Reconstruction & Generation in 3D

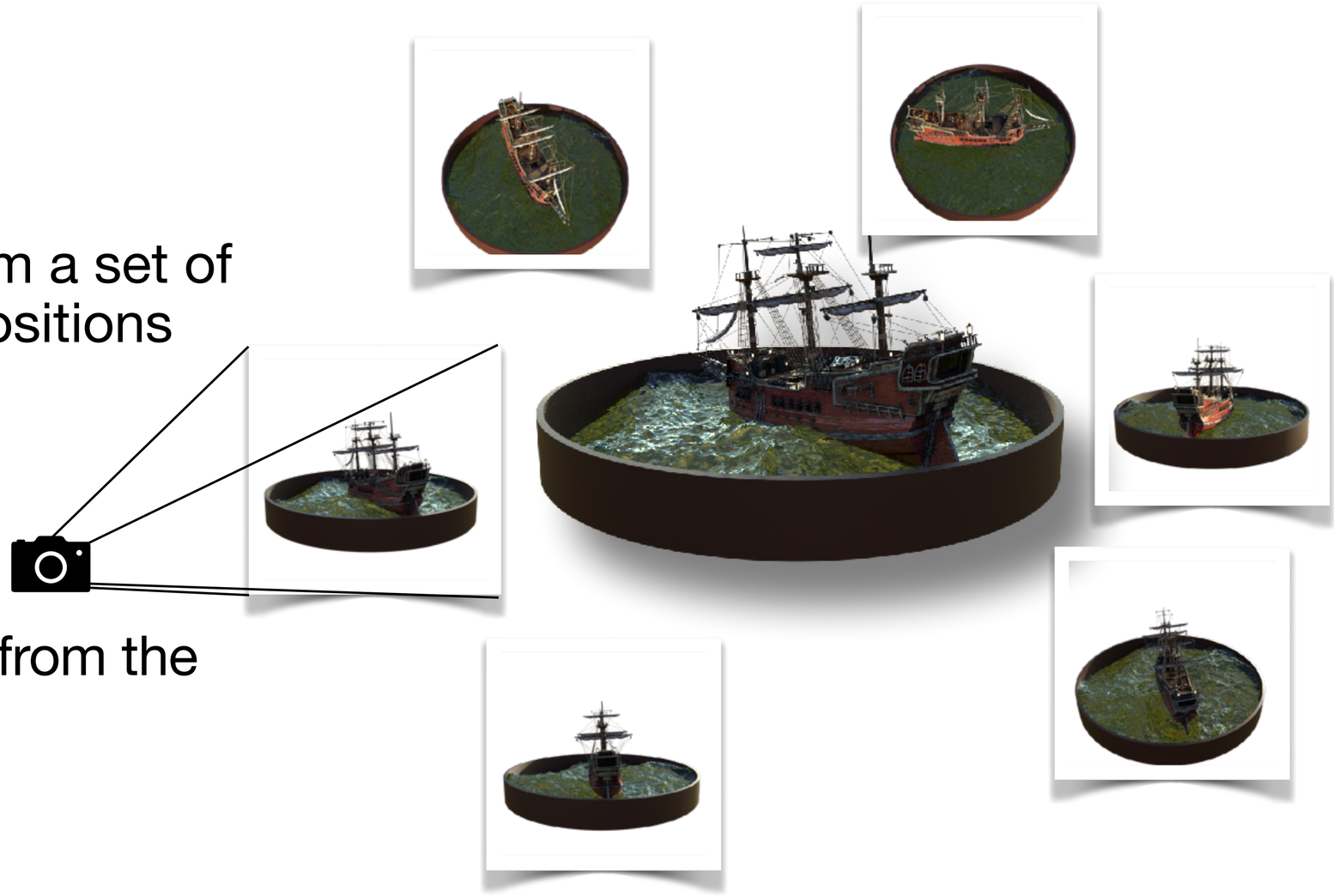
Aliev Mishan

Based on https://github.com/struminsky/hse_3dcv

Novel View Synthesis

Novel View Synthesis

- Problem setup:
  - A set of pictures taken from a set of pre-determined camera positions
  - Test camera position
- Goal:
  - A picture of a scene taken from the test camera position

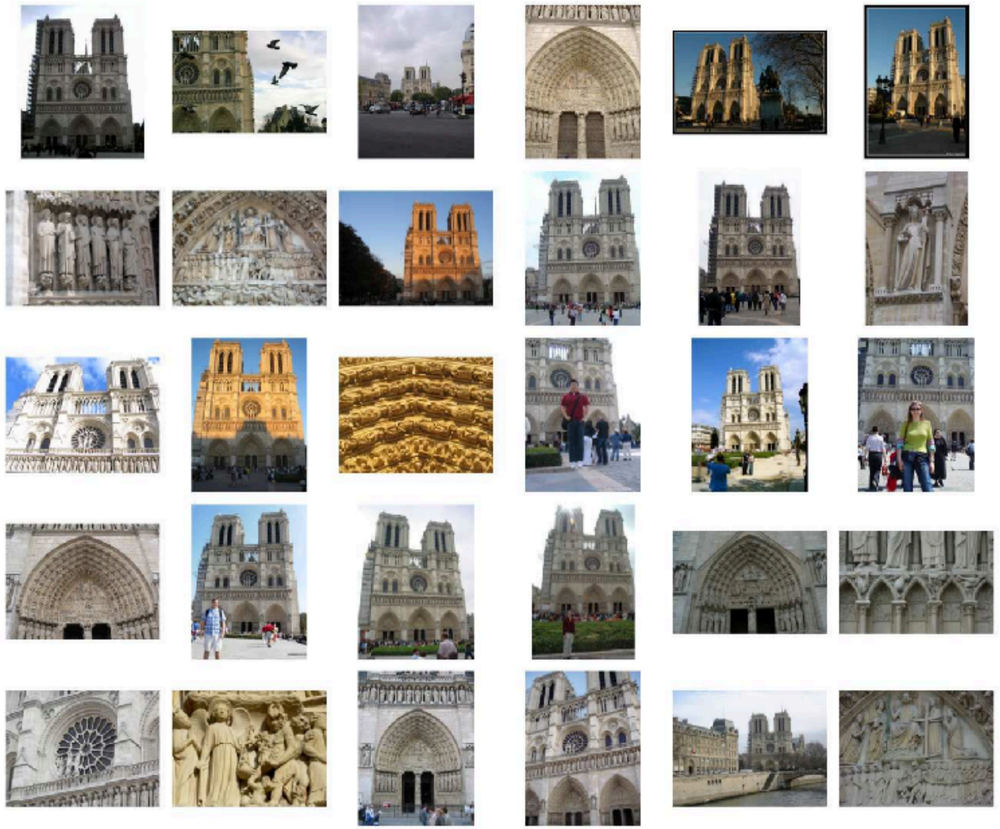


Mildenhall B. et al. Nerf: Representing scenes as neural radiance fields for view synthesis //Communications of the ACM. – 2021. – Т. 65. – №. 1. – С. 99-106.

Where Does the Data Come From?

- Structure From Motion (COLMAP)
  - Takes unposed images as input
- ARKit
  - Records device trajectory along with images

Side Note: Structure from Motion



Snavely, Noah, Steven M. Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D." ACM SIGGRAPH 2006

Neural Radiance Fields (NeRF)

Representing a Scene with Neural Fields

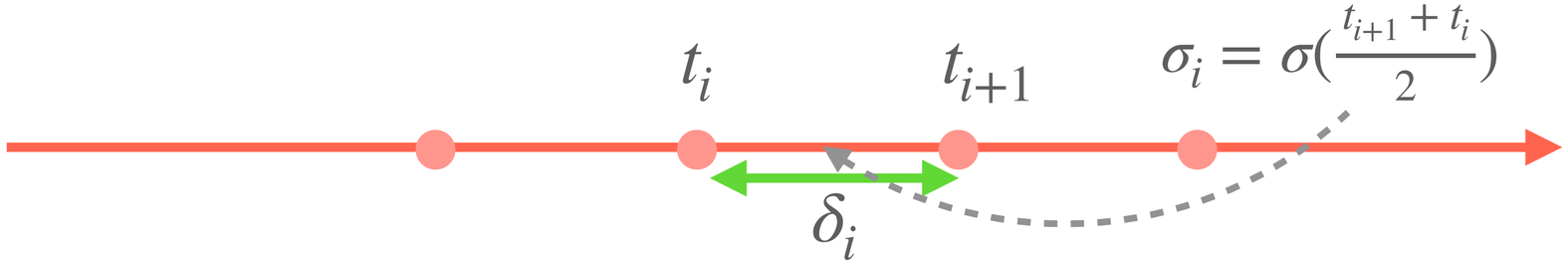
- Represent a scene with two fields
  - Density: $\sigma(x) : \mathbb{R}^3 \rightarrow \mathbb{R}^+$
  - Radiance: $C(x, d) : \mathbb{R}^3 \times S^2 \rightarrow \mathbb{R}^3$
- Density represents is related to opacity
- Radiance represents color of a point
- Architecture: MLP + positional embeddings

Mildenhall B. et al. Nerf: Representing scenes as neural radiance fields for view synthesis //Communications of the ACM. – 2021. – Т. 65. – №. 1. – С. 99-106.

Computing Pixel Color

- Density $\sigma(t) \in [0, +\infty)$ is related to opacity at t
- Divide the ray with points $t_1, \ldots, t_n$ and define

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



- Expected color along a ray is given by

$$C = \sum_i c(t_i) \cdot \alpha_i \prod_{j<i} (1 - \alpha_j)$$

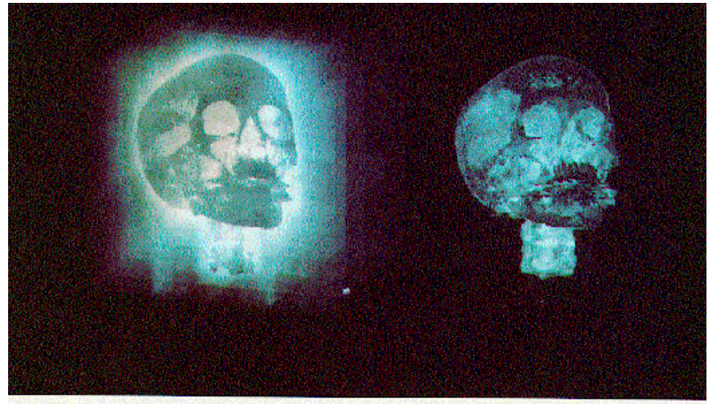


Fig. 11. Costs of rendering Figure 8 using hierarchical enumeration and adaptive termination.

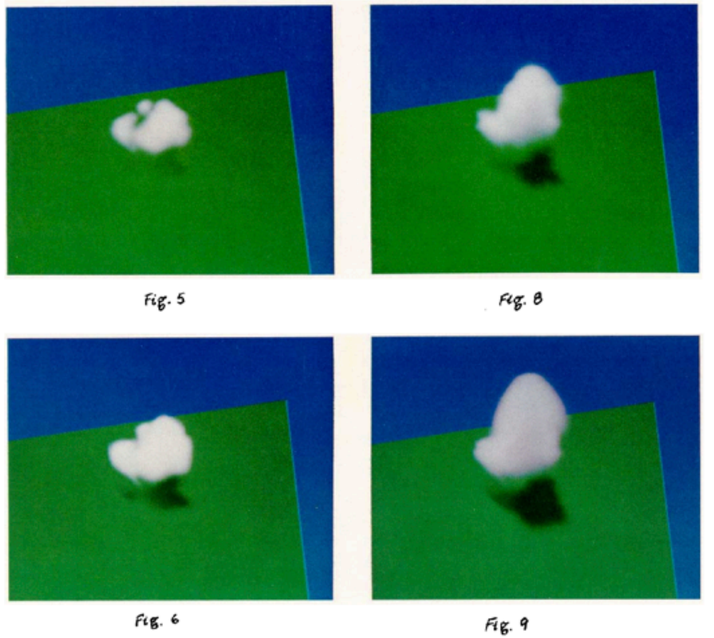


Fig. 5 Fig. 8

Fig. 6 Fig. 9

Neural Radiance Fields

Optimisation

- Training: minimise the mean squared error $\mathbb{E}_D \| C - C_{gt} \|^2$

$$C = \sum_i c_i \, \alpha_i \prod_{j<i} (1 - \alpha_j)$$

  - Optimise w.r.t. parameters of radiance C and density σ

Training Visualisation

- Synthetic data

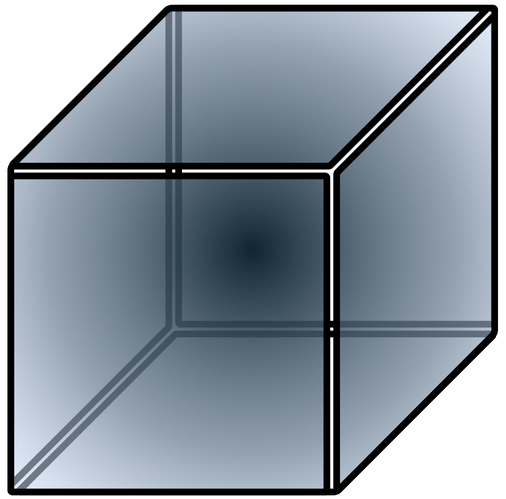
- Real forward-facing

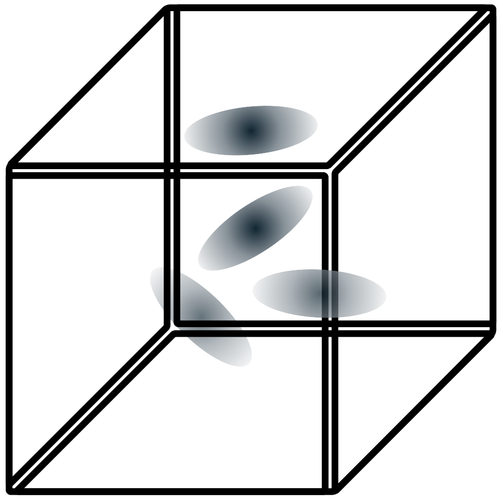
https://github.com/sunset1995/DirectVoxGO

Gaussian Splatting

Idea: Volumetric Rendering + Rasterisation

- Neural Radiance Fields uses **a dense** field parameterisation
- Rendering algorithm runs independently for each pixel
- Gaussian splatting uses **a sparse** parameterisation
  - Similar to meshes, scene consists of primitive shapes
  - Unlike meshes, these primitives have volume
  - Volumetric rendering algorithm based on **rasterization**

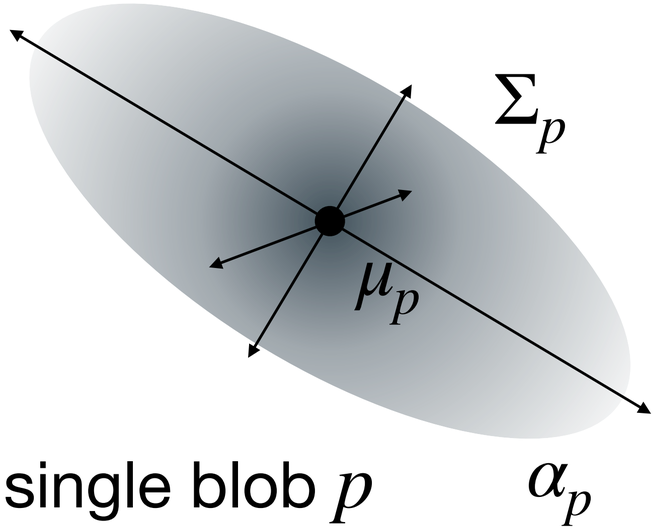




Final Rendering

3D Gaussian Visualization

Representing Scene with Gaussians



- We will fill the volume with multiple tiny blobs
- In NeRFs we relied on a density field $\sigma(x)$
- Here we will define opacity $\alpha(x) = 1 - \exp(\sigma(x)\Delta x)$ of a single blob p

$$\alpha(x) = \alpha_p \exp\left(-\frac{1}{2}(x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p)\right)$$

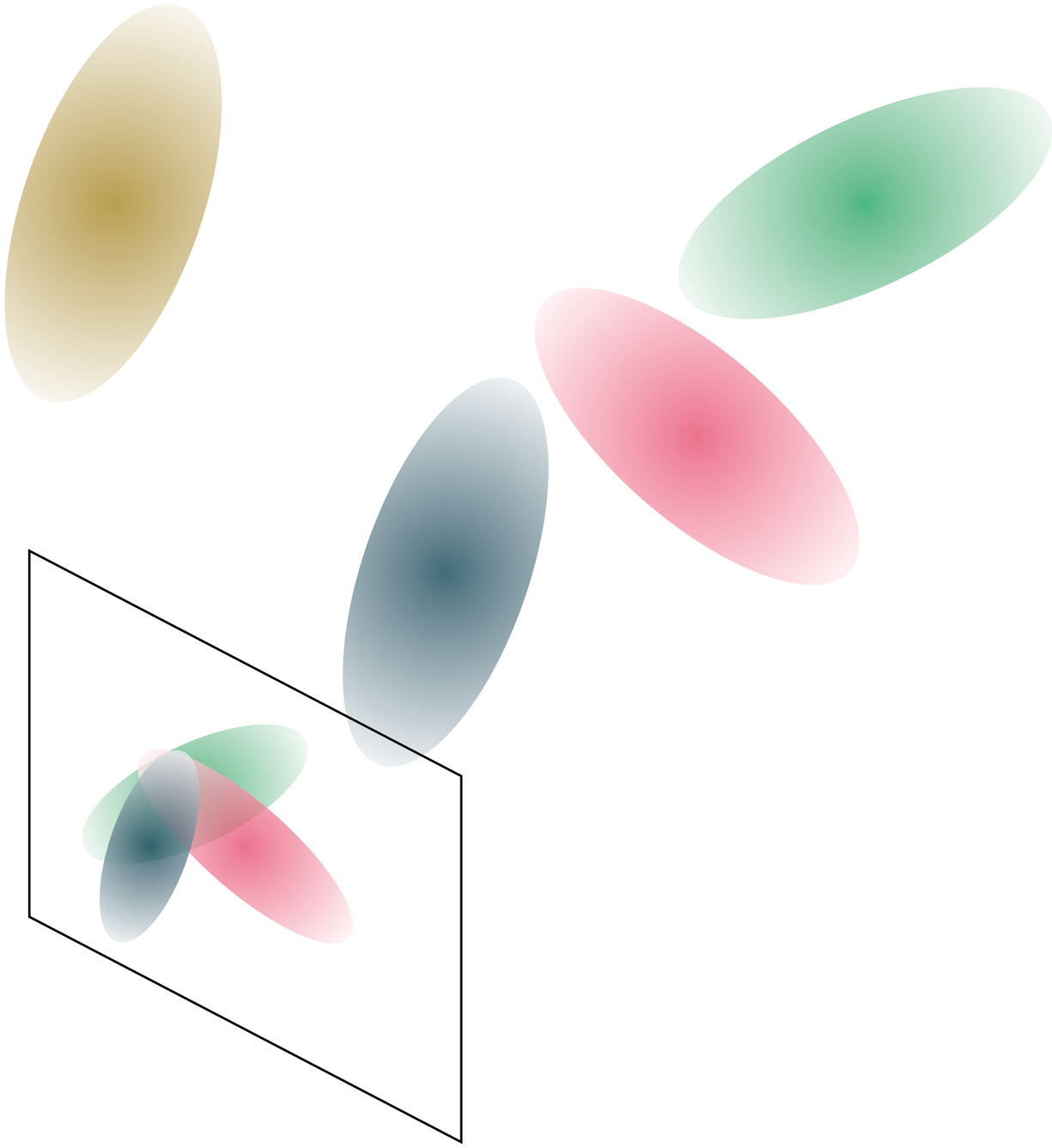
- Parameters $\alpha_p, \mu_p, \Sigma_p$ define opacity, position, and shape of the blob respectively

Rendering Algorithm

Rendering multiple spats

- Culling
  - Exclude splats outside of the frame
- Render each splat from closest to farthest
  - Blend frame with alpha-compositing

$$C = \sum_i C_i \alpha_i \prod_{j<i} (1 - \alpha_j)$$



Pseudo-Code

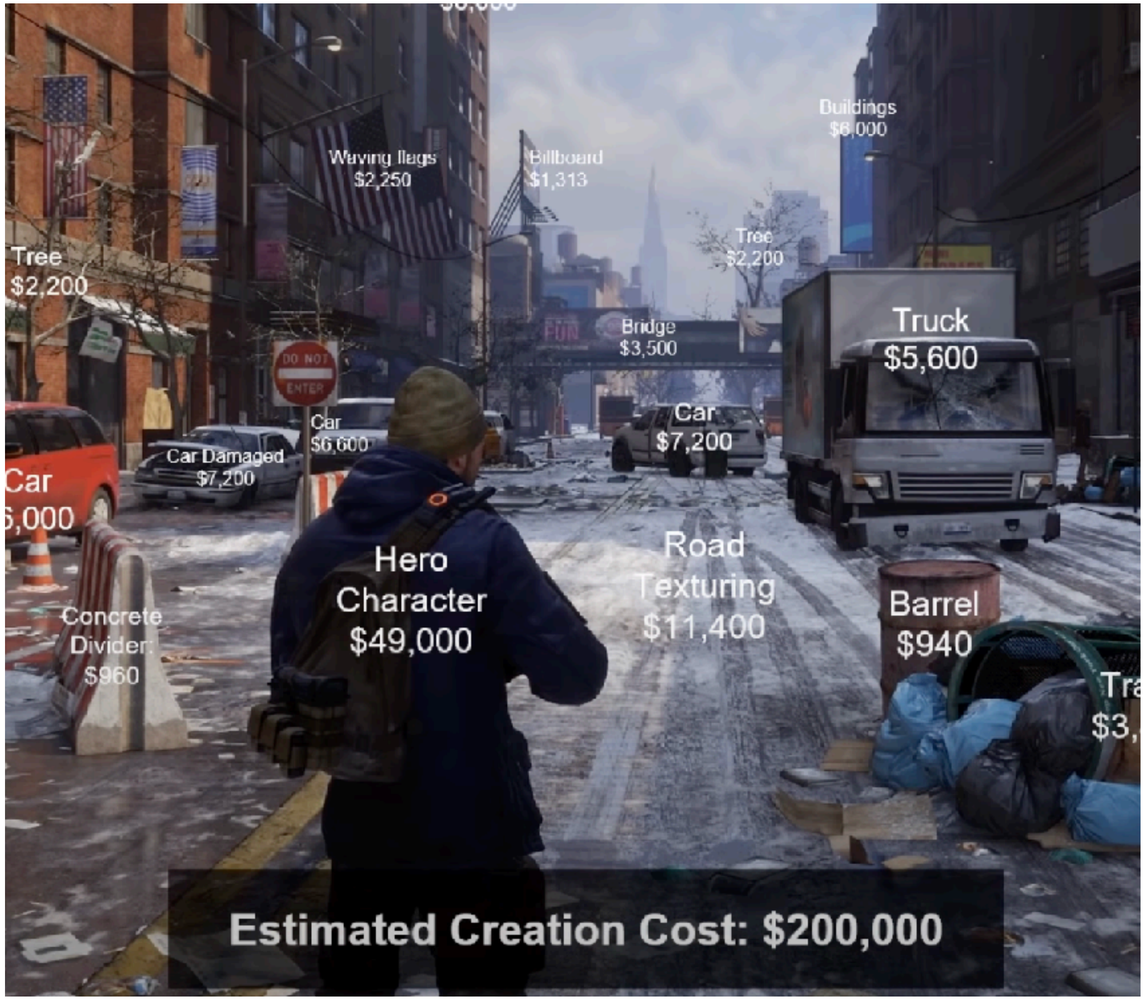
Why Rasterization is so Efficient?

```
for each pixel  # NeRF, 10e6 pixels
  for each segment on a ray  # 10e2 ray segments
    compute & accumulate radiance


for each splat  # Gaussian splatting, 10e4 splats
  for each pixel on a splat  # 10e2 pixels in a splat
    compute & accumulate radiance
```

Generation of 3D Scenes

Motivation



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

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

Setup

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

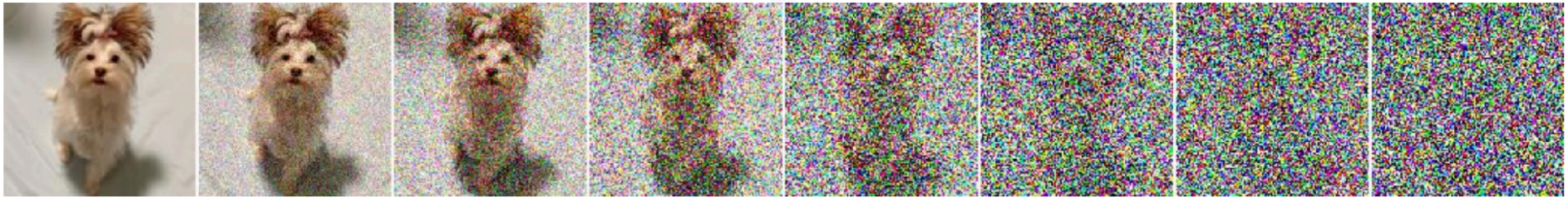
  A DSLR photo of a squirrel wearing a kimono reading a book
- Output: 3D model

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

Diffusion Models Recap

Mapping data to noise

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



Trained reverse map

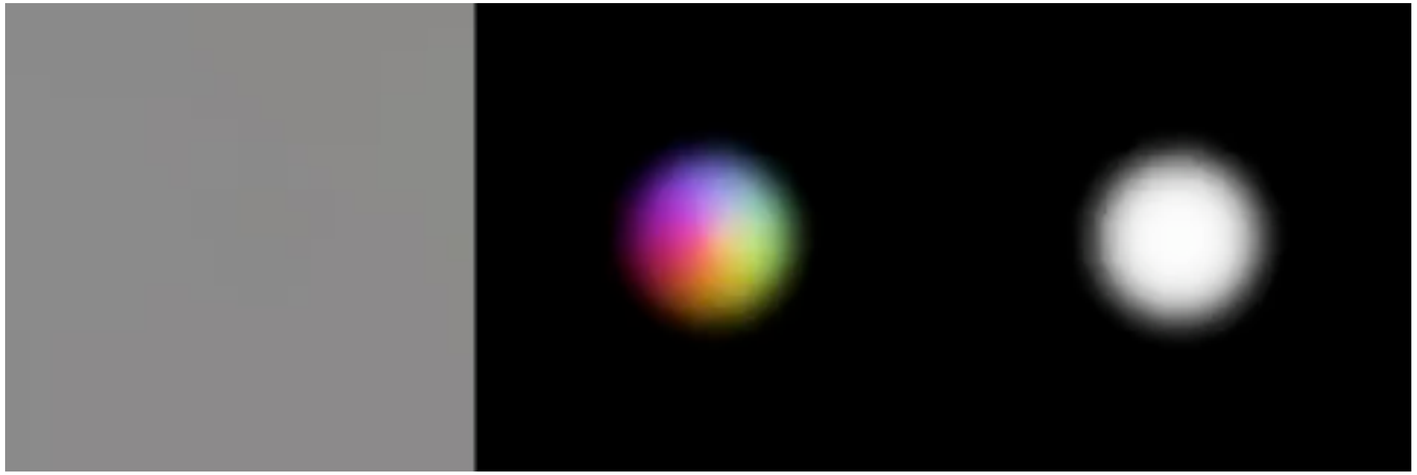
$$\hat{x}_\phi(x_t, t) = \frac{(x_t - \sigma_t \varepsilon_\phi(x_t, t))}{\alpha_t}$$

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

https://arxiv.org/abs/2011.13456

Training Dreamfusion

- Choose prompt y_t (omitted for clarity)
- Training loop:
  - Sample camera position z
  - Render image $x_0 = g(\theta, z)$
  - Sample t, x_t
  - Estimate $x_\phi(x_t, t)$
  - Make a gradient step to minimize



$$w(t)\Big(x_0 - \text{detach}(x_\phi(x_t, t))\Big)^2$$

Tackling Low Sample Diversity

Input images

in the Acropolis

swimming

sleeping

in a doghouse

in a bucket

getting a haircut

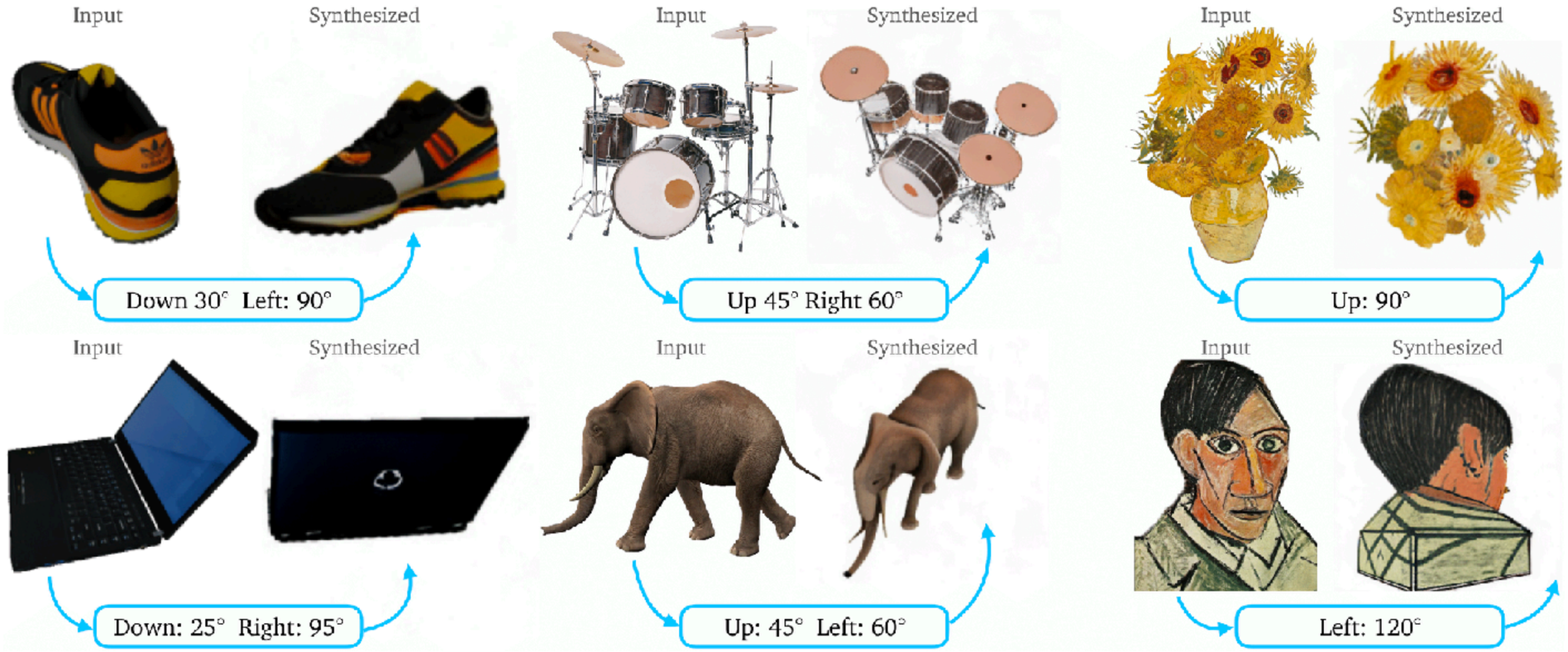
Dreambooth 3D

https://dreambooth3d.github.io/

Tackling Janus Problem

- Janus problem
- First remedy: specify camera angle in prompts

Zero-123: viewangle control

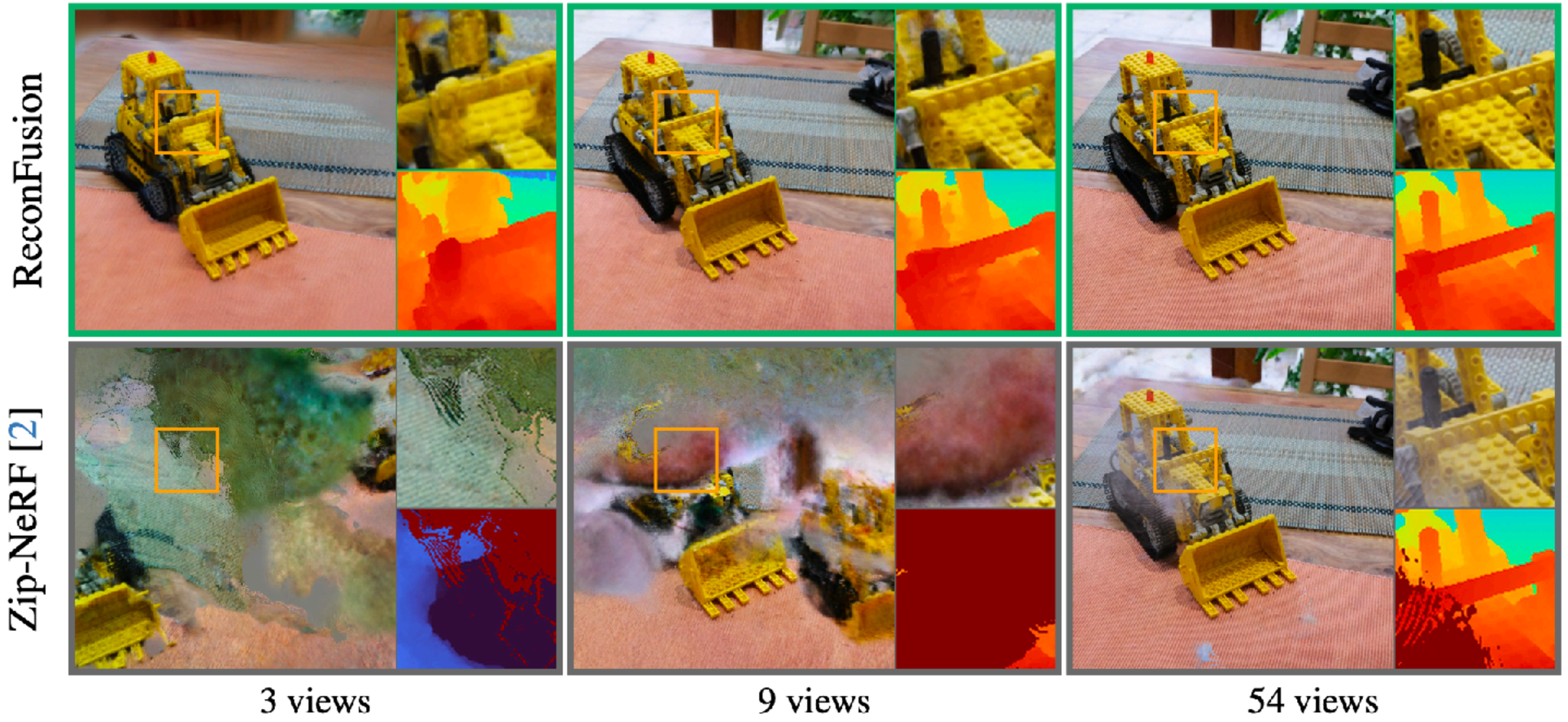


https://zero123.cs.columbia.edu/

Zero-123: examples

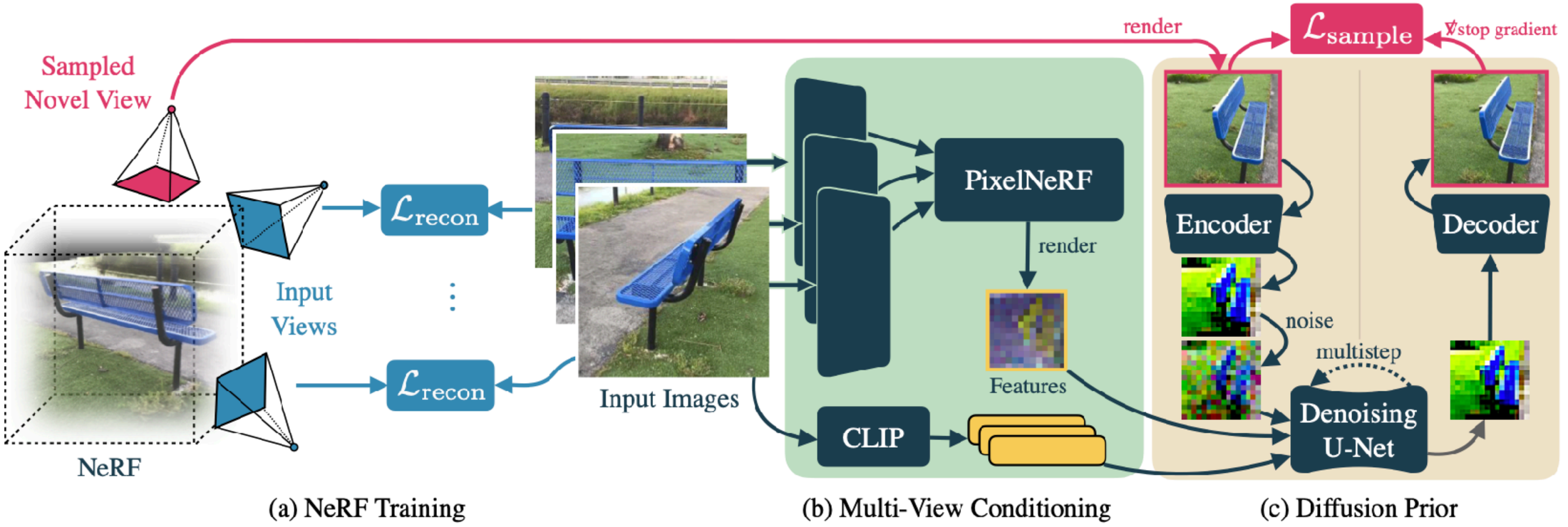
https://zero123.cs.columbia.edu/

Reconfusion



https://reconfusion.github.io/

Overall Scheme



(a) 3 input views

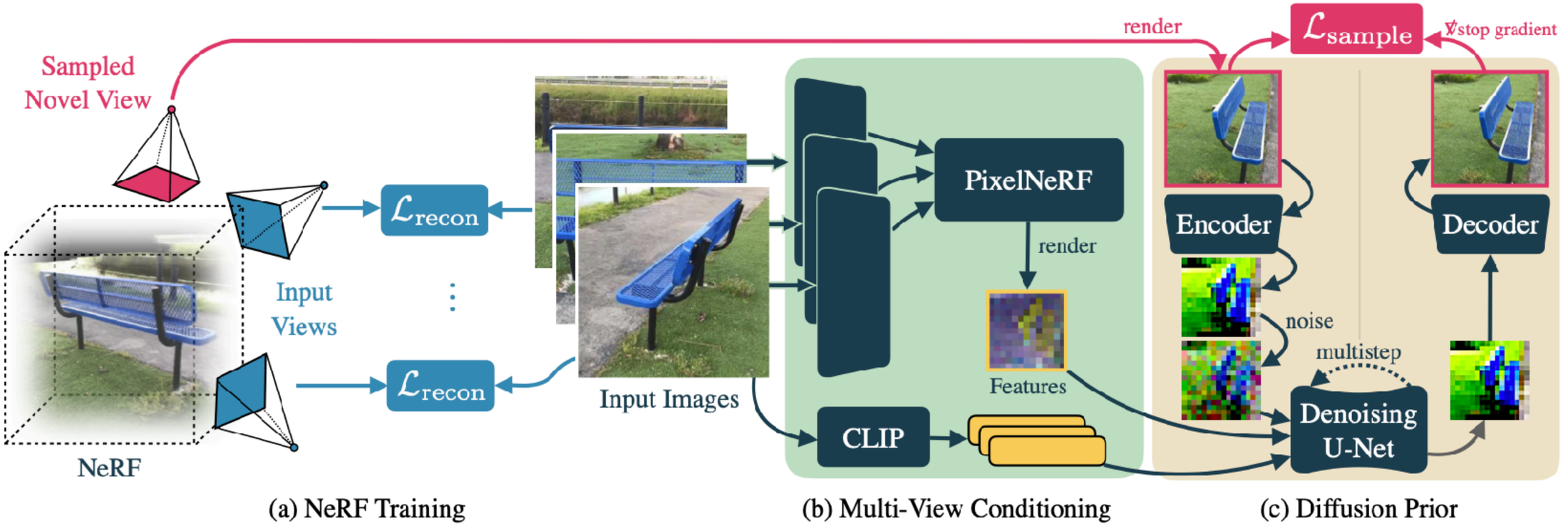
(b) 6 input views

(c) 9 input views

(d) Ground truth

Pixel-NeRF for View Conditioning

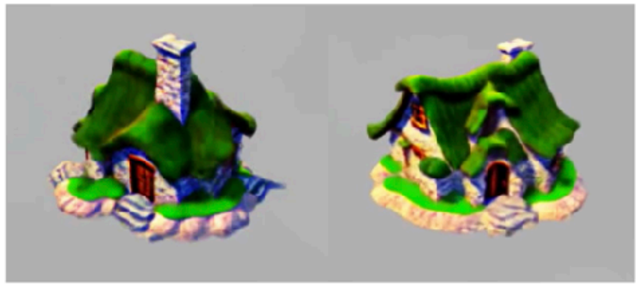
Regularising NeRFs with Diffusion

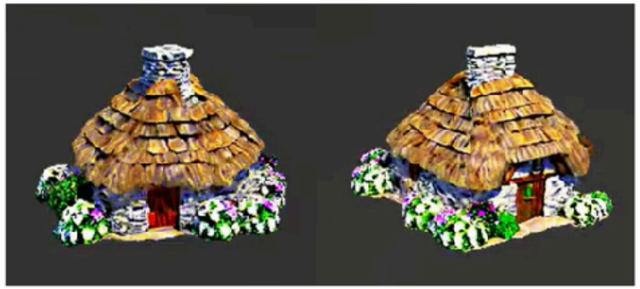


$$\mathcal{L}_{\text{Recon}}(\psi) = \mathbb{E}_{x^{\text{obs}},\pi^{\text{obs}}}\left[\ell(x(\psi,\pi^{\text{obs}}),x^{\text{obs}})\right] \qquad \mathcal{L}_{\text{sample}}(\psi) = \mathbb{E}_{\pi,t}\left[w(t)\left(\|x-\hat{x}_\pi\|_1 + \mathcal{L}_{\text{p}}(x,\hat{x}_\pi)\right)\right]$$


Zip-NeRF

Sora Generates Videos with Stunning Geometrical Consistency





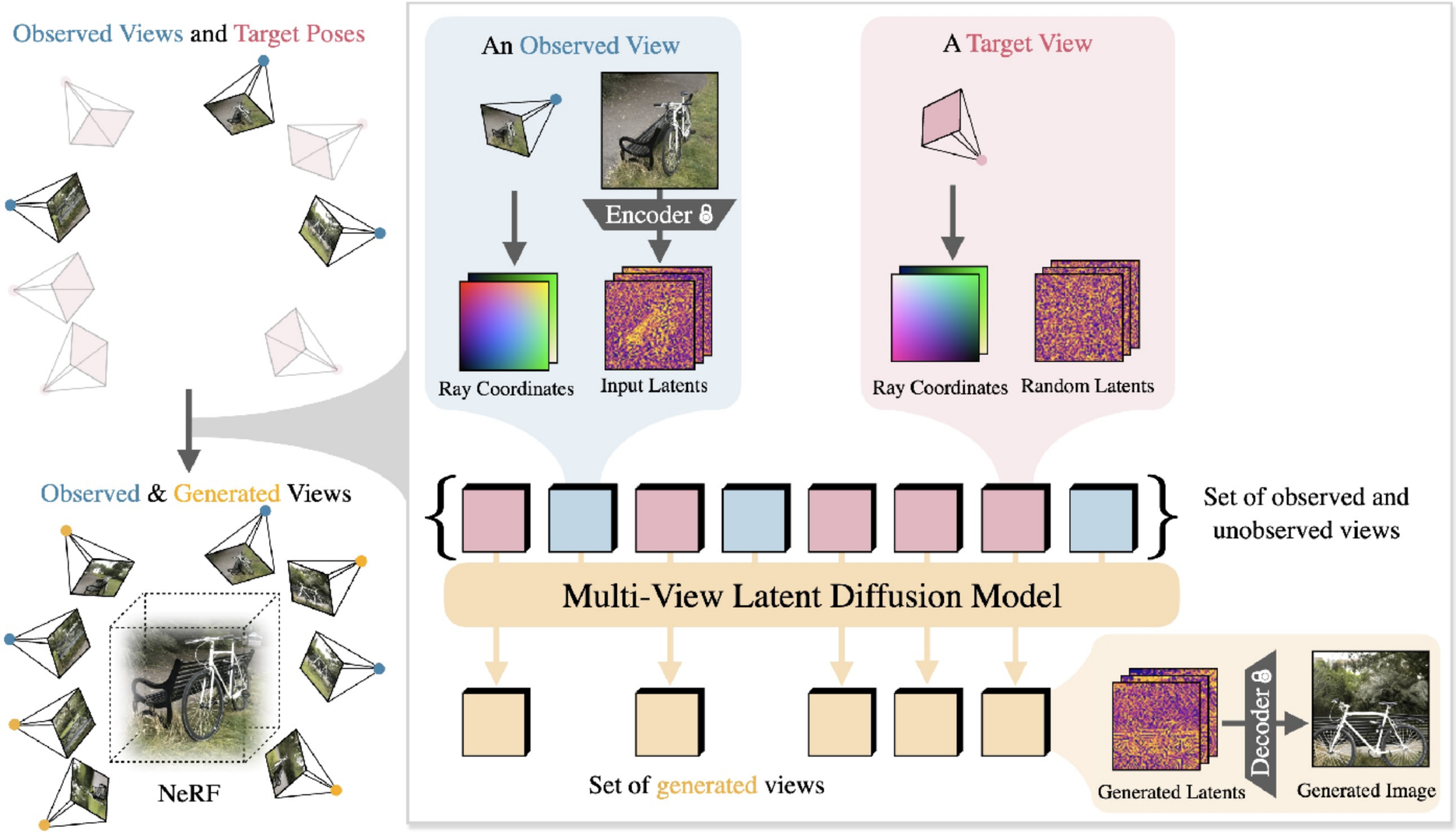
https://sora-geometrical-consistency.github.io/

Cat3D

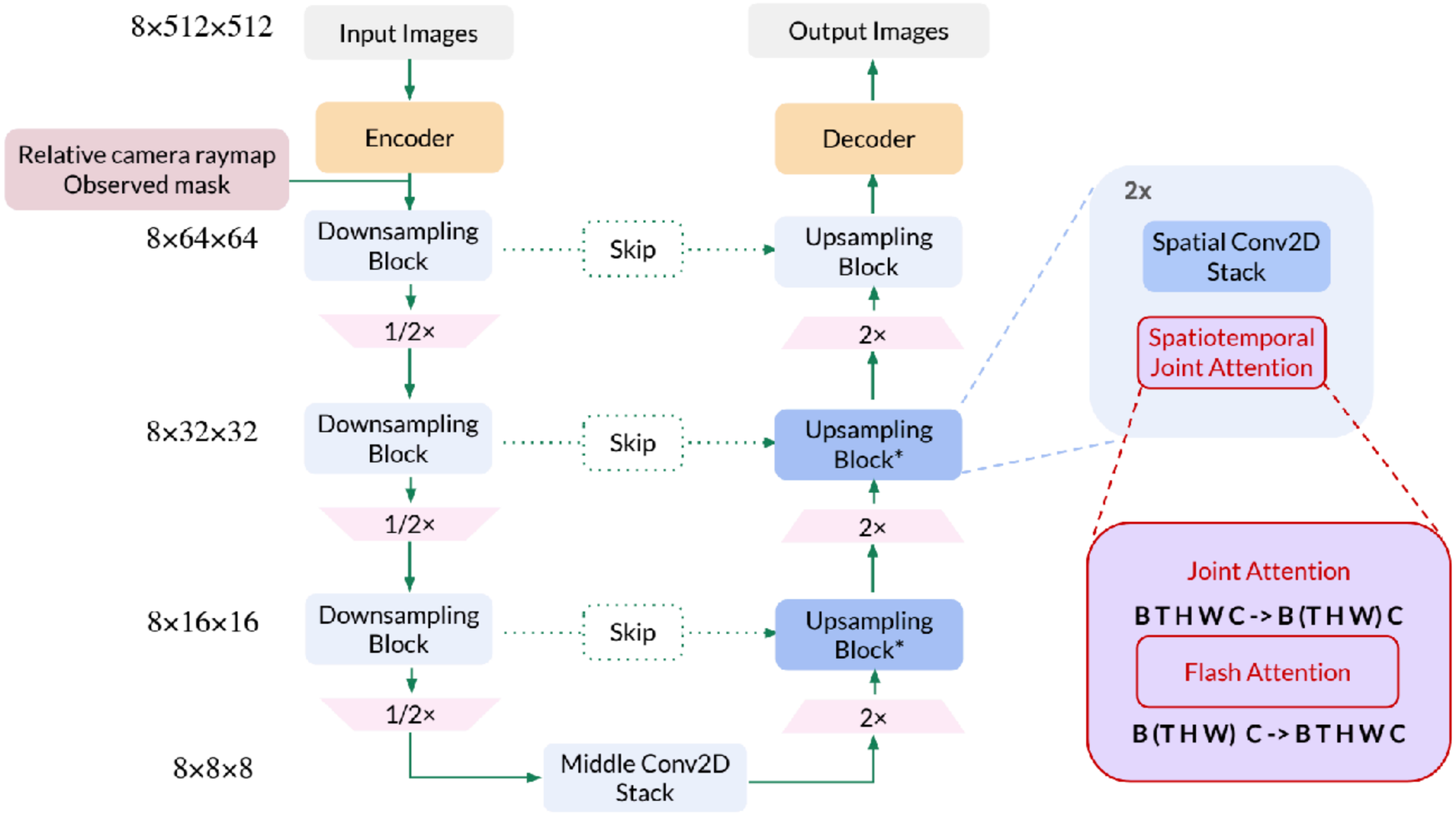
Main Contributions

- Novelty
  - Simpler generation pipeline
  - Simpler diffusion conditioning
- Not novel, yet important:
  - Simultaneous generation of views
- Tasks
  - Text-to-3d, Image-to-3D, Novel View Synthesis

Overall Scheme



Diffusion Model Architecture

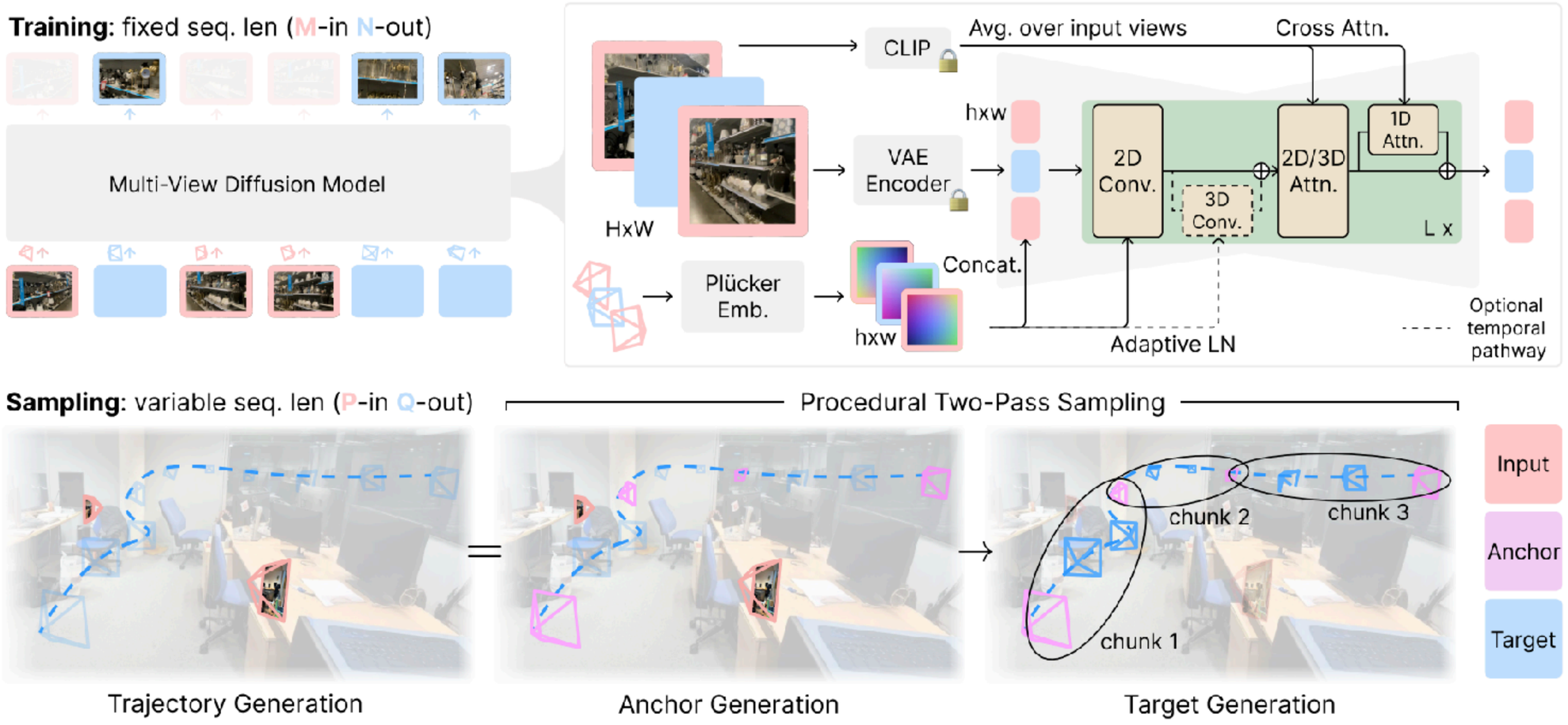


Training

- Initialise with a pertained text-to-image model
- Fine-tune *collect as much data as you can*
  - Objaverse
  - CO3D
  - RealEstate10k
  - MVImageNet

Stable Virtual Camera

Temporal Consistency, No 3D Distillation, Open Source



Stable Virtual Camera

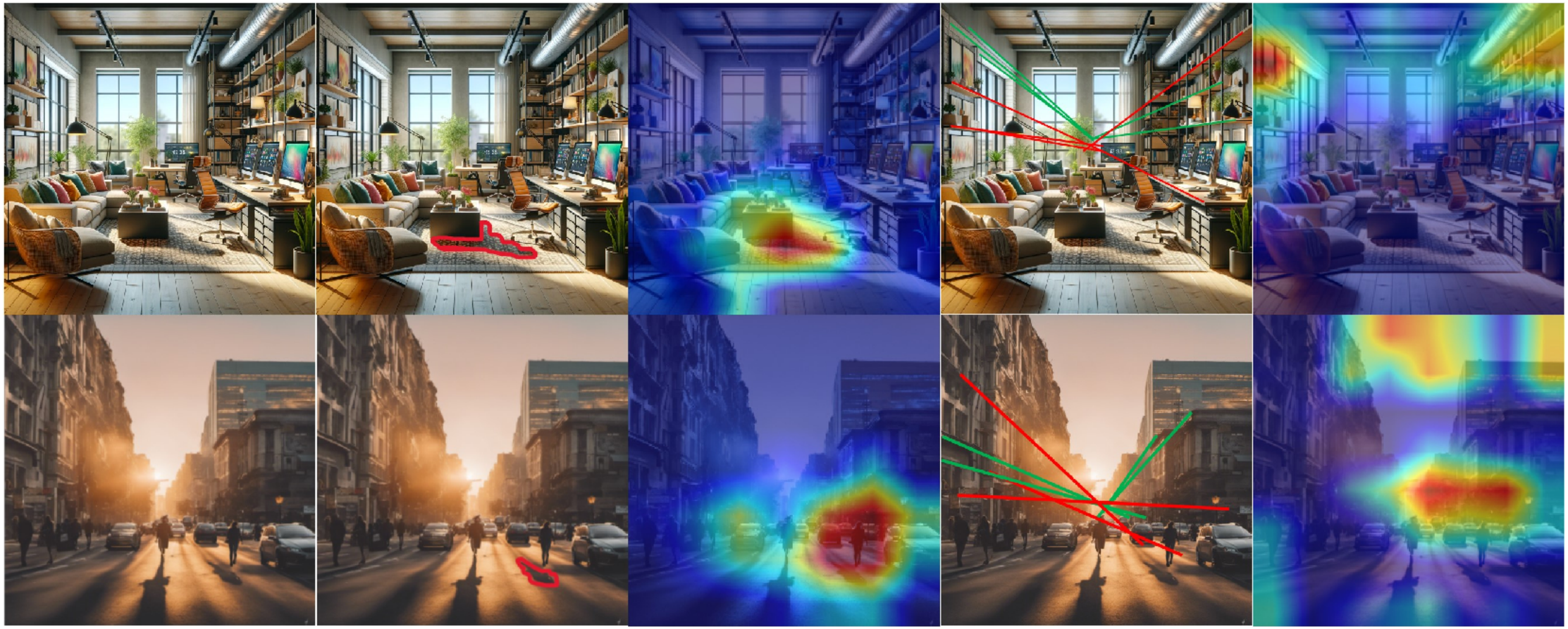
Single frame

Multiple Frames:

Stable Virtual Camera

Single frame

Shadows Don’t Lie and Lines Can’t Bend!
Generative Models don’t know Projective Geometry...for now



Generated Image | Shadow Errors | Detected Shadow Errors | Vanishing Point Errors | Detected Perspective Errors

https://projective-geometry.github.io/

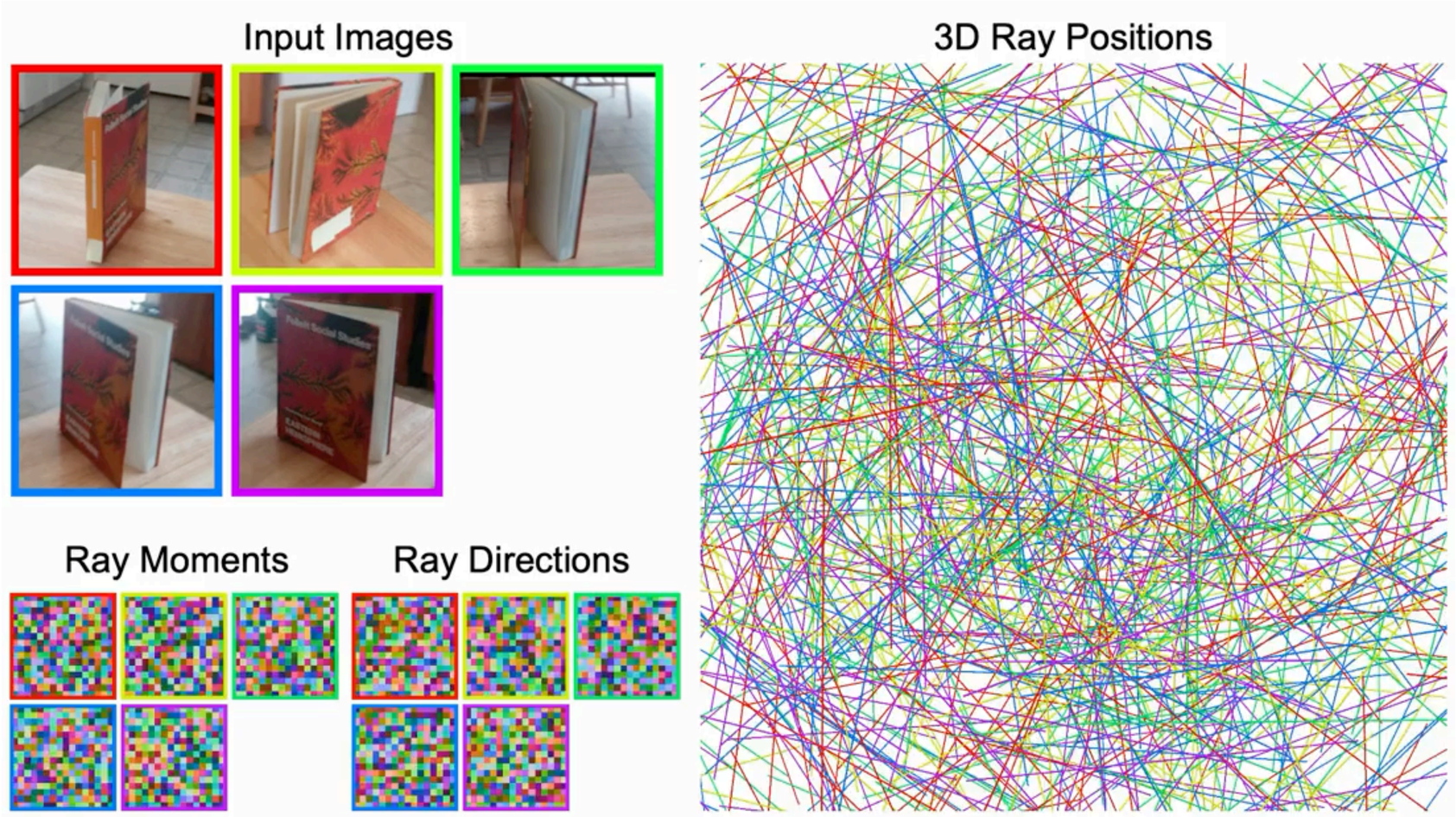
Other Applications

Depth Estimation

https://marigoldmonodepth.github.io/

Other Applications

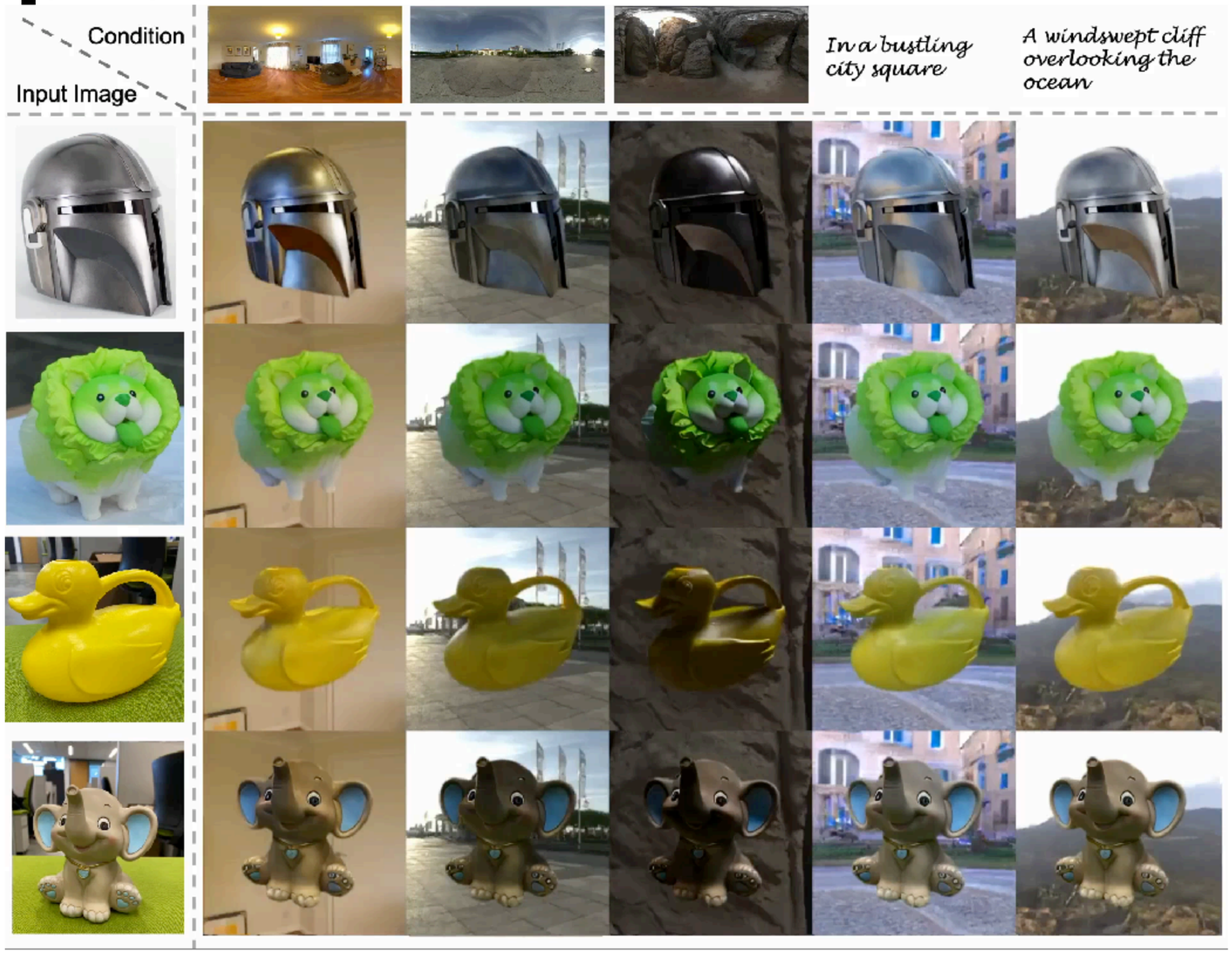
Camera Pose Estimation



https://jasonyzhang.com/RayDiffusion/

Other Applications

Relighting



https://neural-gaffer.github.io/

Other Applications

Environmental Light Estimation

https://diffusionlight.github.io/

Other Applications

Environmental Light Estimation

Conclusions

- Learning-based approaches to reconstruction & generation are booming
- Diffusion models offer a nice prior for visual content

- Multi-view and video models are taking over 3D content generation at the moment
- Graphic primitives provide a convenient representation for the content

- Multiple downstream tasks in graphics

Thank for your attention!