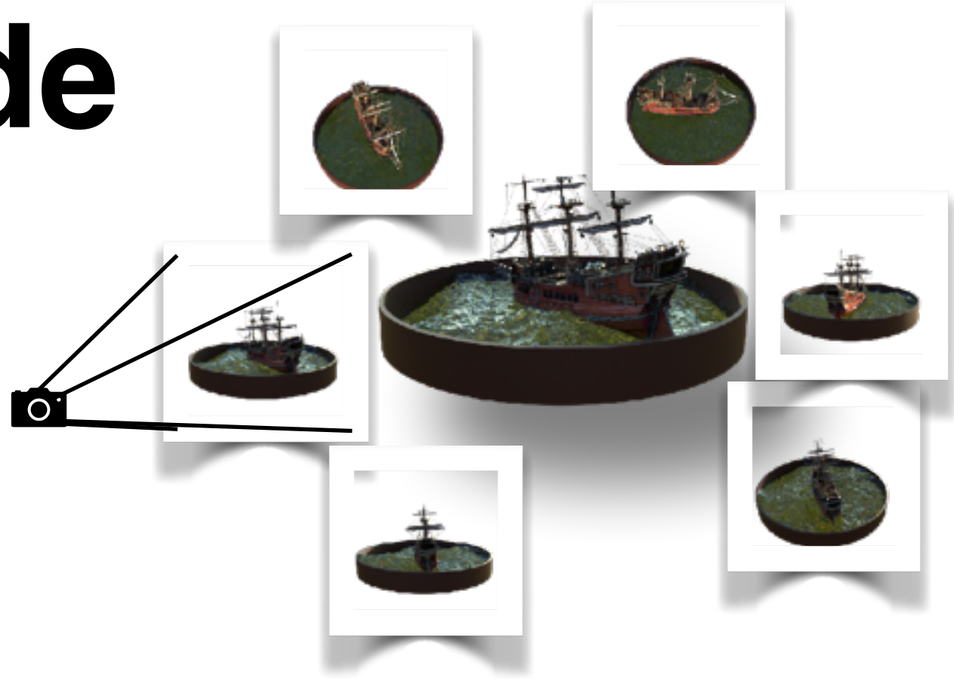
VIII: Generation of 3D scenes

3D CV

Kirill Struminsky

In the previous episode

- Novel view synthesis
  - Input: posed images of a scene
  - Goal: render the scene from novel viewpoints
- Approaches based on volumetric representations
  - Neural Radiance Fields
  - Gaussian Splatting
- Today: 3D asset generation & beyond
  - Same tool: volumetric representations
  - New task: no posed images



https://cat3d.github.io/

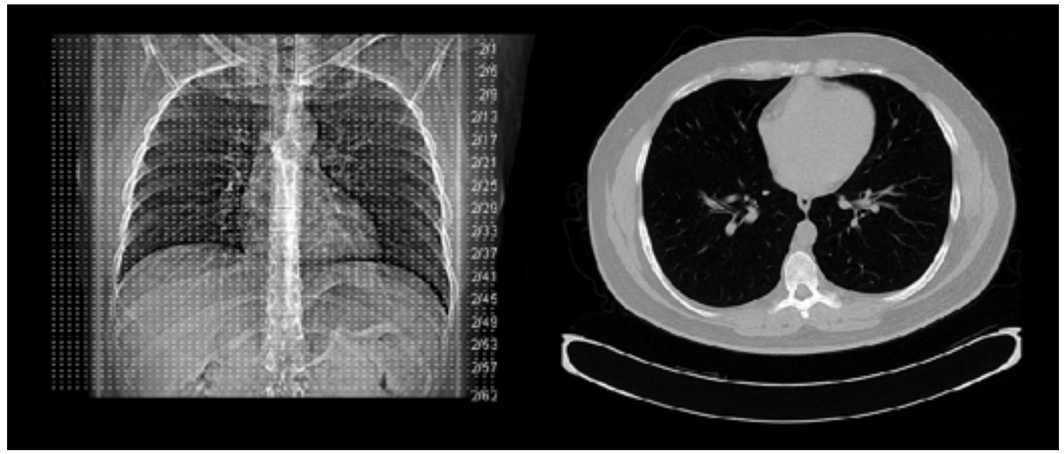
Motivation

- 3D assets are expensive
- Demand goes beyond game industry

Image taken from “The Next Leap: How A.I. will change the 3D industry” talk by Andrew Price

Motivation

- Inverse problems
  - Observe $x_i = g(\theta, z_i)$
  - Find θ
- Generative model $P(X)$
  - Find θ
  - Such that $x_i = g(\theta, z_i)$ are plausible samples
- Can we use $P(X)$ as an objective?



Setup

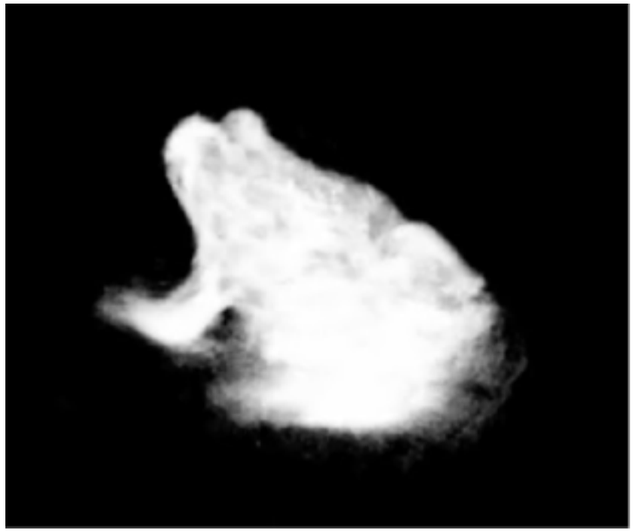
- Pre-trained text-to-2D diffusion model
- Input: text prompt

  A DSLR photo of a squirrel wearing a kimono reading a book
- Output: 3D model

Video taken from “DreamFusion: Text-to-3D using 2D Diffusion” project page

Parameterising & Rendering 3D Models

- Neural Radiance Fields
- Volumetric density $\sigma : \mathbb{R}^3 \rightarrow \mathbb{R}_{\geq 0}$
- Radiance field $c : \mathbb{R}^3 \rightarrow [0,1]^3$

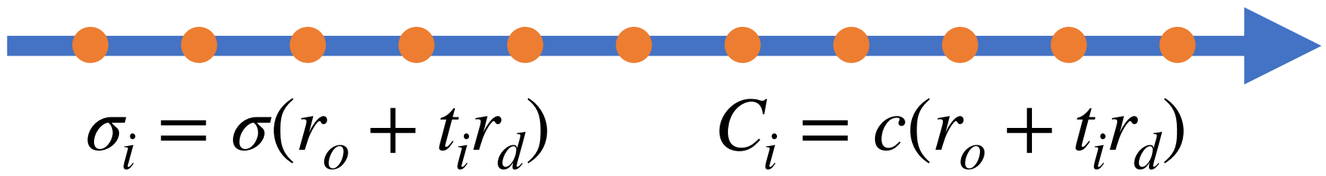


- Often, σ and c are neural networks

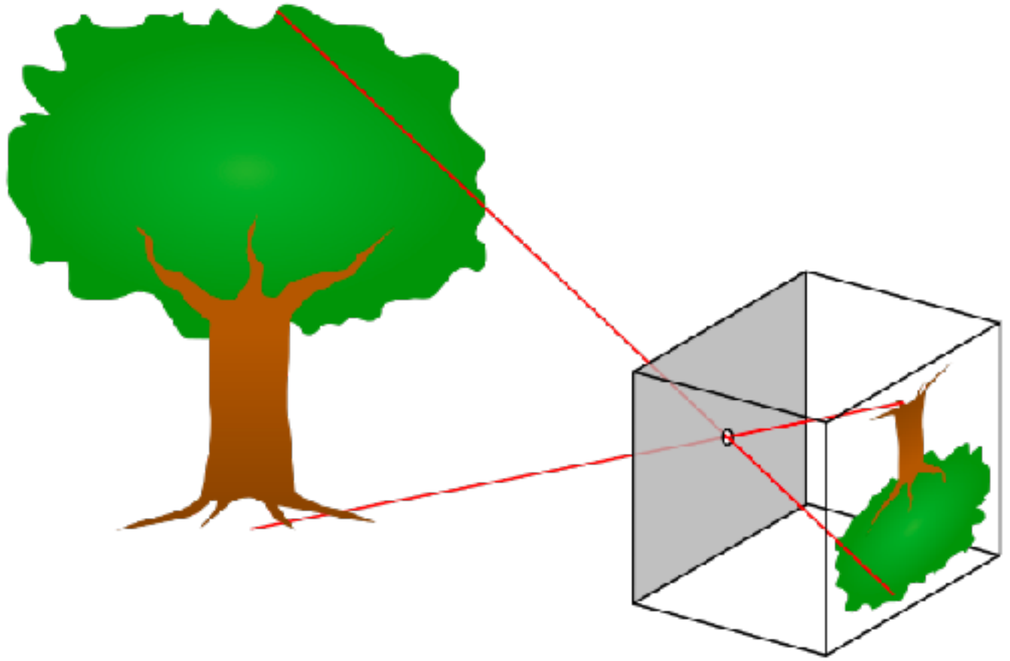
Parameterising & Rendering 3D Models

- Fields σ and c parameterise the scene
- Render image pixel by pixel

$r = (r_o, r_d)$ t_i

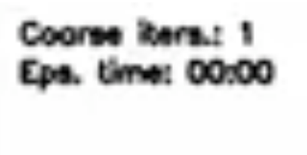


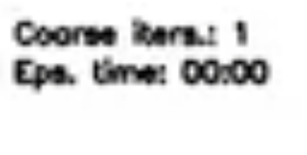
$\sigma_i = \sigma(r_o + t_i r_d)$ $C_i = c(r_o + t_i r_d)$



- Pixel color is

$$C = \sum_i (1 - \exp(-\sigma_i))\exp(-\sum_{j<i} \sigma_j) C_i$$

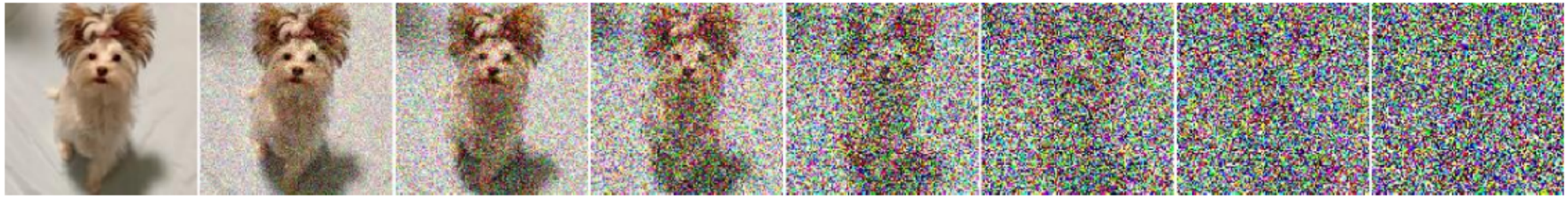




Diffusion Models Recap

Mapping data to noise

$$x_t = \alpha_t x_0 + \sigma_t \varepsilon$$



Trained reverse map

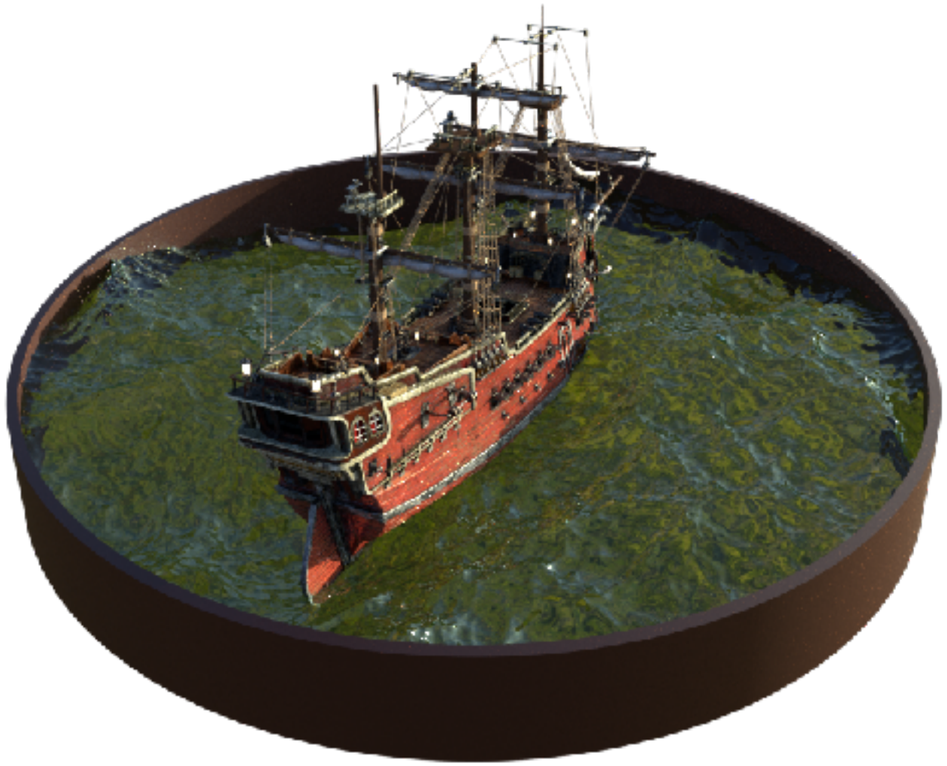
$$\hat{x}_\phi(x_t, t) = \frac{(x_t - \sigma_t \varepsilon_\phi(x_t, t))}{\alpha_t}$$

$$\nabla_{x_t} \log p_t(x_t) = s(x_t) \approx -\frac{\varepsilon_\phi(x_t, t)}{\sigma_t}$$

https://arxiv.org/abs/2011.13456

DreamFusion: Text-to-3D using 2D Diffusion

- Consider rendered image $x = g(\theta, z)$
  - Scene parameterisation θ
  - View angle, camera parameters, etc. z
- How do we find θ?
- Optimize $\mathscr{L}_{Diff}(\phi, x)$?
  - Training optimises ϕ
  - Can we optimise for x?
- Minimize $\mathbb{E}_z \mathscr{L}_{Diff}(\phi, g(\theta, z))$



Score Distillation Sampling

- Loss $\mathscr{L}_{Diff}(\phi, x_0) = \mathbb{E}_{t\sim U[0,1],\varepsilon\sim\mathcal{N}(0,I)}\left[w(t)\|\varepsilon_\phi(x_t, t) - \varepsilon\|_2^2\right]$
- Image $x_0 = g(\theta, z)$ and $x_t = \alpha_t x_0 + \sigma_t \varepsilon$
- Compute gradient

$$\nabla_\theta \mathscr{L}_{Diff} = 2\mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\phi(\alpha_t x + \sigma_t\varepsilon, t) - \varepsilon\right)\cancel{\frac{\partial \hat{\varepsilon}}{\partial x_t}}\frac{\partial x_t}{\partial x_0}\frac{\partial g(\theta, z)}{\partial \theta}\right]$$

- Omitting score Jacobian helps

Score Distillation Sampling Loss

- Let $\nabla_\theta \mathscr{L}_{SDS} := \mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\phi(x_t, t) - \varepsilon \right) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right]$
- Can we interpret $\mathscr{L}_{SDS}$?

$$\mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\phi(x_t, t) - \varepsilon \right) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right] = \mathbb{E}_t\left[w(t)\mathbb{E}_\varepsilon\left(\hat{\varepsilon}_\phi(x_t, t) - \varepsilon \right) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right] =$$

$$\mathbb{E}_t\left[w(t)\mathbb{E}_\varepsilon \hat{\varepsilon}_\phi(x_t, t) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right]$$

Score Distillation Sampling Loss

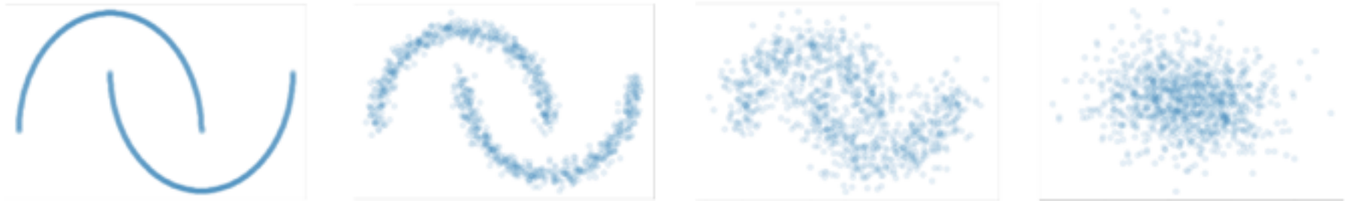
- Let $\nabla_\theta \mathscr{L}_{SDS} := \mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\phi(x_t, t) - \varepsilon \right) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right]$
- Can we interpret $\mathscr{L}_{SDS}$?
- Recall $\hat{\varepsilon}_\phi(x_t, t) \approx -\sigma_t \nabla_{x_t} \log p_t(x_t)$ and rewrite

$$\mathbb{E}_\varepsilon \hat{\varepsilon}_\phi(x_t, t) = \mathbb{E}_{p_t(x_t|x_0)} \hat{\varepsilon}_\phi(x_t, t) \approx -\mathbb{E}_{p_t(x_t|x_0)} \sigma_t \nabla_{x_t} \log p_t(x_t)$$

- Finally

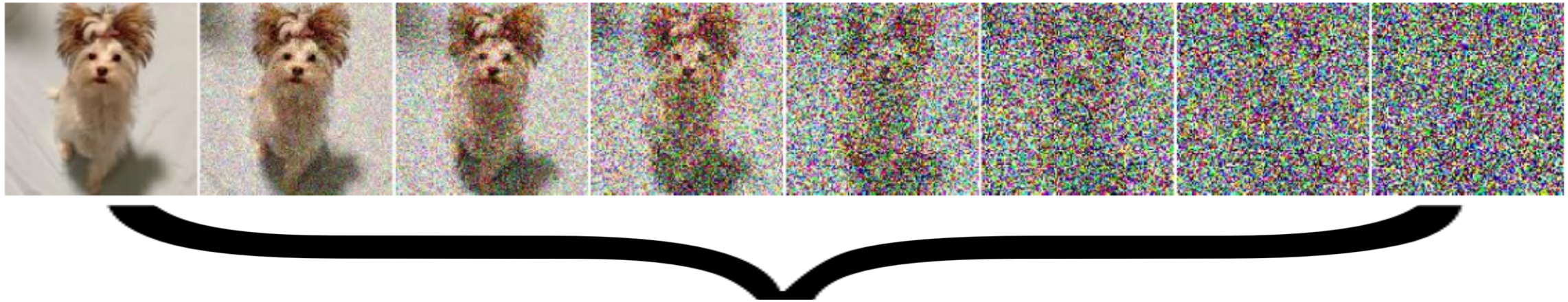
$$\mathbb{E}_t\left[w(t) \mathbb{E}_\varepsilon \hat{\varepsilon}_\phi(x_t, t) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right] \approx -\mathbb{E}_t\left[w(t)\sigma_t \mathbb{E}_{p_t(x_t|x_0)} \nabla_{x_t} \log p_t(x_t) \frac{\partial x_t}{\partial x_0} \frac{\partial g}{\partial \theta} \right] = -\nabla_\theta \mathbb{E}_t w(t) \sigma_t \mathbb{E}_{p_t(x_t|x_0)} \log p_t(x_t)$$

Score Distillation Sampling Loss



- In sum,
  - $\mathscr{L}_{Diff} \approx -\log p_0(x_0)$
  - $\mathscr{L}_{SDS} \approx -\mathbb{E}_t w'(t) \mathbb{E}_{p_t(x_t|x_0)} \log p_t(x_t)$

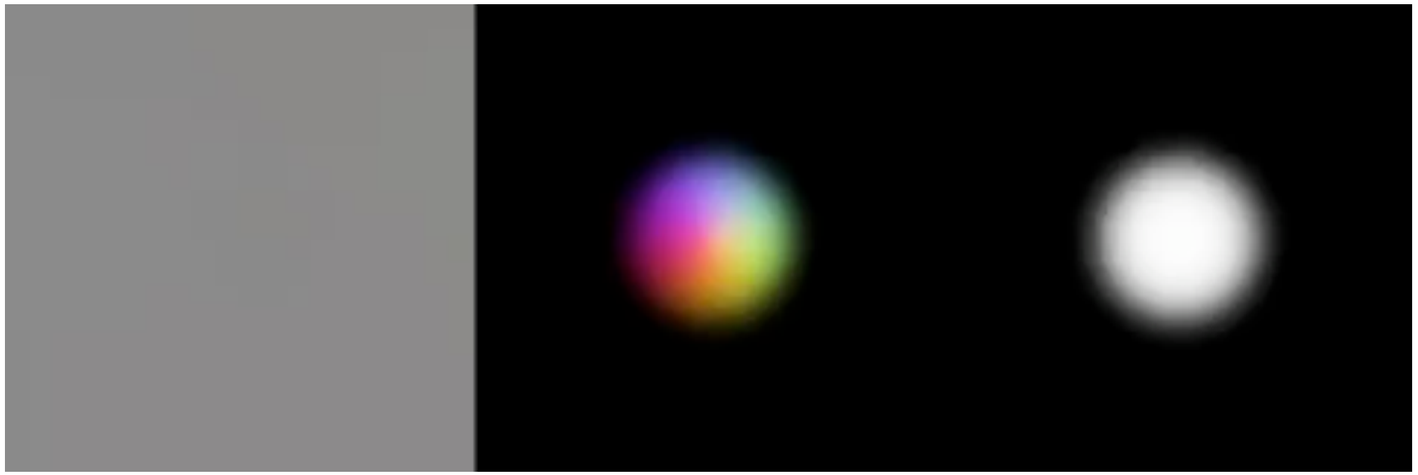
$\mathscr{L}_{Diff}$



$\mathscr{L}_{SDS}$

Training Dreamfusion

- Choose prompt y_t (omitted for clarity)
- Diffusion model $\hat{\varepsilon}_\phi(x_t, t)$ (Imagen)
- Training loop:
  - Sample camera position z
  - Render image $x_0 = g(\theta, z)$
  - Sample t and x_t
  - Update θ with



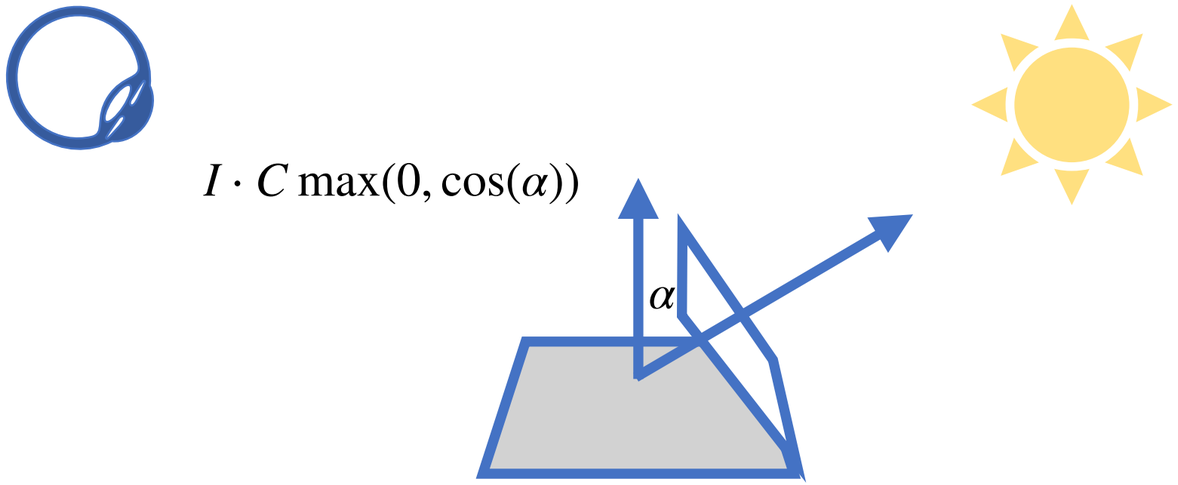
$$\nabla_\theta \mathscr{L}_{SDS} := \mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\phi(x_t, t) - \varepsilon\right) \frac{\partial x_t}{\partial x_0} \frac{\partial g(\theta, z)}{\partial \theta} \right]$$

Additional Heuristics (I)

- Janus problem
- Remedy: specify camera angle in prompts

Additional Heuristics (II)

- Poor geometry
- Remedy (1/2): simulate shadows



- Remedy (2/2): process colourless images

Problems to Fix

- Poor model quality
- Seemingly ad-hoc objective
- Low sample diversity
- Janus problem

Adapting to Latent Diffusion

- Imagen is not public
- Naive adaptation works fine
  - Render image x
  - Encode $z_0 = E(x)$
  - Compute SDS loss in latent space

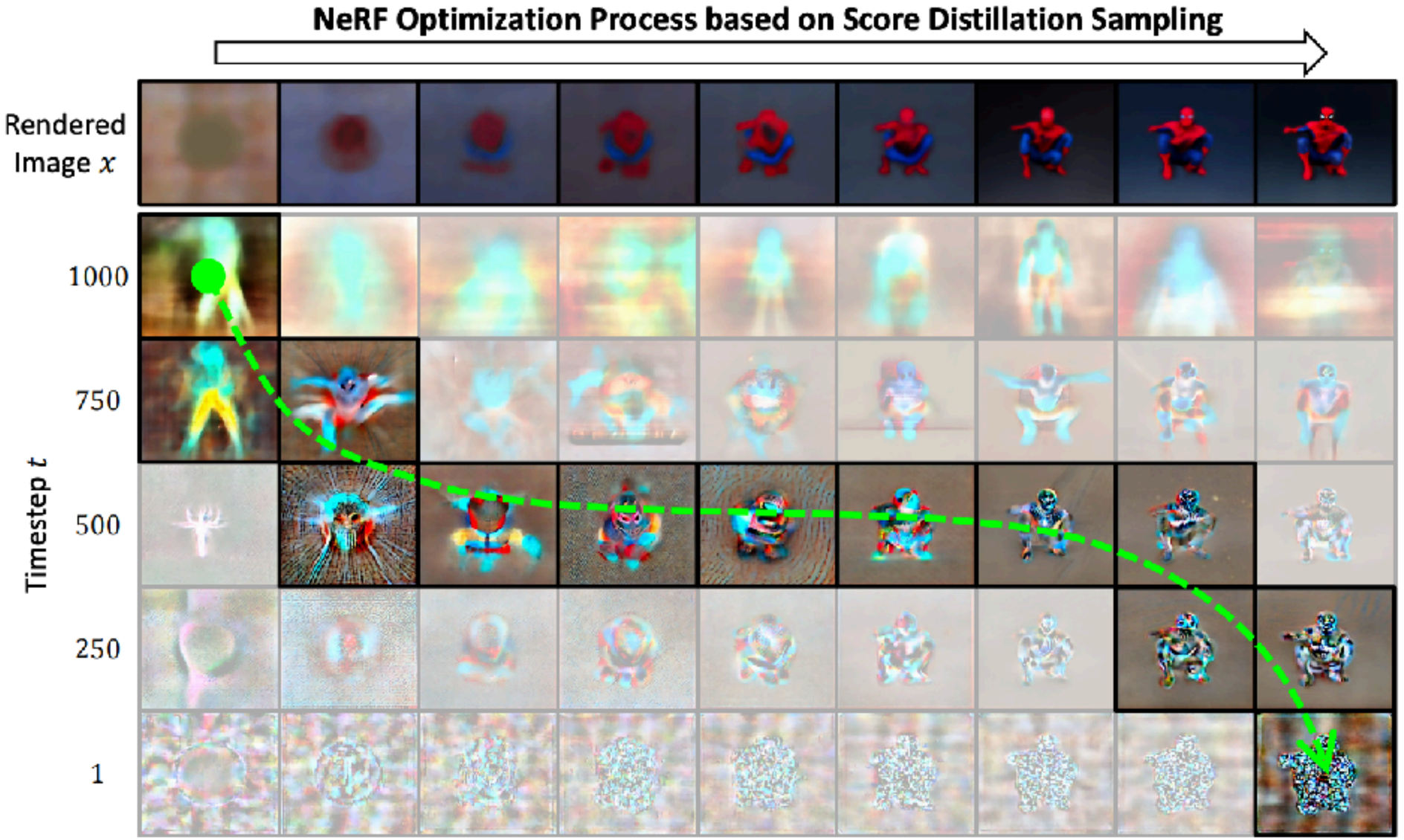
a blue poison-dart frog sitting on a water lily

neuschwanstein castle, aerial view

(1) 64 rendering (2) + 512 rendering (3) + Annealed t (4) + VSD

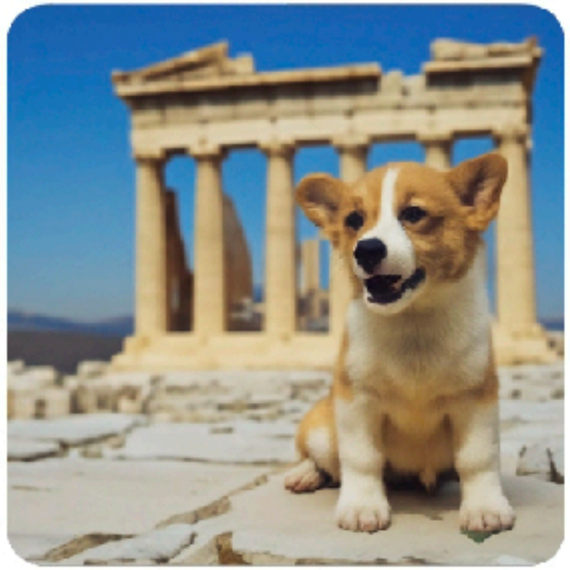
https://research.nvidia.com/labs/dir/magic3d/
https://ml.cs.tsinghua.edu.cn/prolificdreamer/

Time Annealing



Tackling Low Sample Diversity

Input images



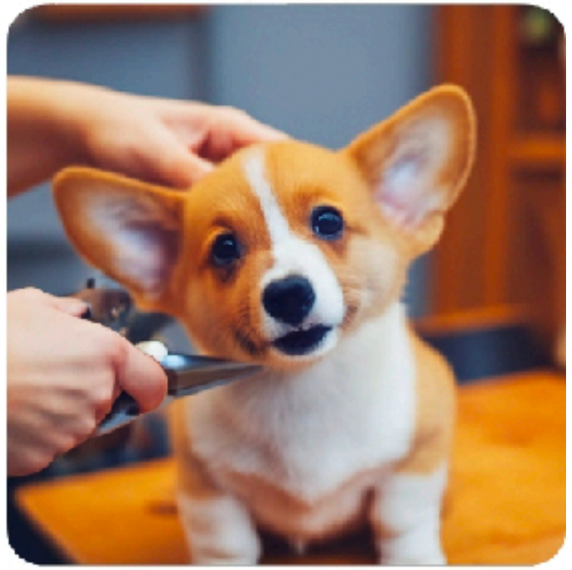
in the Acropolis

swimming

sleeping

in a doghouse

in a bucket

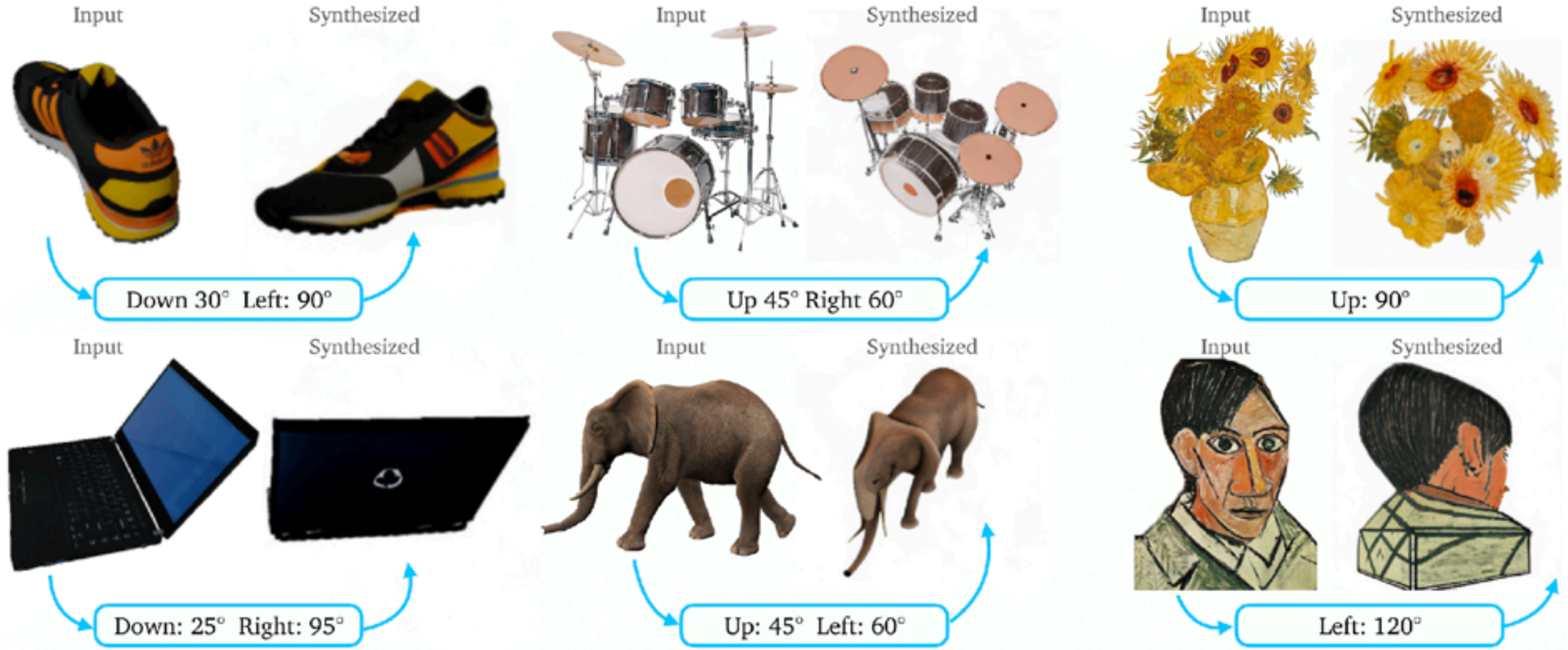


getting a haircut

Dreambooth 3D

https://dreambooth3d.github.io/

Zero-123: viewangle control



https://zero123.cs.columbia.edu/

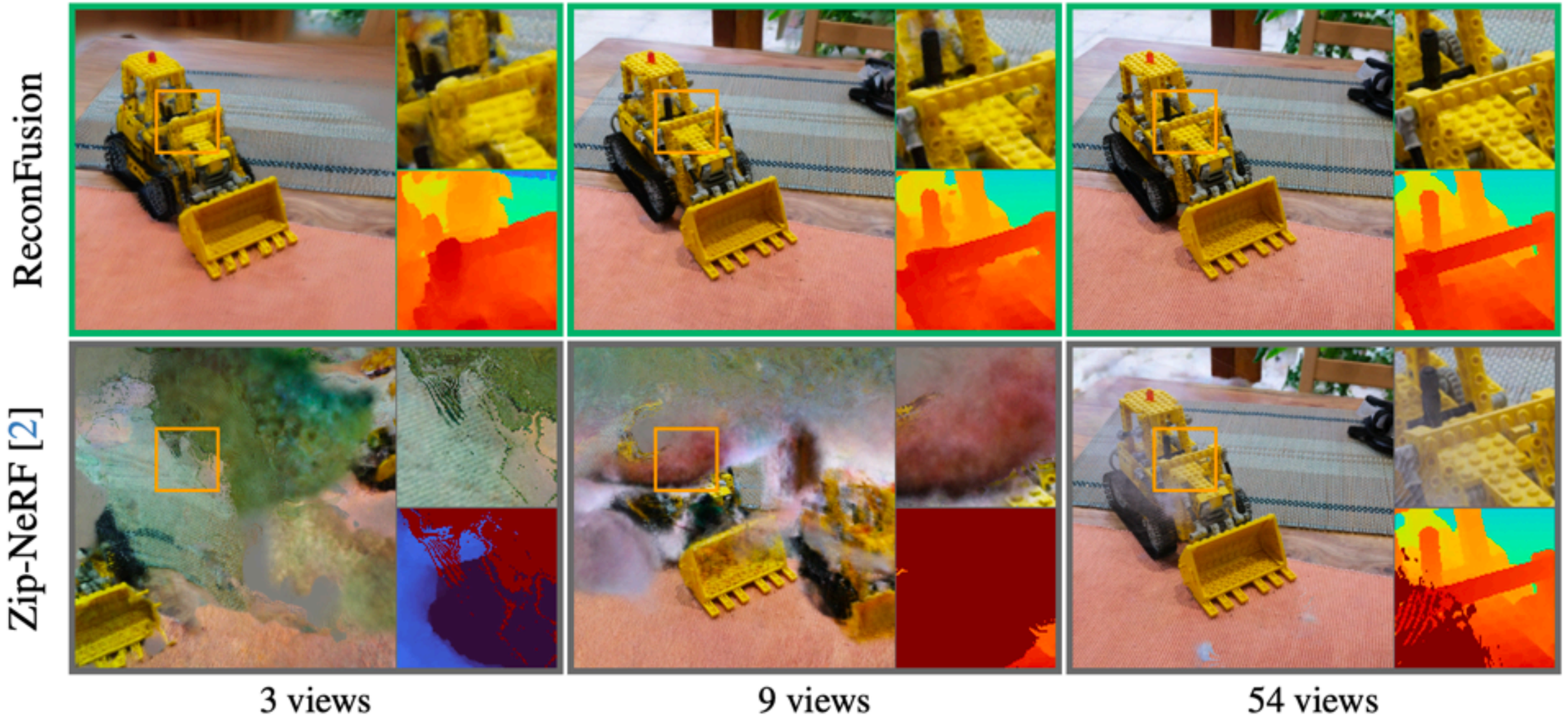
Zero-123: examples

https://zero123.cs.columbia.edu/

Magic123

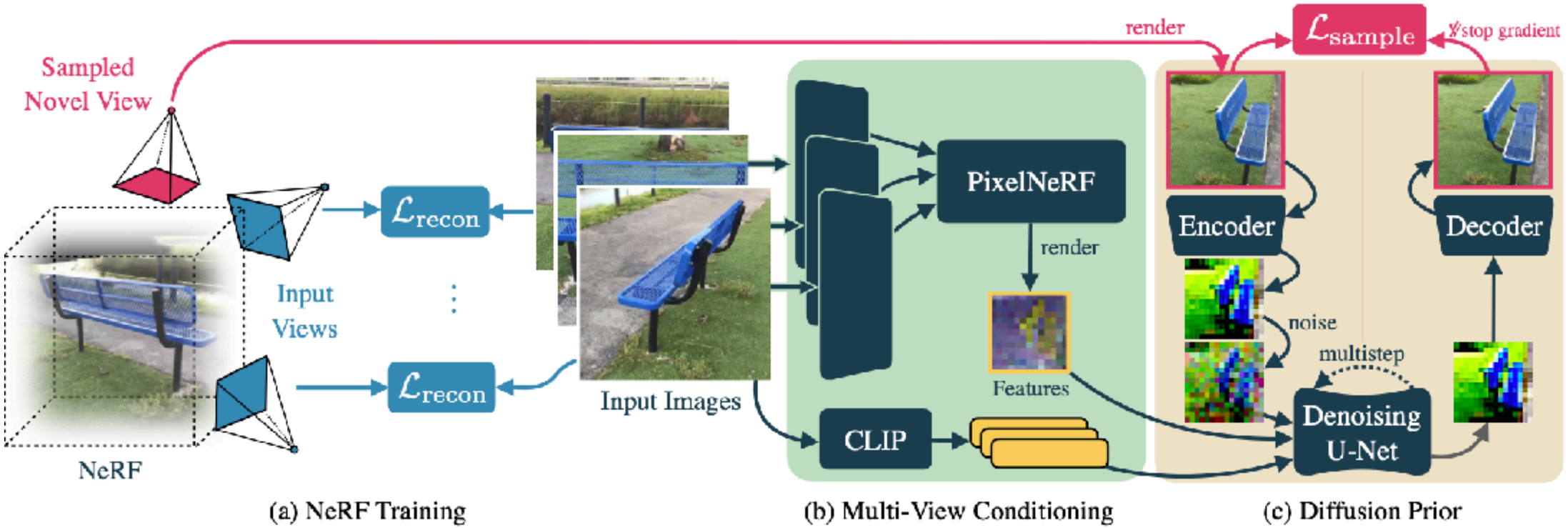
https://guochengqian.github.io/project/magic123/

Reconfusion



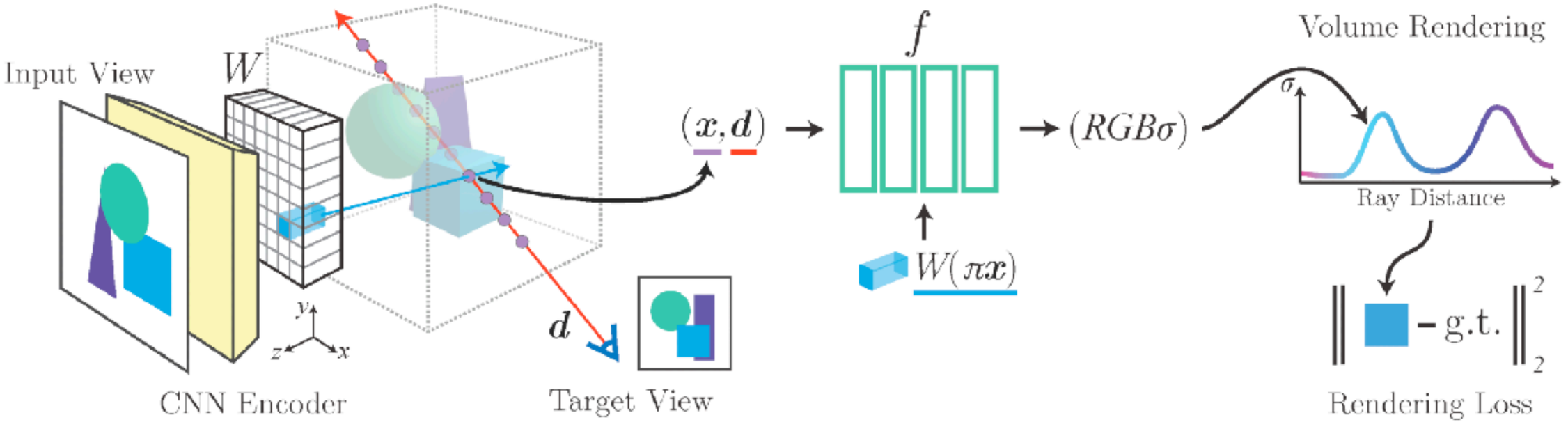
https://reconfusion.github.io/

Overall Scheme



Pixel-NeRF for View Conditioning

$$p(x \mid x_{obs}, \pi_{obs}, \pi)$$



https://arxiv.org/abs/2012.02190

Pixel-NeRF Training

$$\mathcal{L}_{\text{Diff}}(\theta, \phi) = \mathbb{E}_{x,\pi,x^{\text{obs}},\pi^{\text{obs}},\epsilon,t} \left\| \epsilon - \epsilon_\theta(z_t, t, e^{\text{obs}}, f) \right\|^2, \quad (2)$$

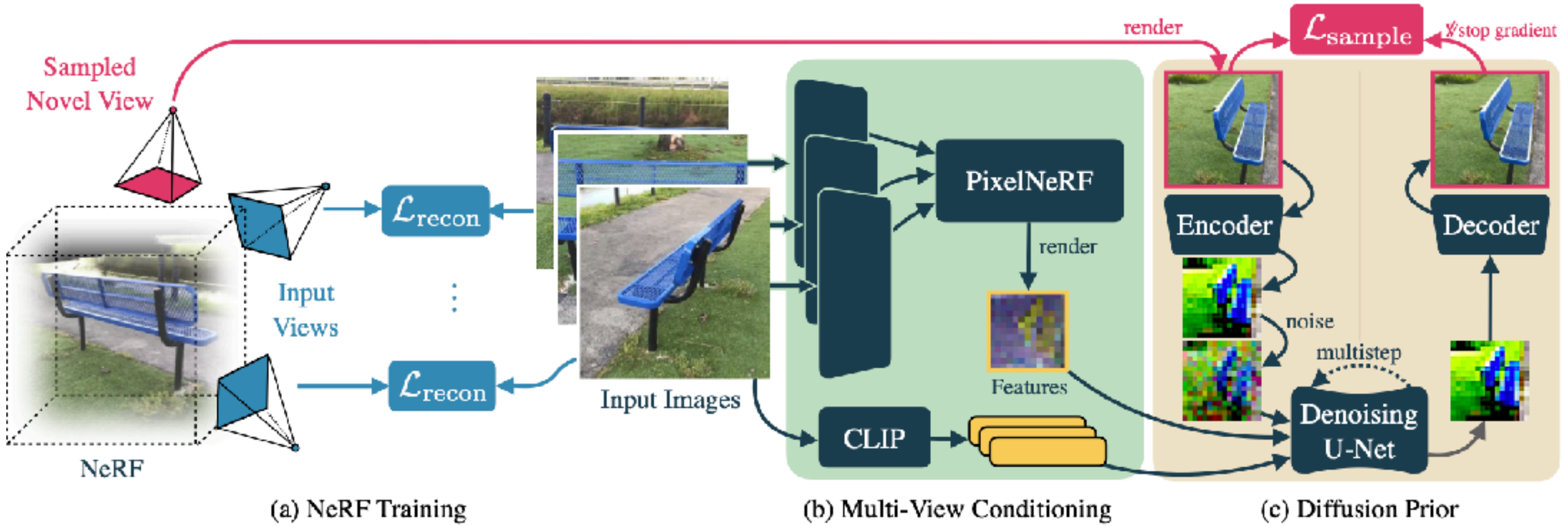
where $t \in \{1, \ldots, T\}$ is the diffusion timestep, $\epsilon \sim \mathcal{N}(0, I)$, $z_t = \alpha_t \mathcal{E}(x) + \sigma_t \epsilon$ is the noisy latent at that timestep, e^{obs} are the CLIP image embeddings for the input images x^{obs}, and f is the rendered feature map from Pixel-NeRF R_ϕ.

$$\mathcal{L}_{\text{PixelNeRF}}(\phi) = \mathbb{E}_{x^{\text{obs}},\pi^{\text{obs}},x,\pi} \| c - x_\downarrow \|^2, \quad (3)$$

where c is an output of the PixelNeRF model (at the same resolution as the feature map f) and $x_\downarrow$ is the target image downsampled to the spatial resolution of z_t and f.

Pixel-NeRF for View Conditioning

Regularising NeRFs with Diffusion



$$\mathcal{L}_{\text{Recon}}(\psi) = \mathbb{E}_{x^{\text{obs}},\pi^{\text{obs}}}\left[\ell(x(\psi,\pi^{\text{obs}}),x^{\text{obs}})\right] \qquad \mathcal{L}_{\text{sample}}(\psi) = \mathbb{E}_{\pi,t}\left[w(t)\left(\|x-\hat{x}_\pi\|_1 + \mathcal{L}_{\text{p}}(x,\hat{x}_\pi)\right)\right]$$


Zip-NeRF

SDS

Multistep

Annealed Multistep

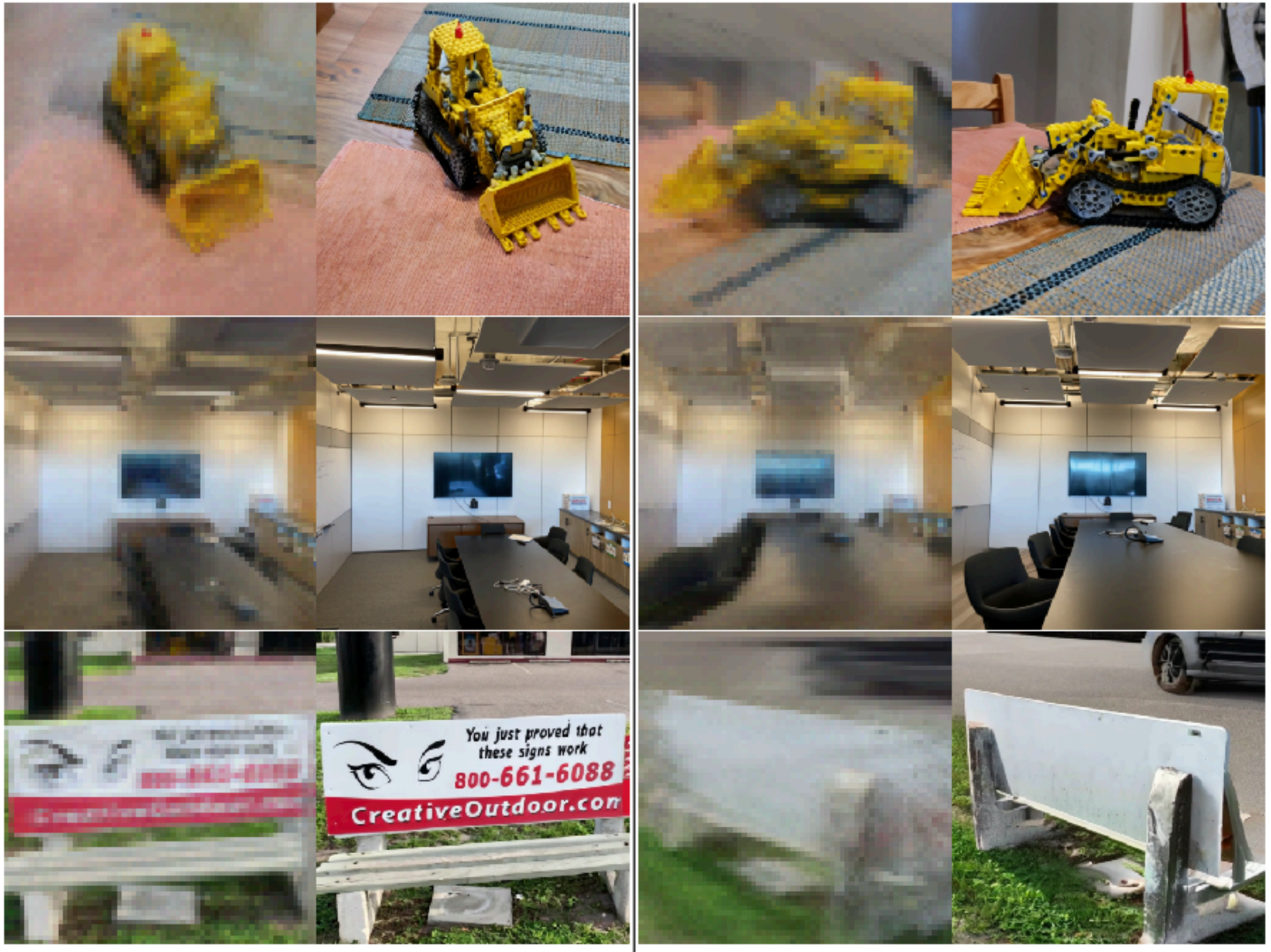
pose
w/o PT
Ours
Ground Truth
Sample 1
NeRF render 1
Sample 2
NeRF render 2

(a) 3 input views

(b) 6 input views

(c) 9 input views

(d) Ground truth



(a) PixelNeRF RGB output (b) Diffusion model sample (c) PixelNeRF RGB output (d) Diffusion model sample

Key takeaways

- Text-to-2D models
  - can solve to text-to-3D
  - can regularise NeRF
- Generation with SDS loss
- Model fine-tuning can address some of SDS issues