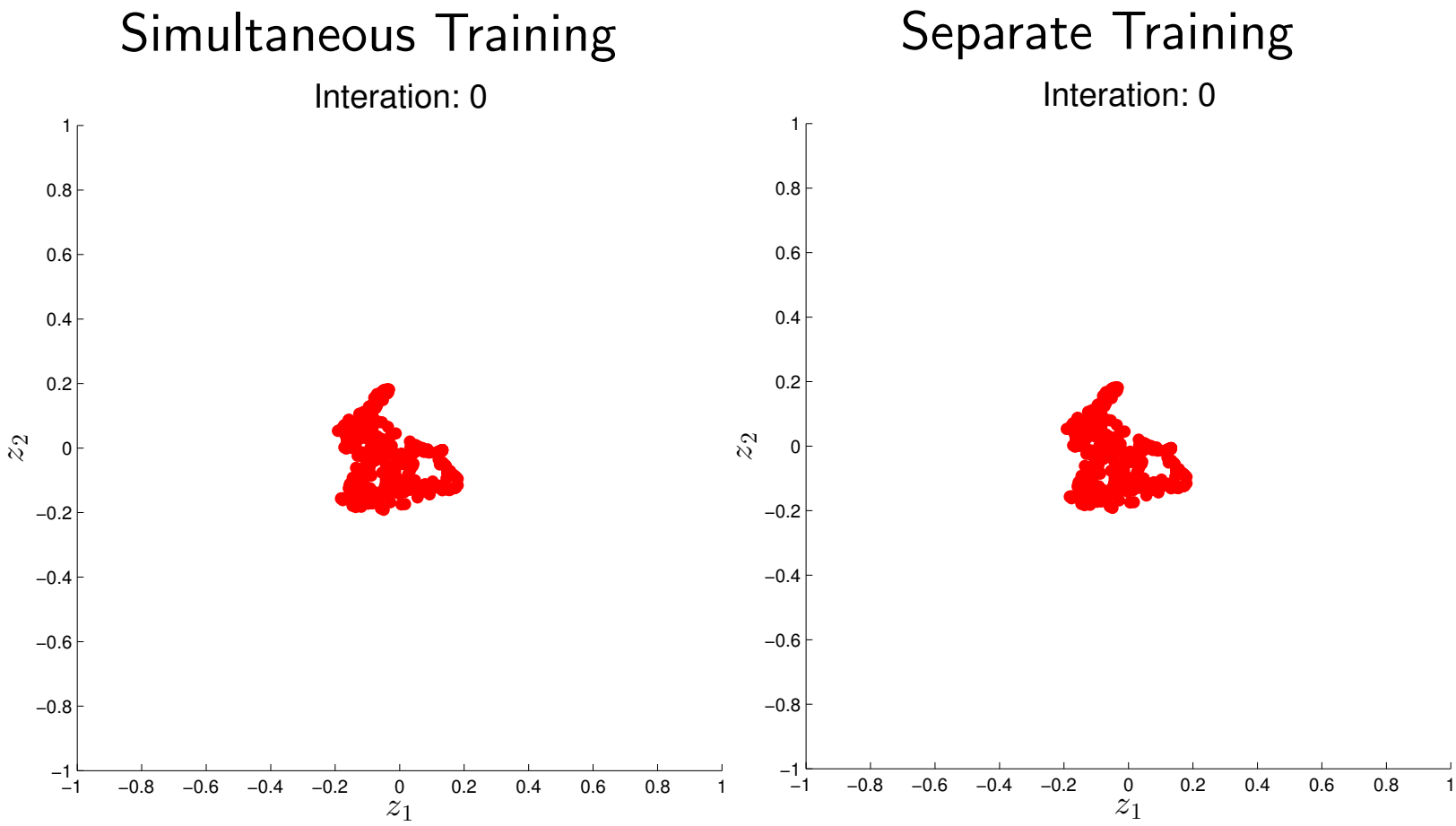


Separate Training



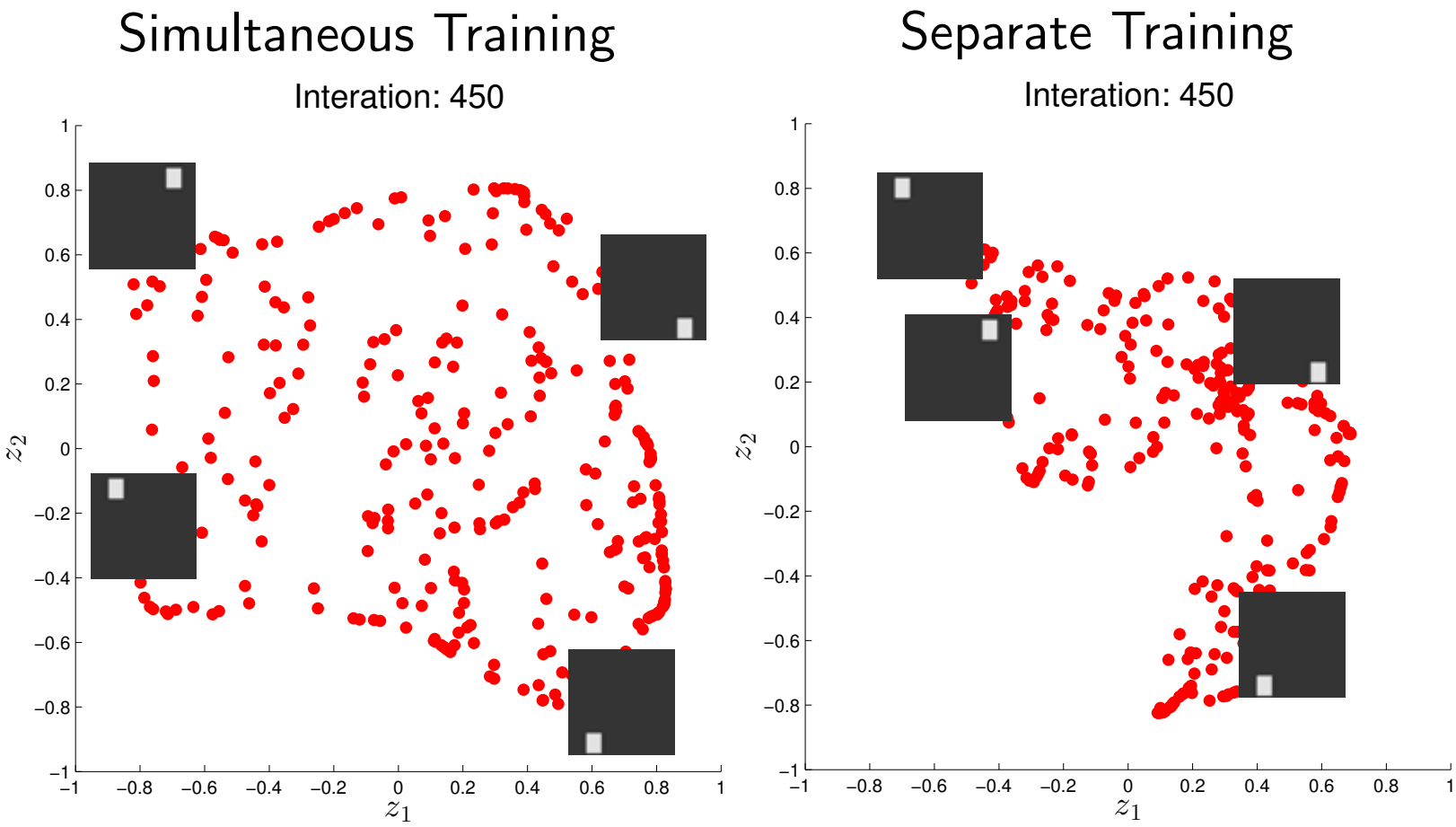
Simultaneous training means that we are training the network (features) jointly with the dynamics.

Experiment: Agent in a Planar System



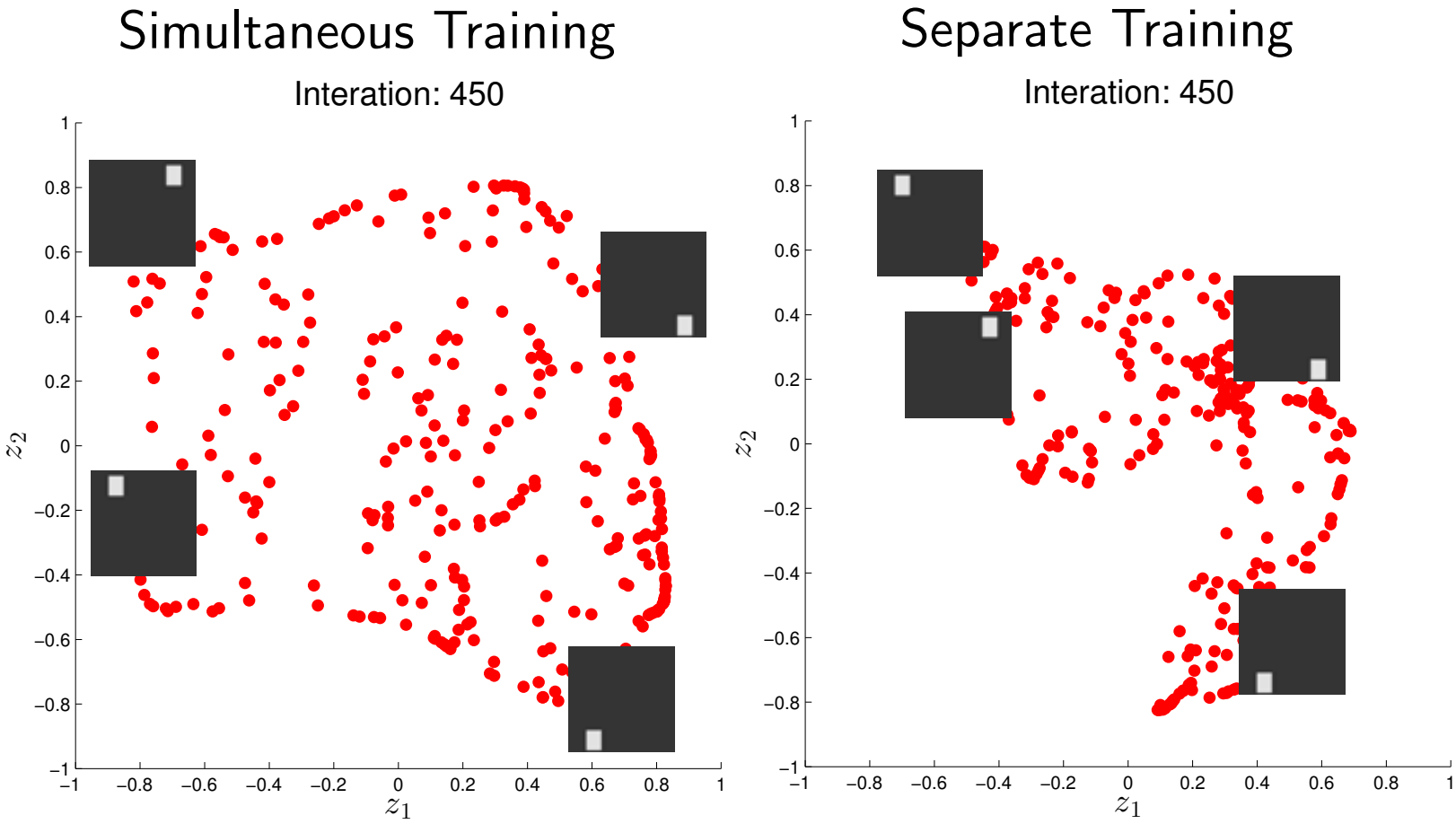
Simultaneous training means that we are training the network (features) jointly with the dynamics.

Experiment: Agent in a Planar System



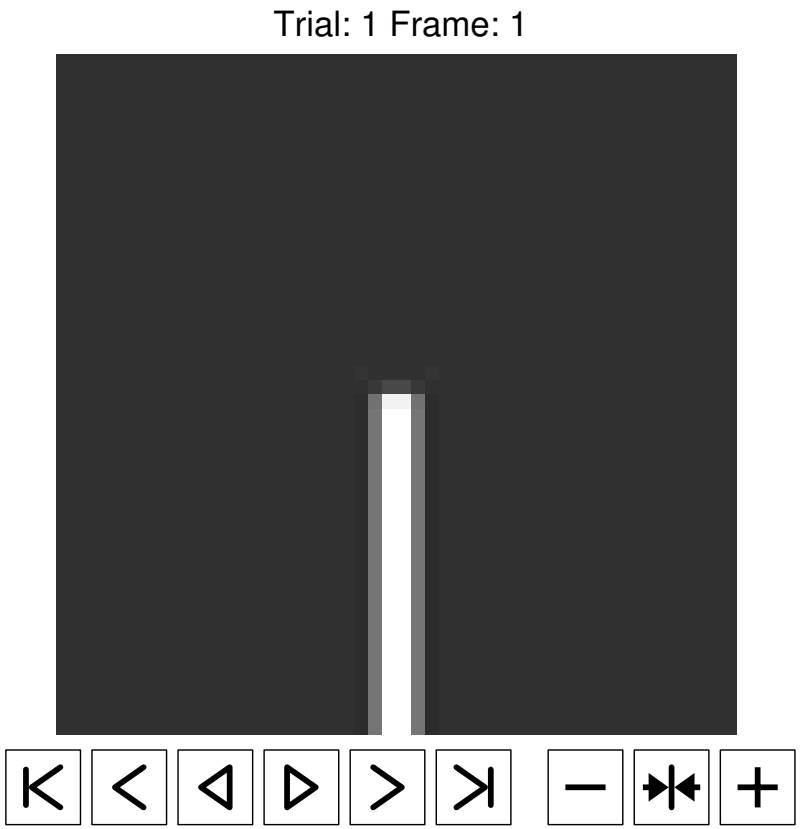
Simultaneous training means that we are training the network (features) jointly with the dynamics.

Experiment: Agent in a Planar System



Simultaneous training means that we are training the network (features) jointly with the dynamics.

Control of Two-Link Arm from Pixels Only



- Deep autoencoder network + nonlinear SSM
- Model predictive control (MPC)
- Ref. value: $\mathbf{z}_{\text{ref}} = \mathbf{g}^{-1}(\mathbf{y}_{\text{ref}})$
- The model is automatically improved (iterative way)

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

UPPSALA UNIVERSITET

Outline

Introduction via two recent DL applications

1. Deep reinforcement learning via an example
   a) Pixels-to-torques problem
   b) Deep autoencoder
   c) Deep dynamical model
2. **Probabilistic programming** – opening up for generative CV?
3. **Rendering** – building a Markov chain in path space

Probabilistic programming

Probabilistic programming makes use of computer programs to represent probabilistic models.

Probabilistic programming lies on the interesting intersection of

1. Programming languages: Compilers and semantics.
2. Machine learning: Algorithms and applications.
3. Statistics: Inference and theory.

Creates a clear **separation** between the model and the inference methods, encouraging model based thinking. Potential to automate inference!

Infer the inputs (in particular the stochastic parts) that match a certain program output. *Why not implement a computer vision system as the inverse of a computer graphics program?*

Vision as "inverse graphics"

As with most ideas they have been around for a long time

Grenander, U. **Pattern Synthesis – Lectures in pattern theory**. Springer, New York, 1976.

Horn, B. K. P. **Understanding image intensities**. *Artificial Intelligence*, 8:201–231, 1977.

Consider an unknown scene description θ that produce an image

$$I = R(\theta),$$

using a rendering engine R. Based on observations of the images $\{J_k\}_{k=1}^{K}$ and a prior $p(\theta)$ compute the posterior $p(\theta \mid J_{1:K})$.

UPPSALA UNIVERSITET

Vision as "inverse graphics"

Not delivered yet, but the idea is still young. Key challenges:

1. Constructing **generative models**. Use computer graphics and the artists way of working here, "procedural modeling".
2. Solve the **inference** problem. Use new Monte Carlo and/or variational methods.

(Really) interesting introduction of prob. prog. into the vision area:

Tejas D. Kulkarni, T. D., Kohli, P. Tenenbaum, J. and Mansinghka, V. **Picture: An Imperative Probabilistic Programming Language for Scene Perception**, CVPR, 2015 (best paper honorable mention)

Mansinghka, V. K., Kulkarni, T. D., Perov, Y. N. and Tenenbaum, J. B. **Approximate Bayesian Image Interpretation using Generative Probabilistic Graphics Programs**, NIPS, 2013.

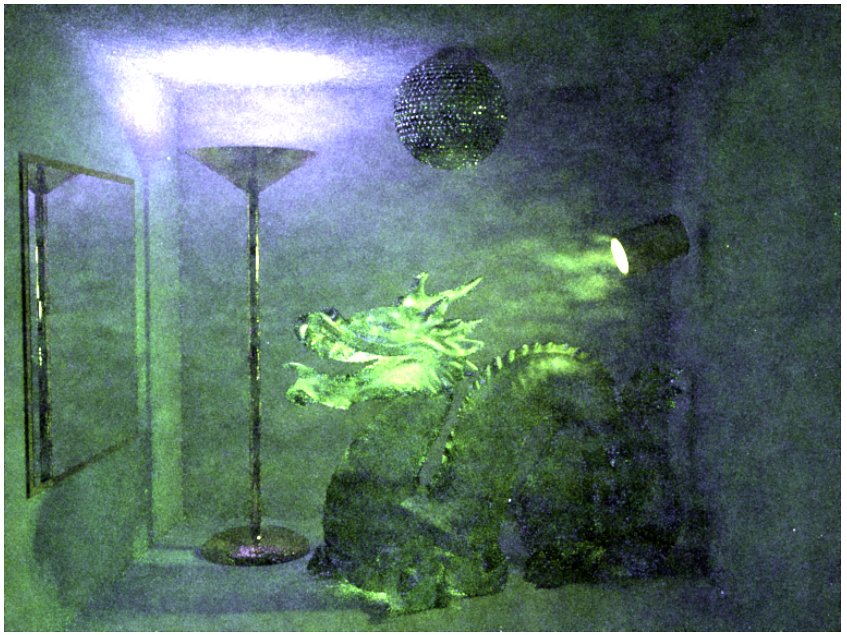
Given the tools we have today, this is a really **exciting idea**!

Rendering in heterogeneous media

A Monte Carlo method that makes use of stochastically sampled light paths connecting the sensor with light sources in the scene.

Results using equal time rendering

Our method that builds on MLT



Metropolis light transport (MLT)

Joel Kronander, Thomas B. Schön and Jonas Unger. **Pseudo-marginal Metropolis light transport**. In *Proceedings of SIGGRAPH ASIA Technical Briefs*, Kobe, Japan, November, 2015.

It comes down to computing integrals

The value I_j of pixel j is expressed as an integral over all possible light paths $\bar{x}$ in the scene (path space $\mathcal{P}$),

$$I_j = \int_{\mathcal{P}} f(\bar{x}) \mathrm{d}\mu(x)$$

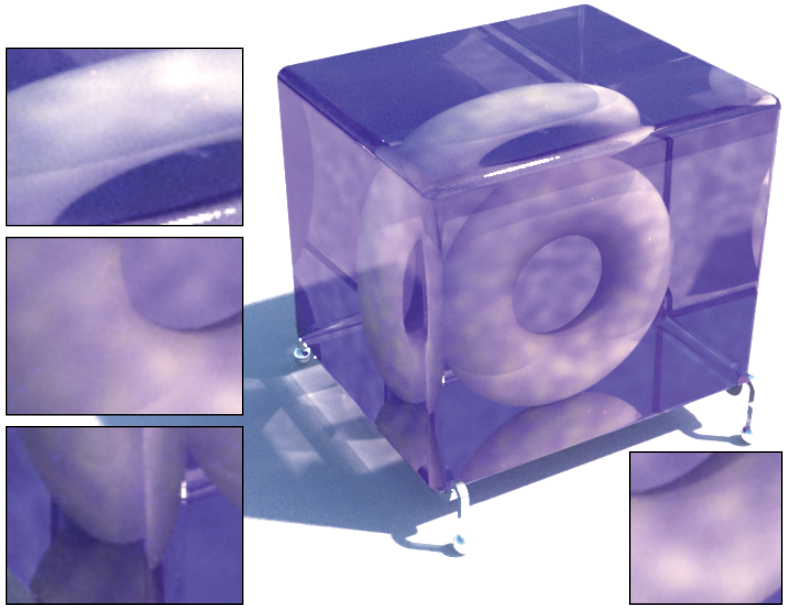
Generate a Markov chain on path space to sample light paths proportional to their contribution $L(\bar{x}) \propto f(\bar{x})$.

Pseudo-marginal Metropolis Hastings allows for **exact** computation of the target distribution, despite the use of an **unbiased estimator** $\widehat{L}(\bar{x})$ for $L(\bar{x})$.

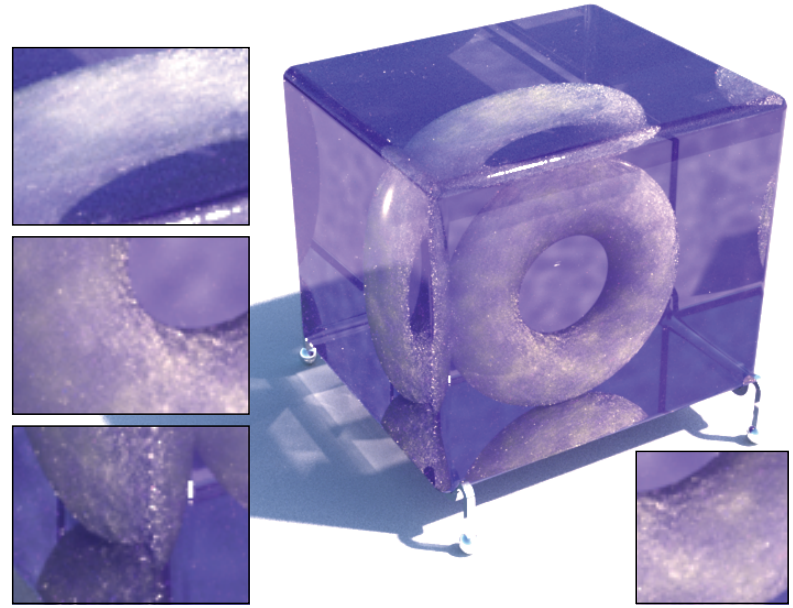
Christophe Andrieu and Gareth O. Roberts, **The pseudo-marginal approach for efficient Monte Carlo computations**, *The Annals of Statistics*, 37(2):697-725, 2009.

One more example

Results using equal time rendering



Our pseudo-marginal ERPT



ERPT

Cline, D., Talbot, J., and Egbert, P. **Energy redistribution path tracing**. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):1186–1195, 2005.

UPPSALA UNIVERSITET

Conclusions

Introduced a **deep dynamical model**

Use the model also for "attention" and control

Reinforcement learning to decide **where** to look for new data (resulting in new knowledge).

Generative models + MC/VI = promising and **exciting**

Built a Markov chain to sample light paths

Used the chain to **render images** in heterogeneous media.

Two additional trends that we can talk about some other time:

1. Bayesian nonparametric models, like (GP, DP, etc.)
2. Bayesian optimization

Remember to talk to people who work on different problems with different tools!!