Robot Learning and Control 7

Diffusion Models

Takamitsu Matsubara[1]

[1]Robot Learning Lab

Dec. 20, 2024

Table of Contents

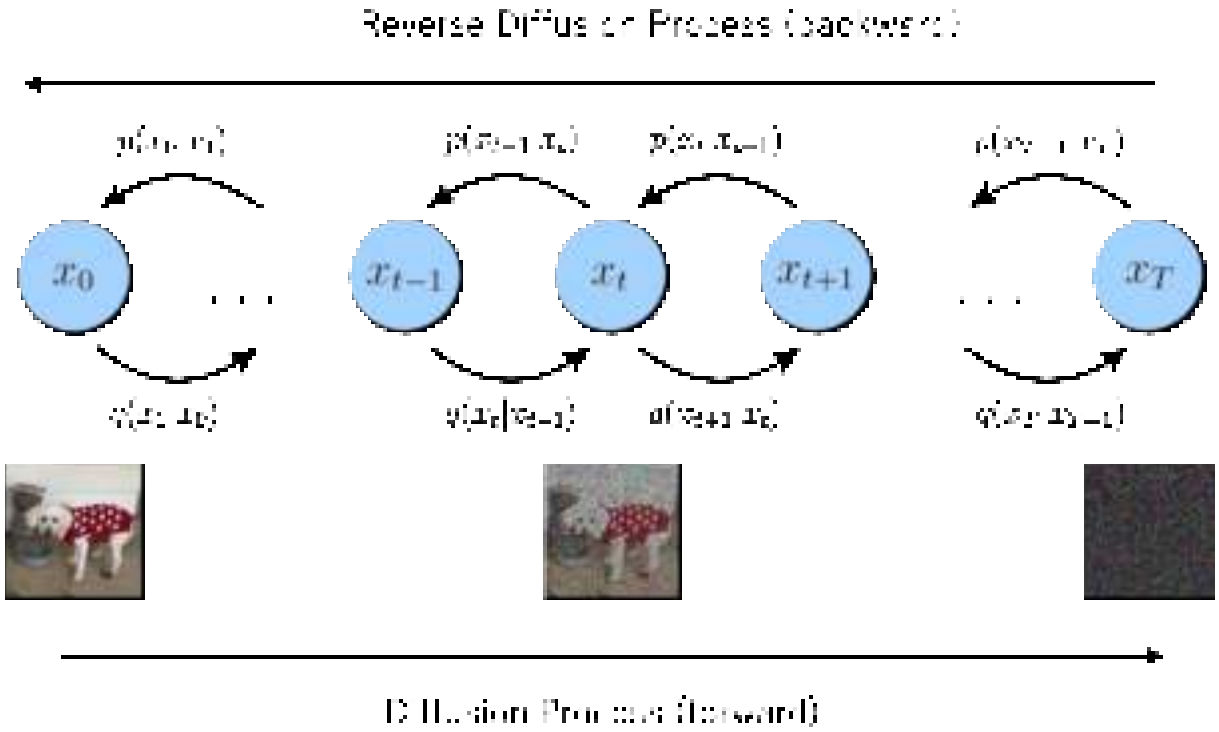
Stable Diffusion (2022)

Stable Diffusion is an image-generating AI service developed by Stability AI based in the United Kingdom. Simply by inputting a prompt, you can easily create images ranging from realistic to anime-style.

https://stablediffusionweb.com/#ai-image-generator

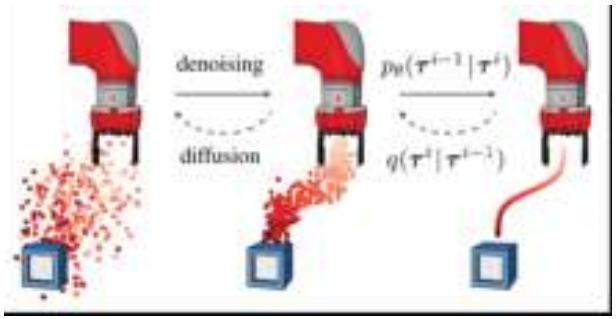
What is Diffusion Models?

- Diffusion models in machine learning refer to a class of generative models used for modeling complex probability distributions, especially in the context of image, video, or data generation tasks.
- These models are based on simulating a (reverse) diffusion process to generate high-quality samples, that is learned from a given dataset.

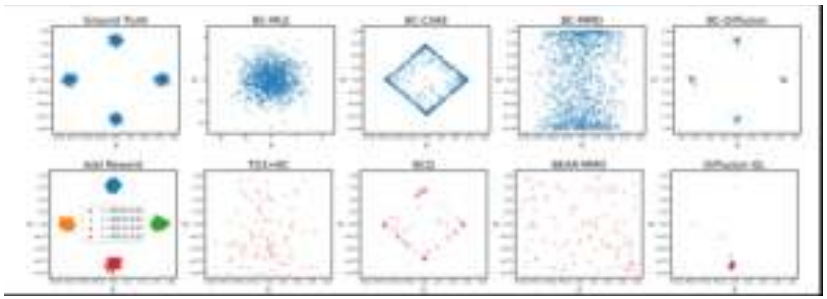


Why Diffusion for Robot Learning and Control?

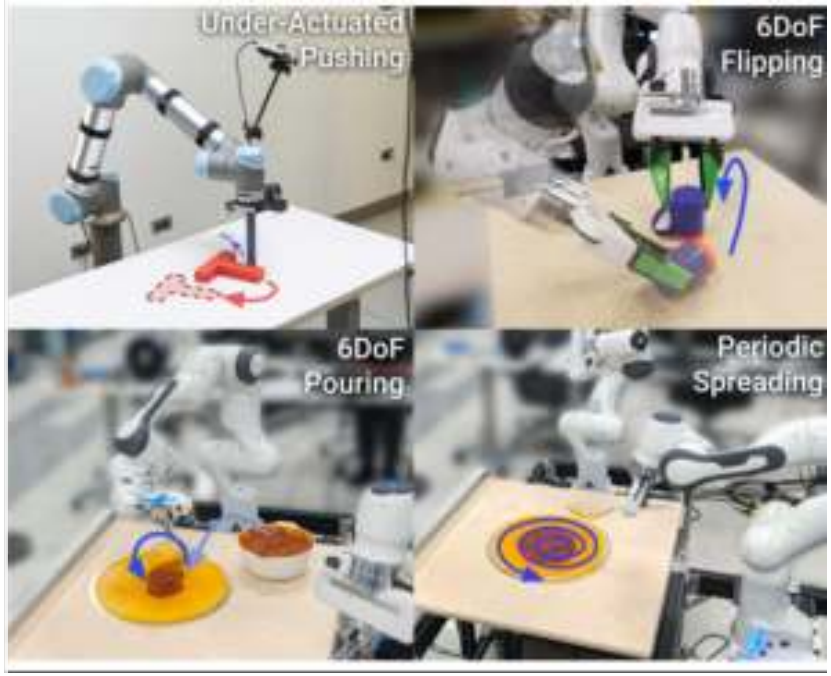
- Diffusion models can be used for robot learning and control.
- Be used to generate either action, dynamics, or value functions in reinforcement learning with richer representations than before.



Model based Reinforcement Learning, ICML2022



Off-line Reinforcement Learning, ICLR2023



Imitation Learning, RSS2023

Table of Contents

Overview of Diffusion Models

Strong features:

- Learning is simple and stable (maximum likelihood rather than EM)
- Natural partitioning of difficult generative problems into easy problems
- Easy to generate samples by conditioning ($p(x)$ to $p(x|y)$)

Two equivalent modeling approaches:

- Score-based Models (SBMs) (Song and Ermon, 2019, 2020)
- Denoising Diffusion Probabilistic Models (DDPMs) (Ho, 2020)

In this lecture, we focus on DDPMs because it is strongly related to Maximum Likelihood, Linear Gaussian State Space Models, Jensen's inequality, EM algorithm (what you have studied!)

Modeling: Forward diffusion process

Assuming $\mathbf{x}_0$ is data, and $\mathbf{x}_1, \cdots, \mathbf{x}_T$ are obtained by adding noise by the following diffusion process.

Definition of diffusion process:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad (1)$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}_t) \quad (2)$$

where $0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$, $\alpha_t = 1 - \beta_t$ so that the data are destroyed gradually to be following $q(\mathbf{x}_t|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ eventually.

Note that the forward diffusion process is a product of linear Gaussian dynamics!

Modeling: Forward diffusion process

Linear Gaussian Dynamics

The next-time state is deviated by the linear mapping from current state and zero-mean Gaussian noise:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_t \quad (1)$$

$$\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I}) \quad (2)$$

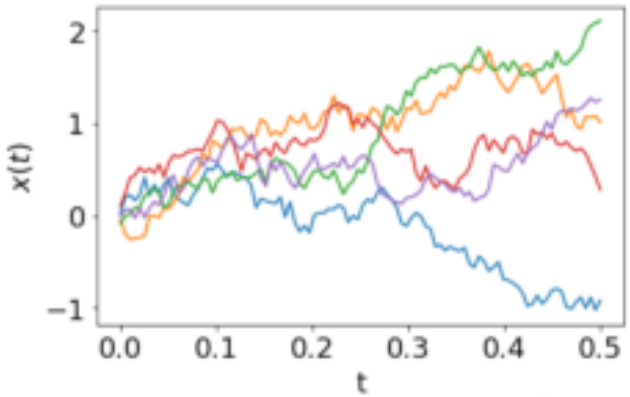


Figure: State trajectory with $A = 1, \sigma^2 = 0.1, \sigma_0^2 \approx 0$

Diversity of state expands over time → prediction becomes harder

Let's confirm this point in the model

Modeling: Forward diffusion process

Linear Gaussian Dynamics

Given the initial state distribution and linear Guassian dynamics,

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (3)$$

$$p(\mathbf{x}_2 \mid \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_2; \mathbf{A}\mathbf{x}_1, \sigma^2\mathbf{I}) \quad (4)$$

One-step predictive state distribution is obtained as follows:

$$p(\mathbf{x}_2) = \int p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)\mathbf{x}_1 \quad \text{(marginalization)} \quad (5)$$

$$= \mathcal{N}(\mathbf{x}_2; \mathbf{A}\boldsymbol{\mu}, \underbrace{\sigma^2\mathbf{I}}_{\text{process}} + \underbrace{\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T}_{\text{dynamics}}) \quad (6)$$

Both uncertainties in initial state and dynamics are marged, and it can be recursively applied over the time!

If $A \geq 1$, the uncertainty is expanded and expanded over the time!

Modeling: Forward diffusion process

Bayes rule for Gaussian distribution

Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \Lambda^{-1}) \tag{23}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\underbrace{\mathbf{A}\mathbf{x}}_{linear}, \underbrace{\mathbf{L}^{-1}}_{const.}) \tag{24}$$

The marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mu, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T) \tag{25}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\Sigma\{\mathbf{A}^T\mathbf{L}\mathbf{y} + \Lambda\mu\}, \Sigma) \tag{26}$$

where

$$\Sigma = (\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1} \tag{27}$$

Analytic solutions! We will use them later.

Modeling: Forward diffusion process

The marginalized forward diffusion process can also be given by Gaussians as follows;

$$q(\mathbf{x}_t|\mathbf{x}_0) = \int q(\mathbf{x}_{1:t}|\mathbf{x}_0)d\mathbf{x}_{1:t-1} = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \bar{\beta}_t\mathbf{I}) \tag{3}$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \tag{4}$$

$$\bar{\beta}_t = 1 - \bar{\alpha}_t \tag{5}$$

The effects of gradually becoming to the noise for the interval are accumulated.

Modeling: Reverse diffusion (denoising) model

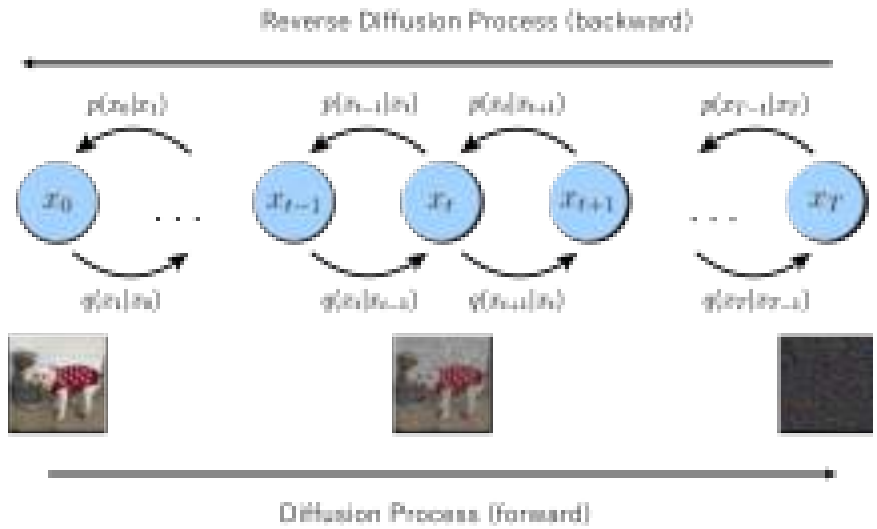
Definition of reverse diffusion (denoising) process:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{6}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}) \tag{7}$$

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I}) \tag{8}$$

where μ_θ is parametrized models (e.g., Neural Network) to be learned from data. See the illustration again:



https://calvinyluo.com/2022/08/26/diffusion-tutorial.html

Learning Reverse Diffusion Models

We want to learn the reverse diffusion model as a data-generative model. Since we only have the data $\mathbf{x}_0$ and others are latent variables, we consider the following Marginal likelihood:

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \tag{9}$$

Let's derive a tractable lower bound using the forward diffusion process:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &= -\log \int \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} && (10)\\
&\leq E_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \text{ (Jensen's inequality)} && (11)\\
&= E_q \left[-\log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1) p_\theta(\mathbf{x}_1|\mathbf{x}_2) p_\theta(\mathbf{x}_2|\mathbf{x}_3) \cdots p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_{T-1}) q(\mathbf{x}_{T-1}|\mathbf{x}_{T-2}) \cdots q(\mathbf{x}_1|\mathbf{x}_0)} \right] && (12)\\
&= E_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] = L(\theta) && (13)
\end{aligned}$$

Learning Reverse Diffusion Models (cont'd)

Since q holds the Markov property, it allows to modify $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)$. Therefore,

$$L(\theta) = E_q \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right]$$

By notifying $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t|\mathbf{x}_0) / q(\mathbf{x}_{t-1}|\mathbf{x}_0)$, we can further expand it as:

$$= E_q \left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \quad (14)$$

$$= D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) (\to L_T) \quad (15)$$

$$+ \sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) (\to L_{t-1}) \quad (16)$$

$$- E_{q_0} [\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] (\to L_0) \quad (17)$$

Focus on L_{t-1}, measures the diff betweeen $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Learning Reverse Diffusion Models (cont'd)

To understand L_{t-1} deeper, let's consider the posterior of the forward diffusion:

$$
\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &\propto q(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{x}_0) && (18)\\
&= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1}|\mathbf{x}_0) && (19)\\
&= q(\mathbf{x}_t|\mathbf{x}_{t-1}) q(\mathbf{x}_{t-1}|\mathbf{x}_0) && (20)
\end{aligned}
$$

This is the linear Gaussian's posterior, so we can obtain the analytic solution:

$$
\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) && (21)\\
\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{\tilde{\beta}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\bar{\beta}_{t-1}}{\bar{\beta}_t}\mathbf{x}_t && (22)\\
\tilde{\beta}_t &= \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t}\beta_t && (23)
\end{aligned}
$$

The mean of $\mathbf{x}_{t-1}$ is located in an interior point between $\mathbf{x}_t$ and $\mathbf{x}_0$

Learning Reverse Diffusion Models (cont'd)

$$L_{t-1} = D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \tag{24}$$

$$= E_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C \tag{25}$$

This means that the mean of the posterior for the forward diffusion process is approximated by the mean of revised diffusion process.
Since $\mathbf{x}_t$ in $\tilde{\mu}$ is taken the expectation over $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \bar{\beta}_t\mathbf{I})$, $\mathbf{x}_t$ can be represented by using $\mathbf{x}_0$ and ϵ:

$$\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{\bar{\beta}_t}\epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{26}$$

Thus, by combining multiple equations,

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{\bar{\beta}_t}}\epsilon\right) \tag{27}$$

The mean of the posterior of the diffusion process seems denoising.

Learning Reverse Diffusion Models (cont'd)

For the reverse diffusion model, since the noise is unknown, we design the model of the noise by $\epsilon_\theta(\mathbf{x}_t, t)$. Using that the data can be modeled by the following denoising process:

$$\tilde{\mathbf{x}}_0 = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \sqrt{\beta_t}\epsilon_\theta(\mathbf{x}_t, t)\right) \tag{28}$$

By using this and fitting it into the same functional model as the posterior of the forward diffusion, we model the mean of the reverse diffusion as:

$$\begin{aligned} \mu_\theta(\mathbf{x}, t) &\approx \tilde{\mu}_t(\mathbf{x}_t, \tilde{\mathbf{x}}_0) && (29) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}}\epsilon_\theta(\mathbf{x}_t, t))\right) && (30) \end{aligned}$$

Learning Reverse Diffusion Models (cont'd)

$$
\begin{aligned}
L_{t-1} &= D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) && (31)\\
&= E_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C && (32)\\
&= E_q\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t\bar{\beta}_t}||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{\bar{\beta}_t}\epsilon, t)||^2\right] + C && (33)
\end{aligned}
$$

We can learn the model ϵ_θ by minimizing the above lower bound.
Thus, DDPM learns a model that can first inject noise into the data, then the injected noise is predicted from that of noise-injected data.

Data generation from DDPM

With the learned model of the mean,

$$\mu_\theta(\mathbf{x}, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \tag{34}$$

we can generate data by:
From $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, for $t = T : T-1 : 0$

$$\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma^2 \mathbf{I}) \tag{35}$$

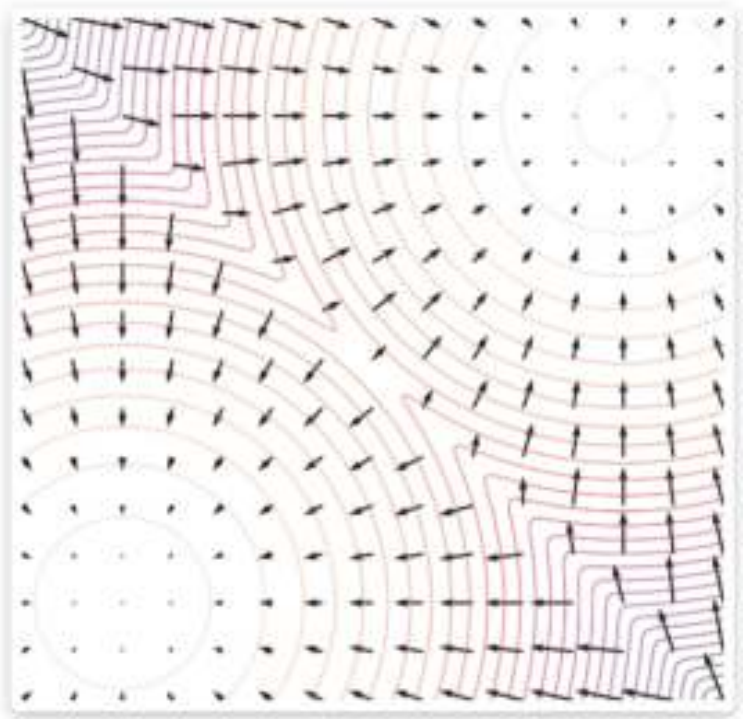
and receive $\tilde{\mathbf{x}}_0$.

DDPM is equivalent to Score-based Model

The denoising (data generation) process with learned DDPM is equivalent to the Langevin Monte Carlo method:

$$x_k = x_{k-1} + \alpha \nabla_x \log p(x_{k-1}) + \sqrt{2\alpha} u_k \tag{36}$$

where, $\log p(x)$ is called **score function**, the gradient of the log-probability density function of data.



Plugin Conditioning for SBM

Conditional distribution:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{37}$$

Log-probability density and its score:

$$\log p(x|y) = \log p(y|x) + \log p(x) - \log p(y) \tag{38}$$

$$\nabla_x \log p(x|y) = \nabla_x \log p(y|x) + \nabla_x \log p(x) \tag{39}$$

where we utilize $\nabla_x \log p(y) = 0$.
This implies that we can replace $\nabla_x \log p(x)$ to $\nabla_x \log p(x|y)$ by simply adding $\nabla_x \log p(y|x) \rightarrow$ **Plugin Conditioning**

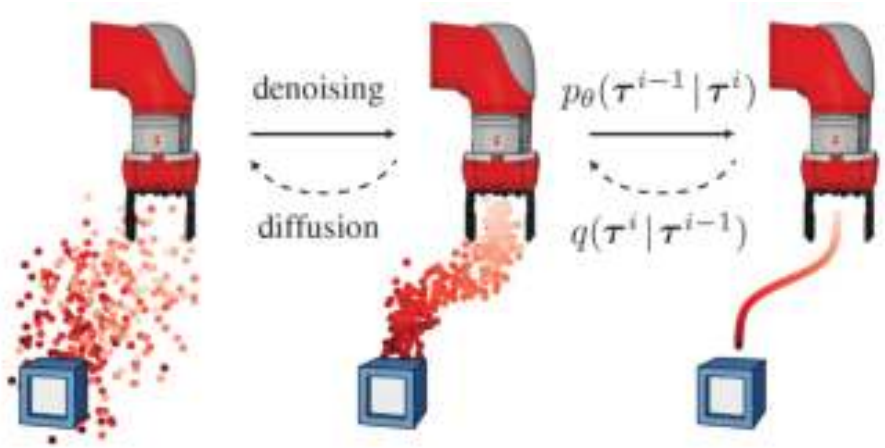
Summary of DDPM

- By using a fixed forward diffusion process, a generative model (reverse diffusion model) can be learned without learning a recognition model, even though it is a latent variable model.
- Since the model uses multiple probability layers, its expressive power can be very large.
- Learning is more stable and can process a larger number of data
- The diffusion model can be seen as learning Score Based Models (SBMs). The compositionality of multiple energy models, which can be freely combined later, is used to perform pulgin-conditioned data generation ($p(x)$ to $p(x|y)$)
- Data generation is slow due to a large number of steps when sampling

Table of Contents

Diffuser: Planning with Diffusion [Janner+, ICML2022]

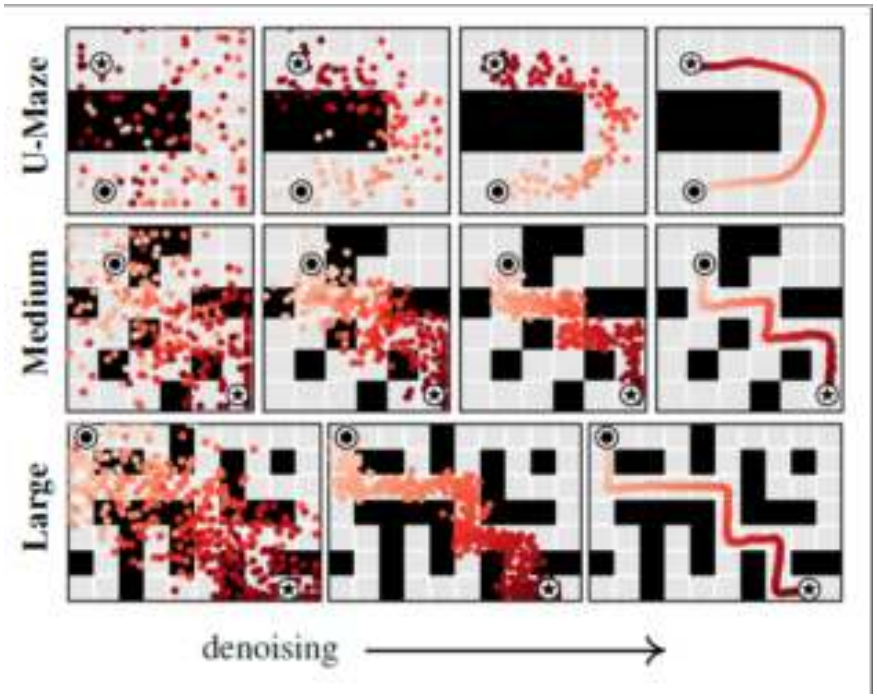
A trajectory-level diffusion probabilistic model called Diffuser.

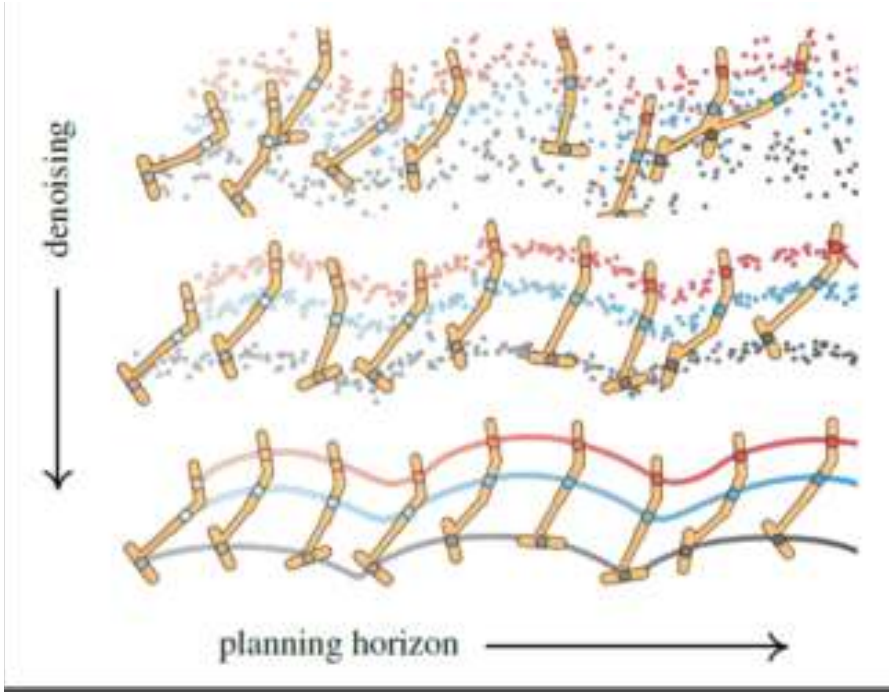


Two key tricks:

- State-action trajectory is directly modeled as the output of diffusion: $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)$ (it was almost impossible before)
- Incorporate reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ by *plugin-conditional* inference: $\tilde{p}_\theta(\tau) = p(\tau|\mathcal{O}_{1:T} = 1) \propto p(\tau)p(\mathcal{O}_{1:T} = 1|\tau)$
  where $\mathcal{O}_t$ is a binary random variable denoting the optimality of timestep t of a trajectory with $p(\mathcal{O}_t = 1|\tau_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$

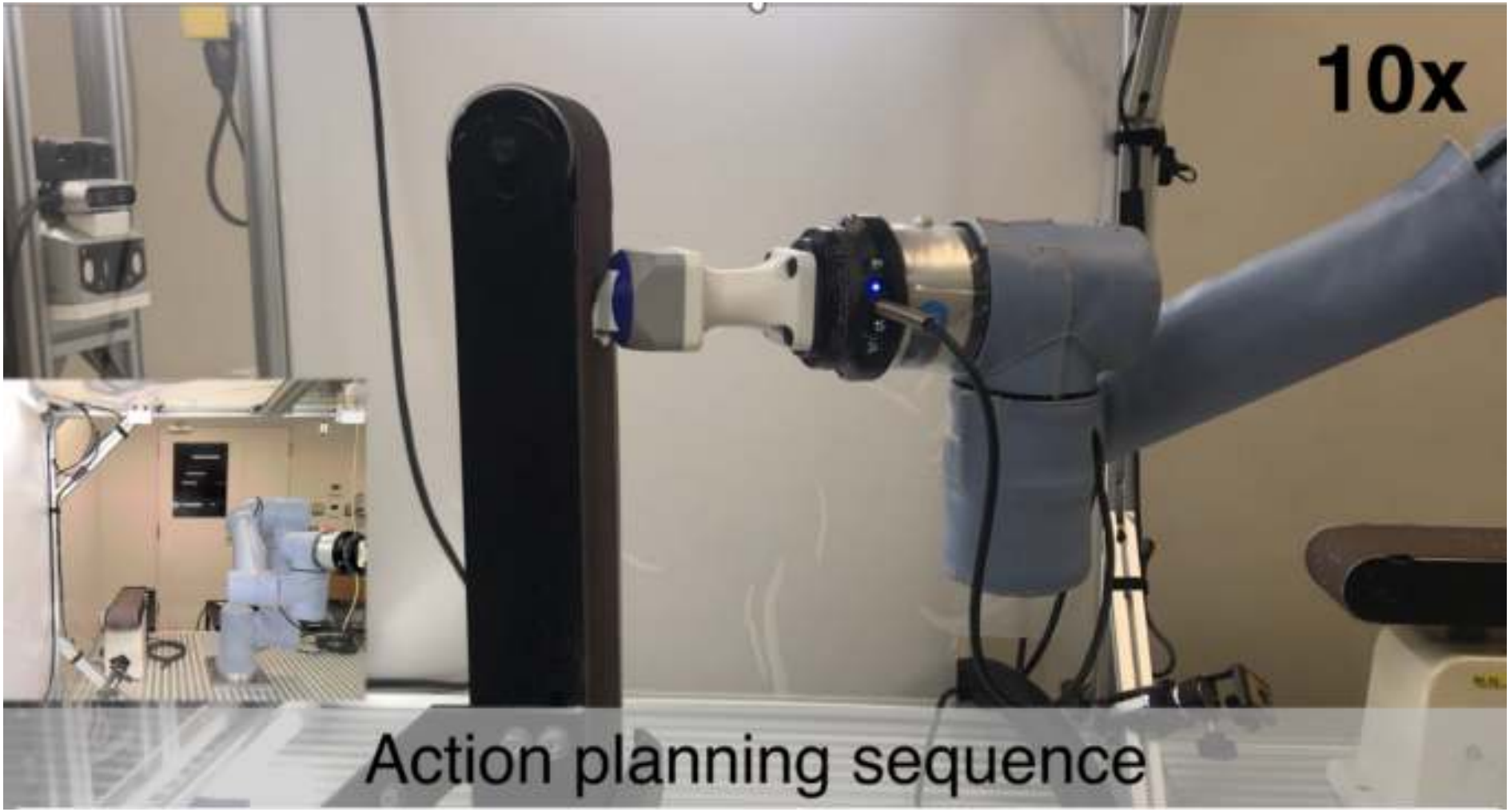
Diffuser: Planning with Diffusion [Janner+, ICML2022]





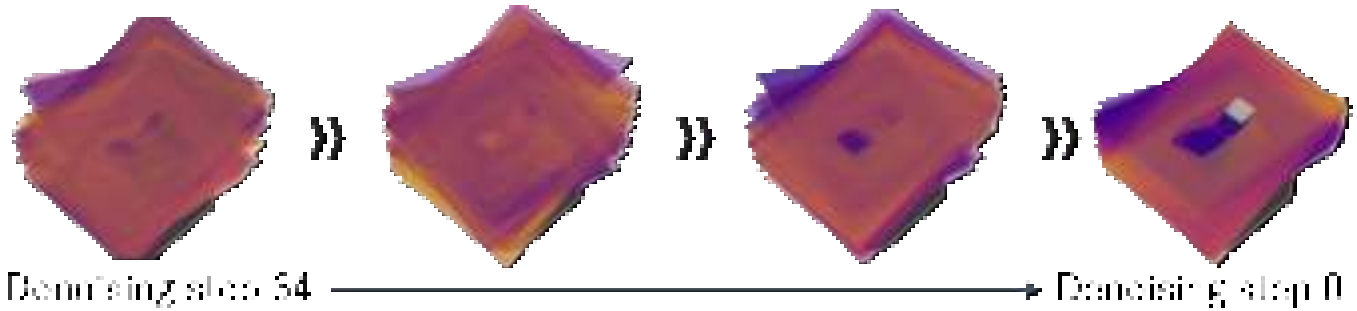
https://diffusion-planning.github.io/

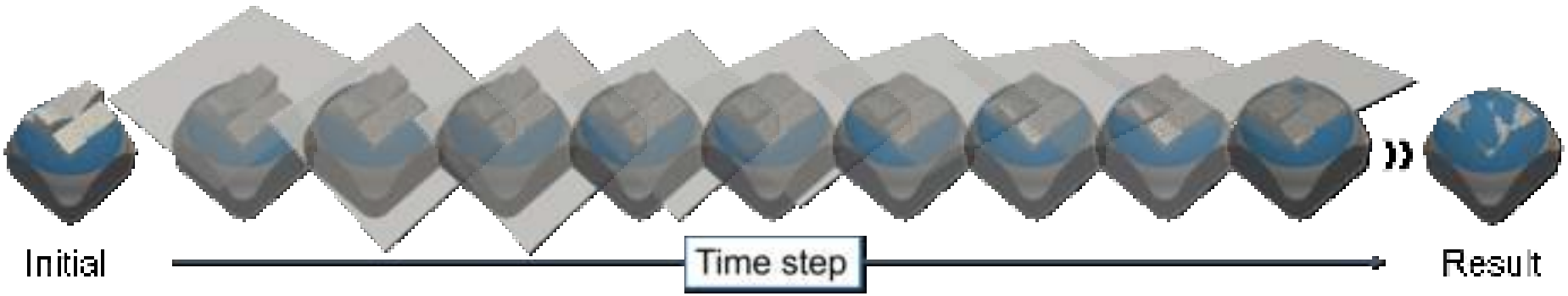
Application to Robotic Grinding [Hachimine et al. 2024]



Application to Robotic Grinding [Hachimine et al. 2024]

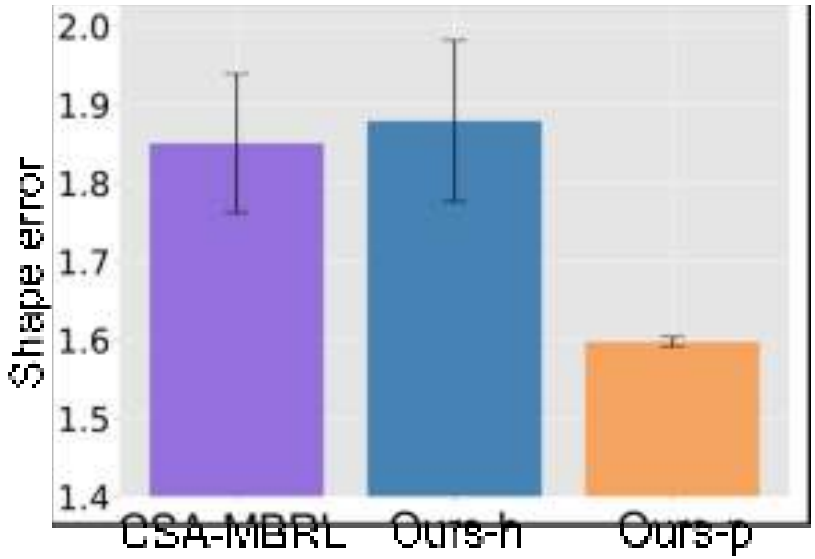
Cutting surface sequence planning by diffusion model.

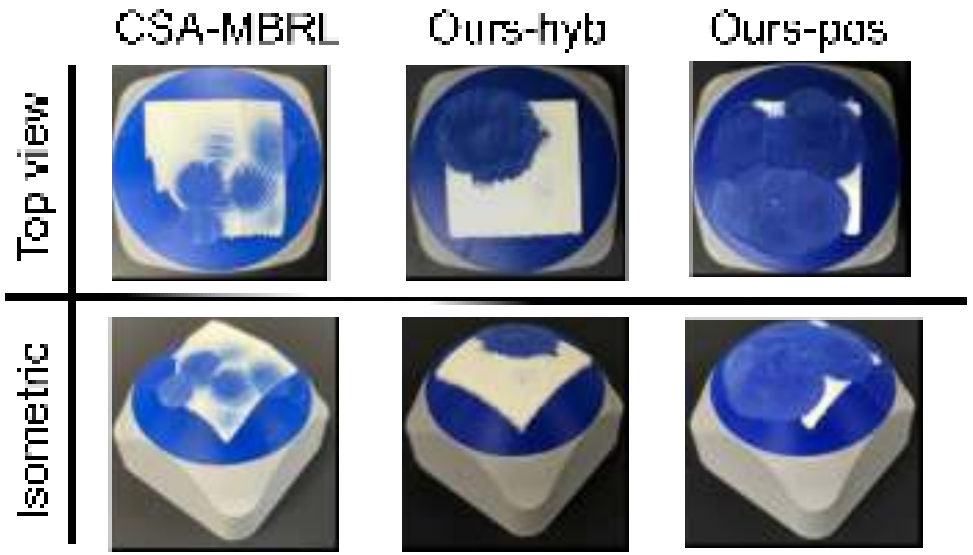




Application to Robotic Grinding [Hachimine et al. 2024]

Ours-h: Result of executing the proposed method with hybrid control of position and force.
Ours-p: Results of running the proposed method with position control





Application to Robotic Grinding [Hachimine et al. 2024]

Because diffuser can generate predictions for long-term action sequences at once, observation and plannning time is reduced compared to model predictive control (conventional optimal control scheme), which plans short-term action sequences.

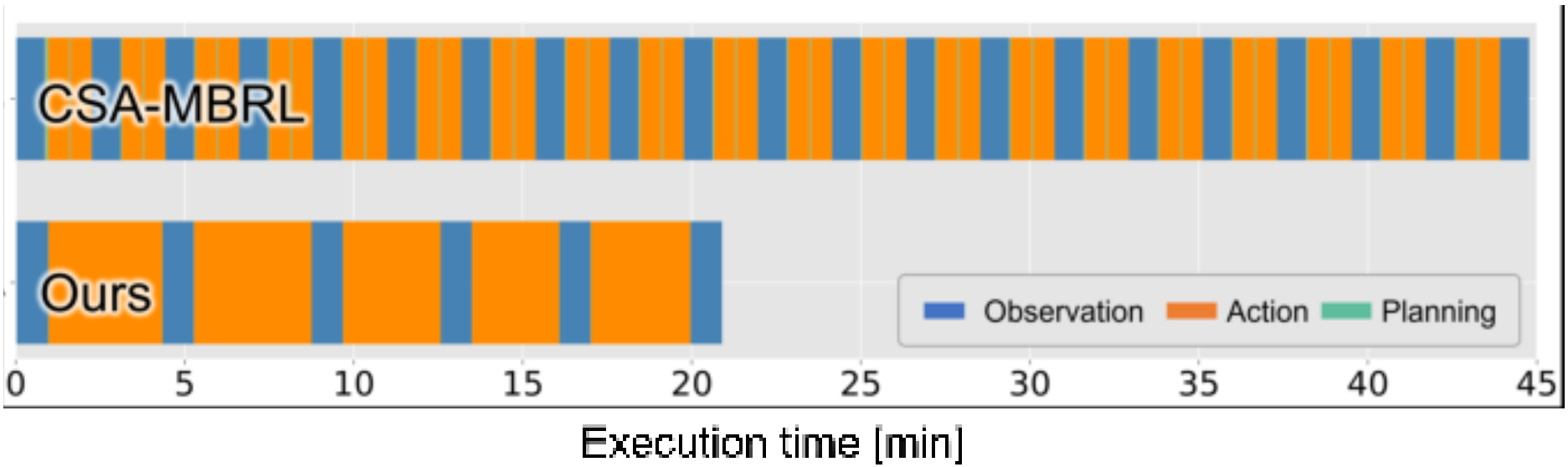


Image Inpainting [Lugmayr et al. 2022]

RePaint: Inpainting using Denoising Diffusion Probabilistic Models

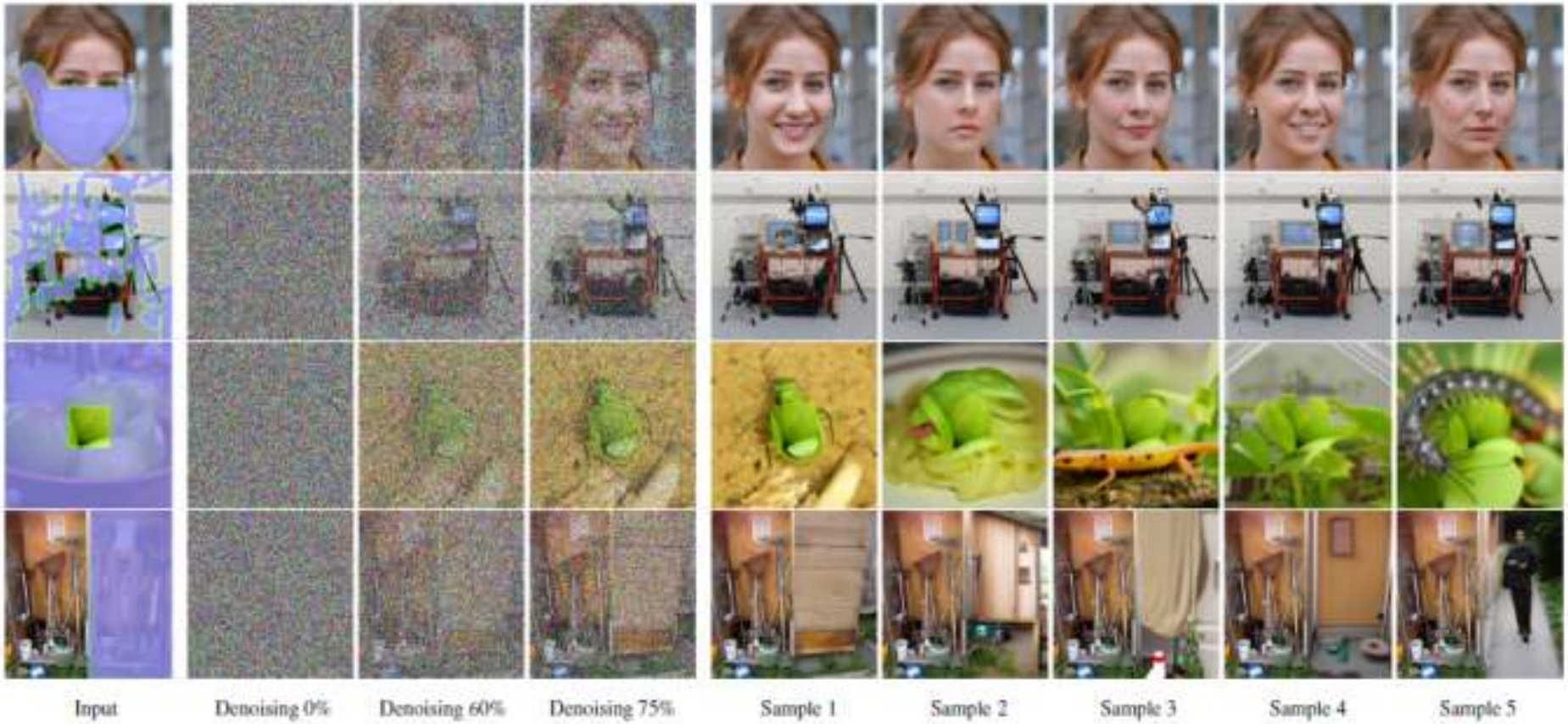
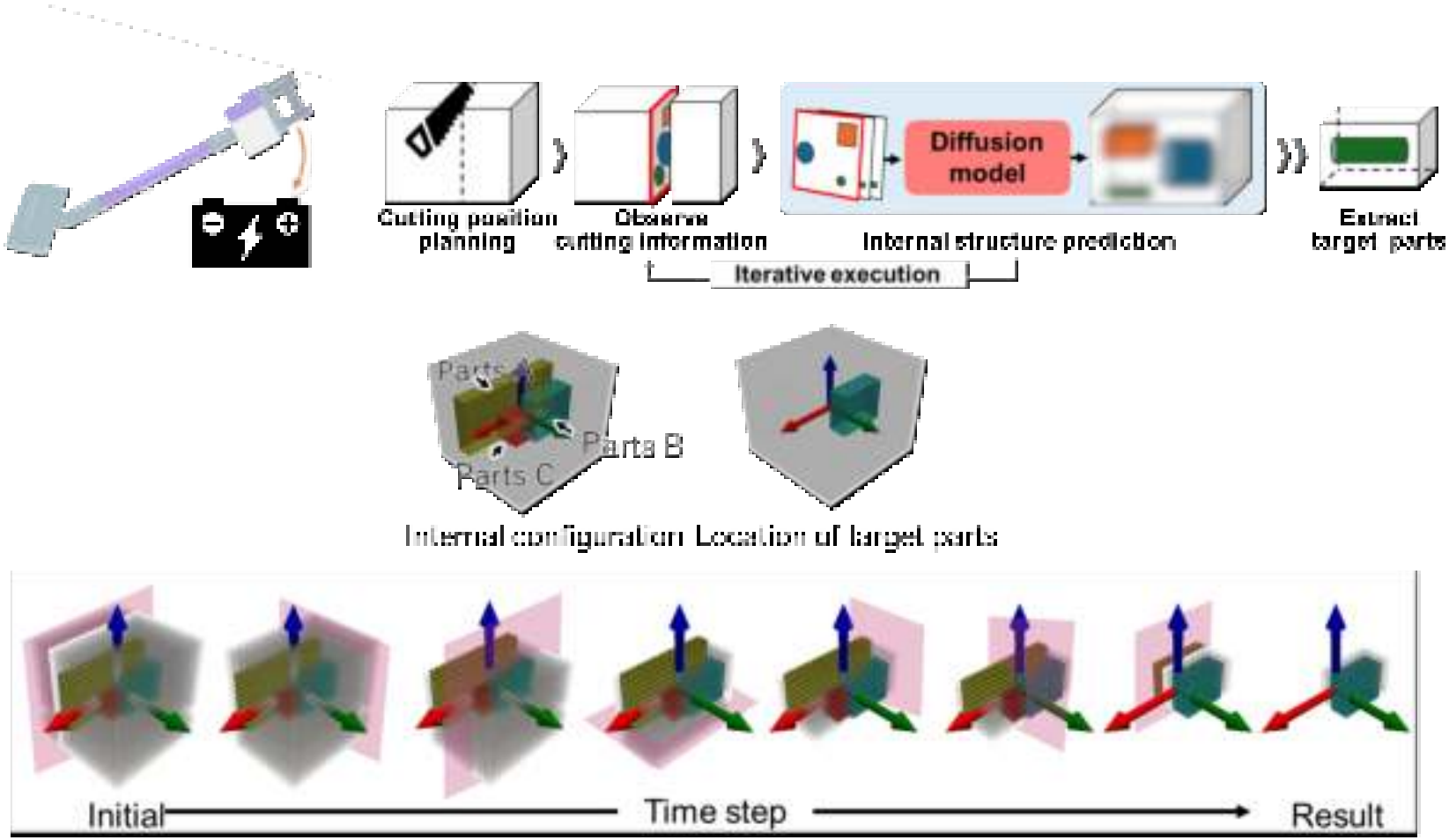


Figure 1. **We use Denoising Diffusion Probabilistic Models (DDPM) for inpainting.** The process is conditioned on the masked input (*left*). It starts from a random Gaussian noise sample that is iteratively denoised until it produces a high-quality output. Since this process is stochastic, we can sample multiple diverse outputs. The DDPM prior forces a harmonized image, is able to reproduce texture from other regions, and inpaint semantically meaningful content.

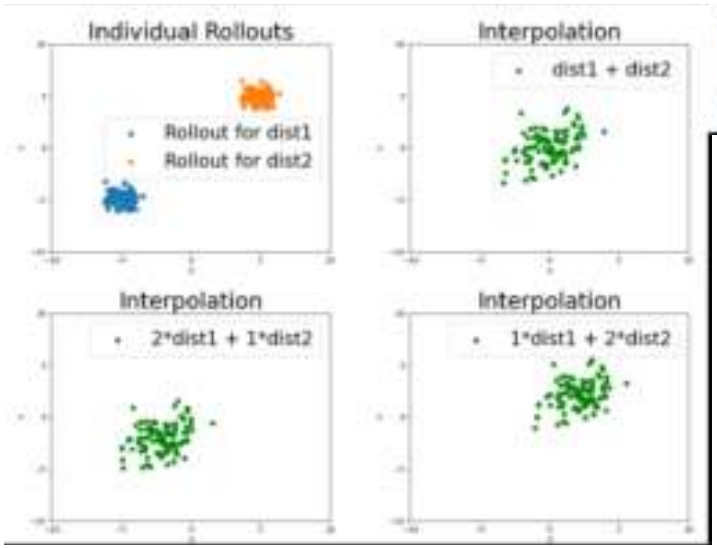
Planing for Extracting Internal Components [Hachimine et al. 2024]

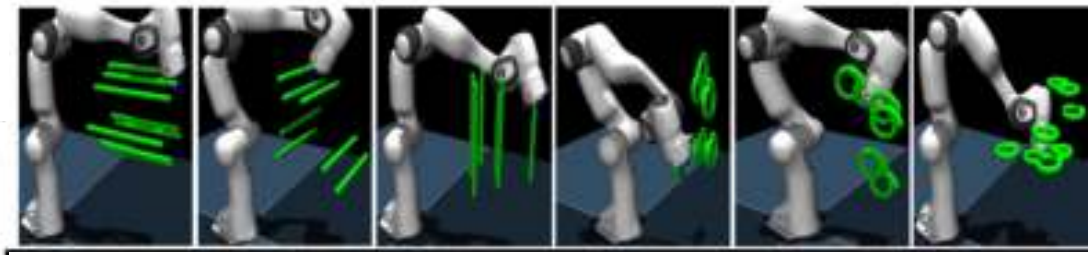
Extracting Internal Components by inpainting Internal Structures by Diffusion

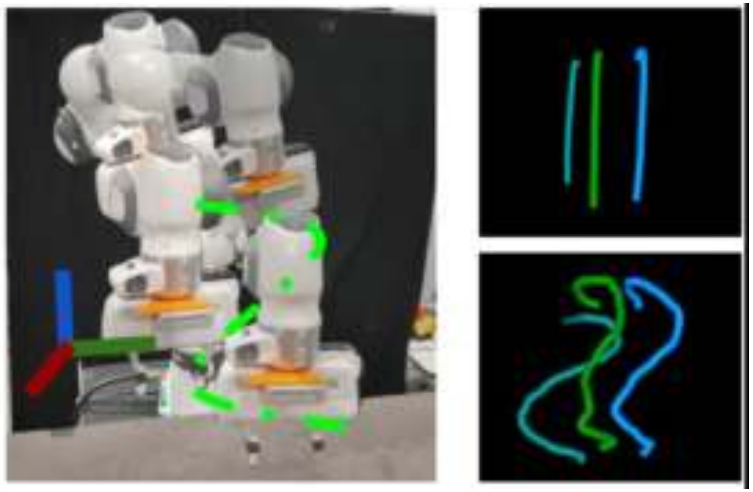


Action Policy Decomposition [Patil et al. 2024]

$$p(\tau)_{\text{composition}} \propto p_1(\tau)p_2(\tau)$$







About written exam on 2024 Dec. 27th

- You can bring anything, PC, smartphone, tablet, books etc
- All questions are basically from the lecture handouts
- Checking the exercises is highly recommended

References

- Yang Song, Stefano Ermon: Generative Modeling by Estimating Gradients of the Data Distribution. NeurIPS 2019: 11895-11907
- Jonathan Ho, Ajay Jain, Pieter Abbeel: Denoising Diffusion Probabilistic Models. NeurIPS 2020
- Michael Janner, Yilun Du, Joshua B. Tenenbaum, Sergey Levine: Planning with Diffusion for Flexible Behavior Synthesis. ICML 2022: 9902-9915
- Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, Shuran Song: Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, RSS2023
- Takumi Hachimine, Jun Morimoto, and Takamitsu Matsubara: Cutting Sequence Diffuser: Sim-to-Real Transferable Planning for Object Shaping by Grinding, RAL2024