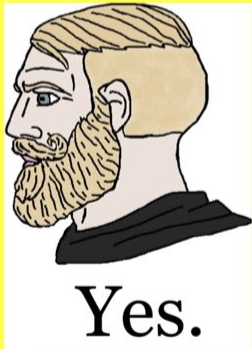



HSE

# 3D methods in CV

Michael Vasilkovsky

MLE @ Snapchat



# Plan

- Intro
- What is the lecture about?
  - What are 3D methods?
  - Applications
- Graphics primitives
  - Cameras, rays, geometry, bsdf, lighting
- Problems & papers
  - Structure-from-Motion
  - Novel View Synthesis (NeRF & modifications)
  - Differentiable & inverse rendering
  - 3D-aware generation
  - Further reading

# My background

MIPT
MOSCOW INSTITUTE OF PHYSICS AND TECHNOLOGY

ШКОЛА АНАЛИЗА ДАННЫХ

Skoltech

NEUROMATION

Dbrain

*tg/in/inst/gmail:* ***waytobehigh***

# What do I mean by "3D methods"?

*Let's define it as "any methods that use explicit 3D prior"*

# What do I mean by "3D methods"?

*Let's define it as "any methods that use explicit 3D prior"*

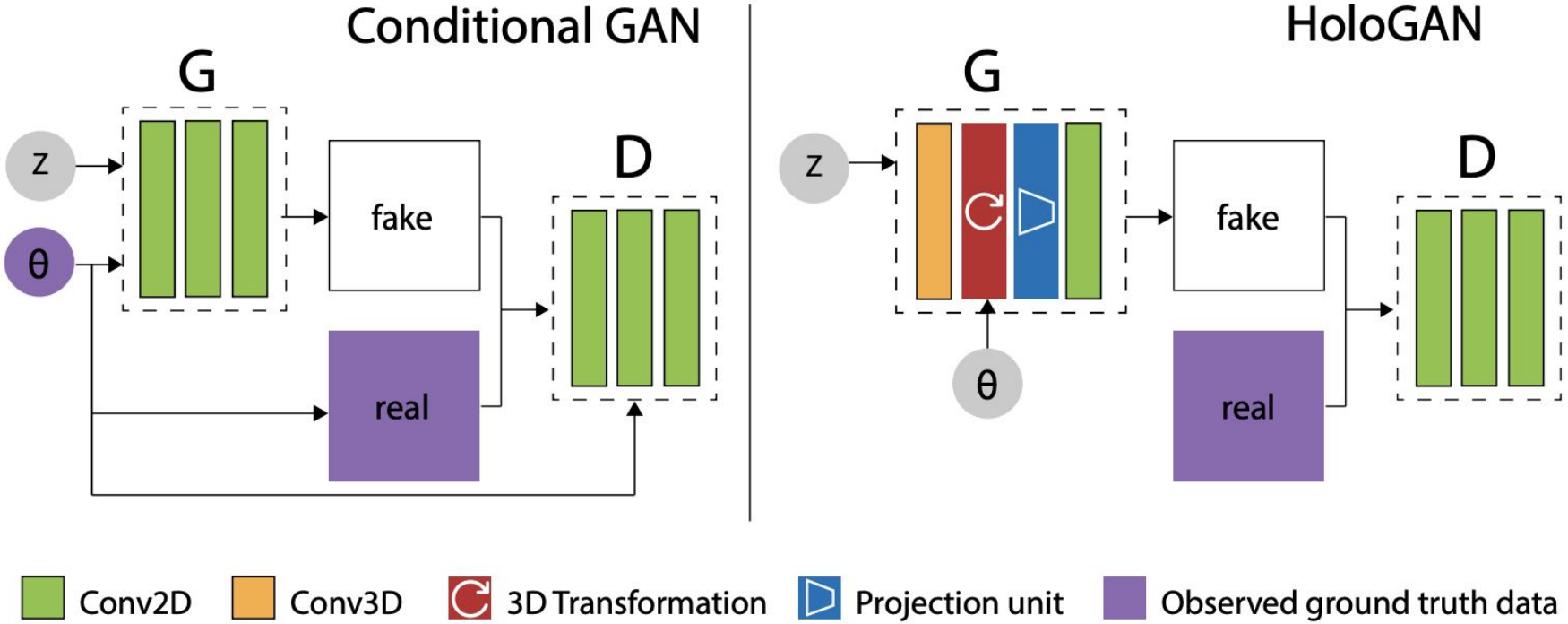
*... but why do we need it?*

# Example task

*Let's spin some generated heads…*

GRAM (2022)

# Example task



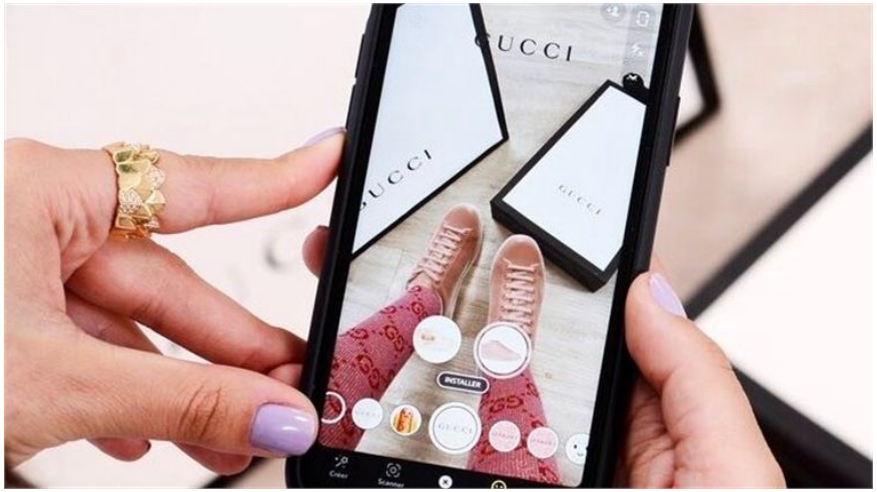

HoloGAN (2019)

# Example task

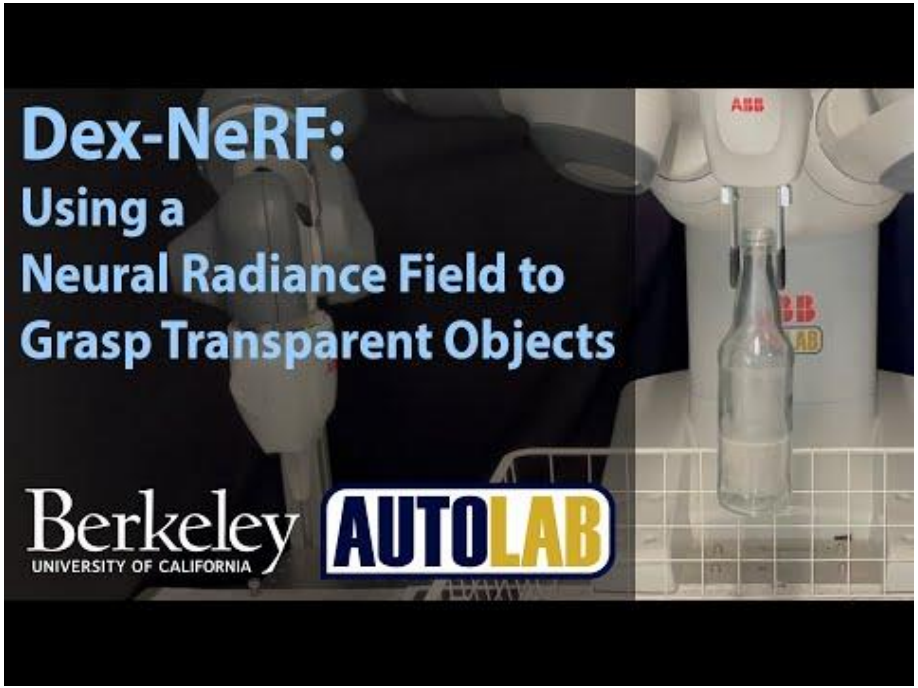
HoloGAN (2019)

# Applications

# Applications: AR



# Applications: self-driving & robotics

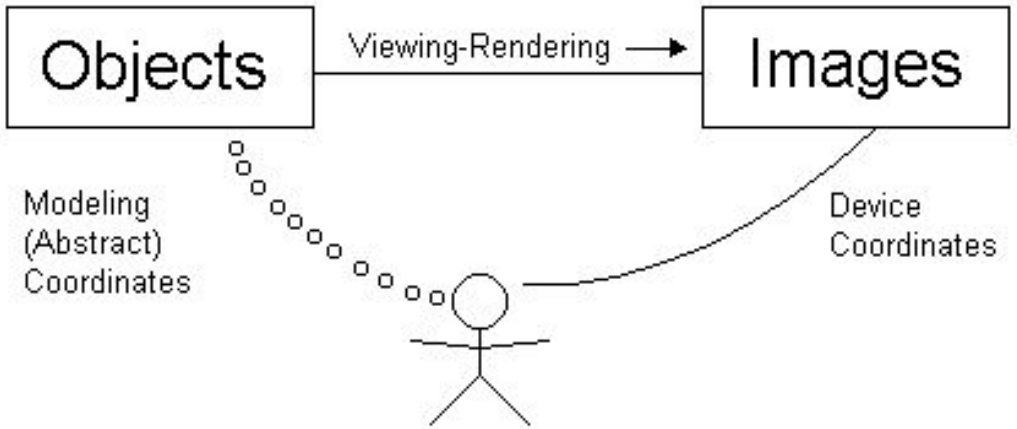



# Applications: realistic rendering

# Graphical primitives

*Some cool concepts are coming.*

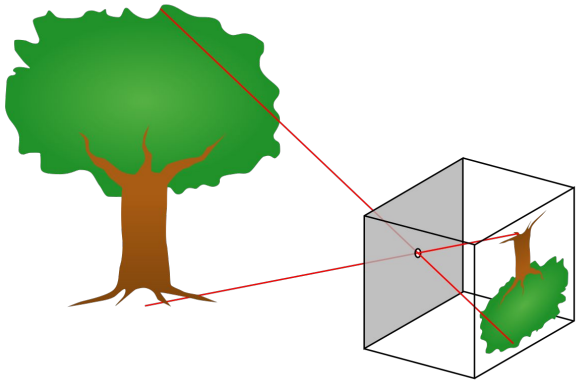
# Graphics

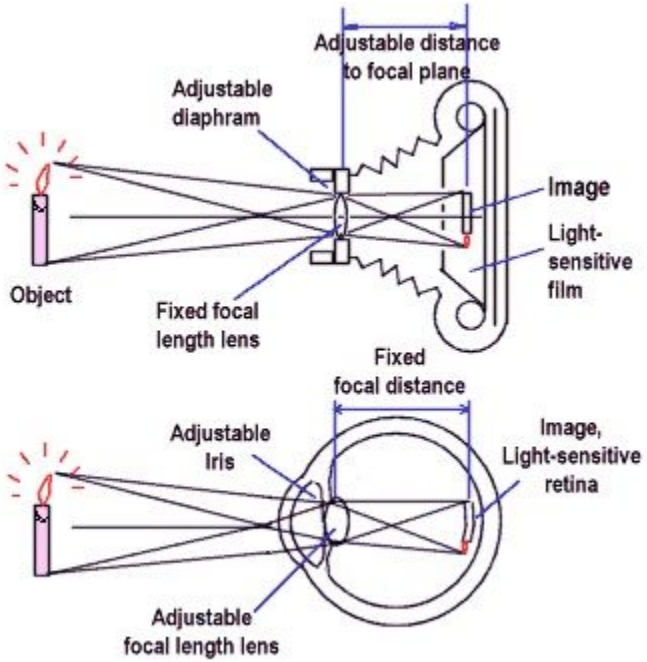

Objects
Viewing-Rendering
Images
Modeling
(Abstract)
Coordinates
Device
Coordinates

# Scene

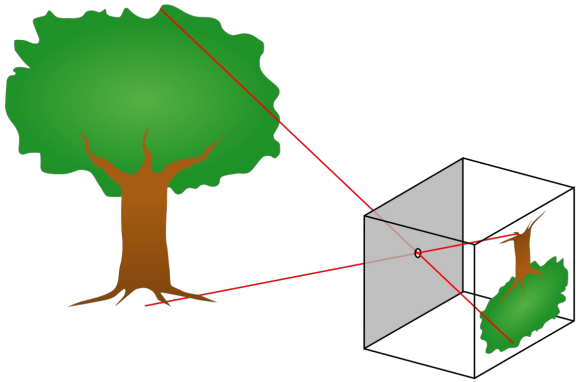
*There are many ways to represent the same scene*

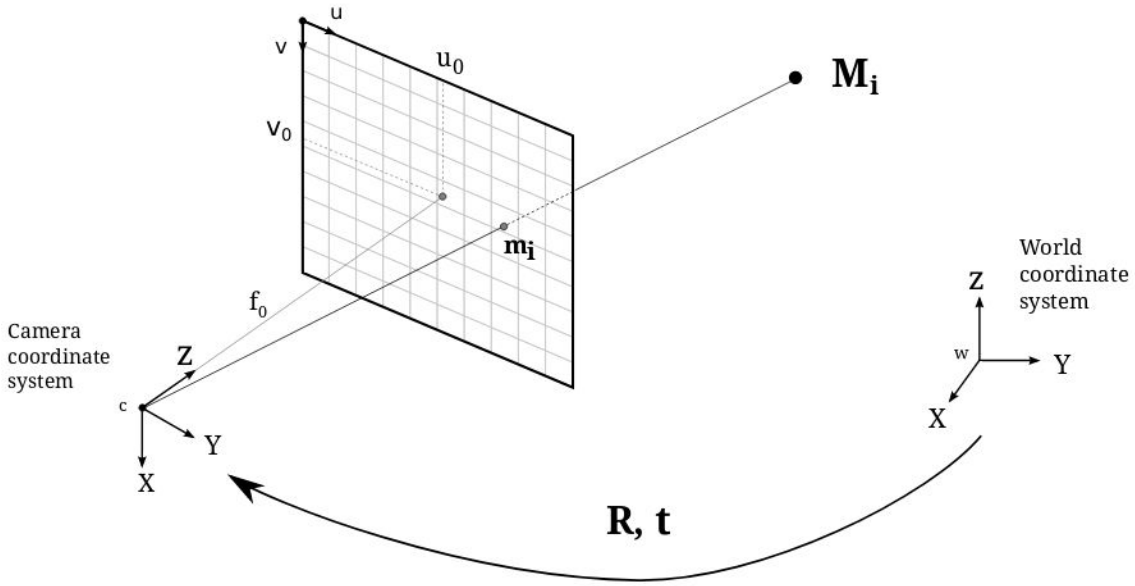
# Camera: real world



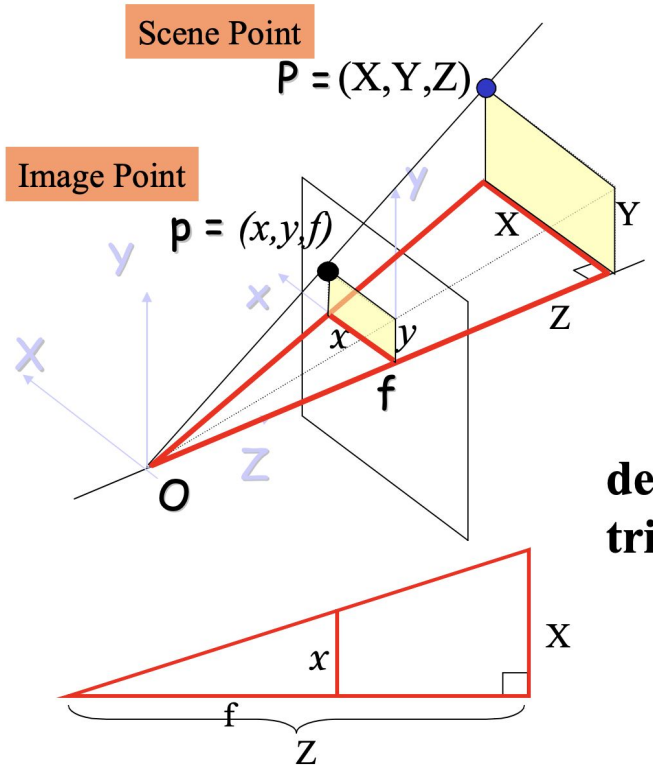


# Camera: model





# Camera: intrinsics




Perspective Projection Eqns

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

**derived via similar triangles rule**

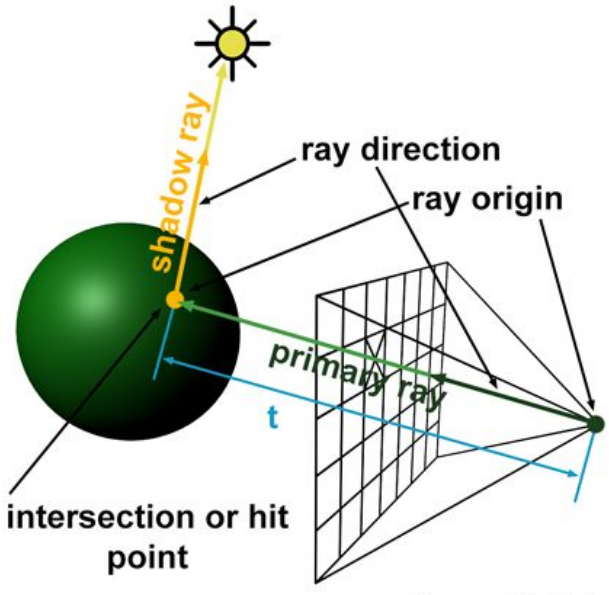
$$x = f\frac{X}{Z} \quad y = f\frac{Y}{Z} \quad \Leftrightarrow \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

x_true = x' / z'
y_true = y' / z'
z_true = z' / z'

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

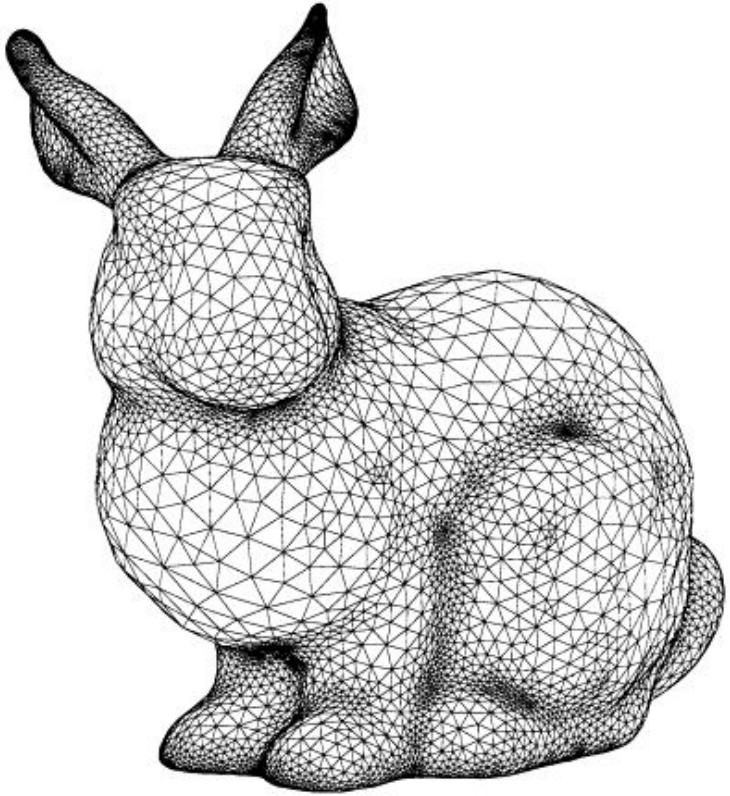
# Geometry vs. appearance

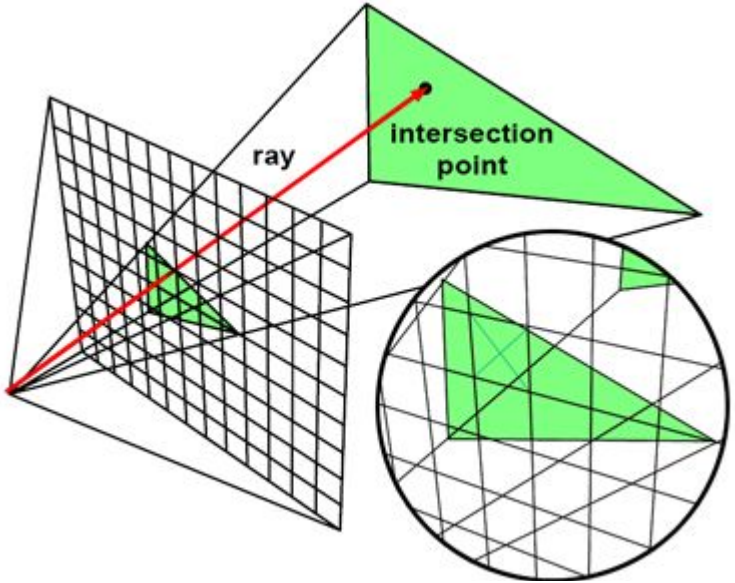
# Camera: raycasting



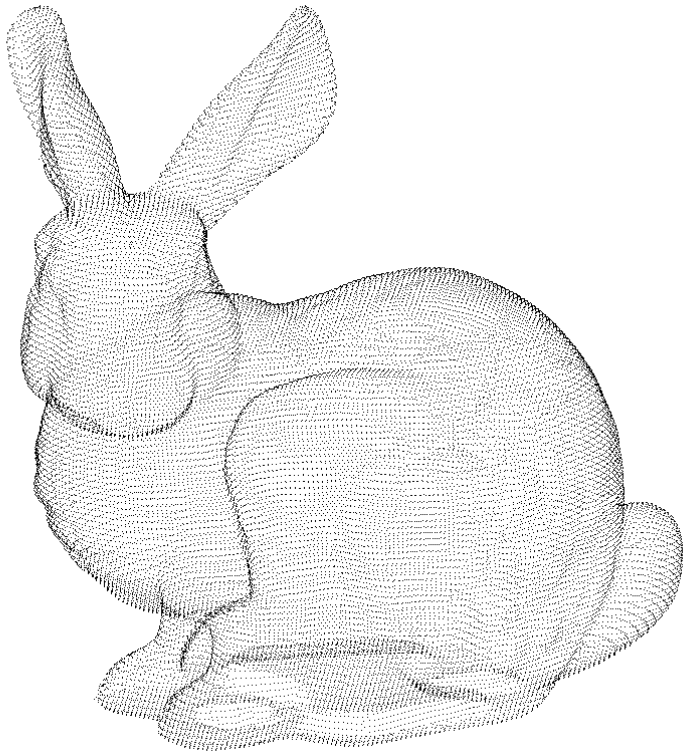

*To get pixel's color you need to aggregate info from the only ray.*

# Geometry: mesh





# Geometry: point cloud



Neural Point-Based Graphics

# Geometry: SDF

$$f : R^3 \rightarrow R$$

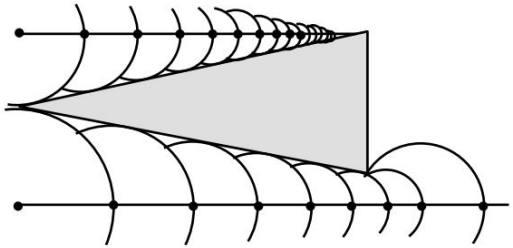
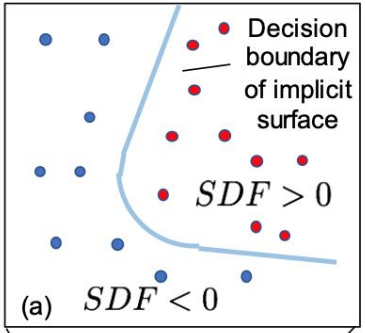
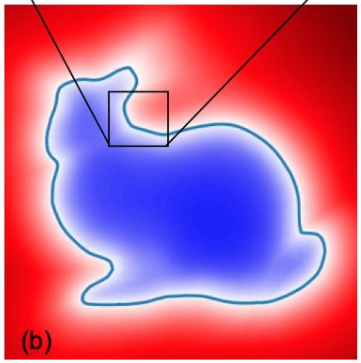


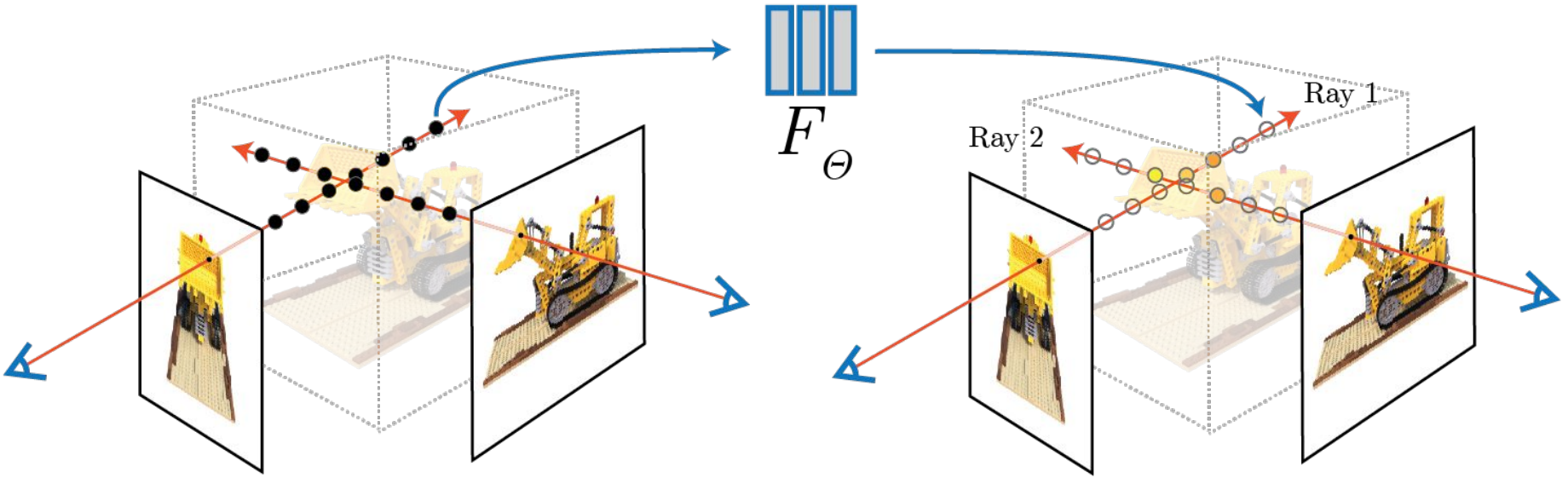
Figure 2: A hit and a miss.




(c)

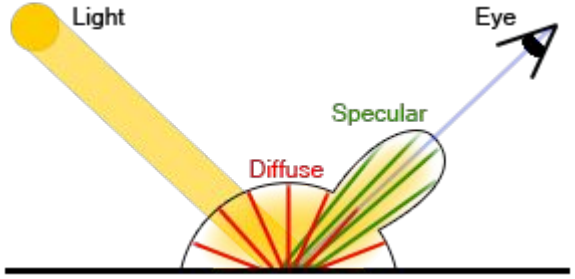
# Geometry: volumetric



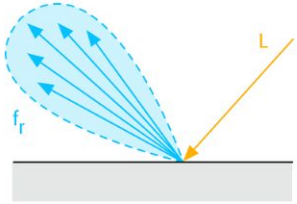

$$\sigma : R^3 \rightarrow R$$
$$c : R^3 \rightarrow R^3$$

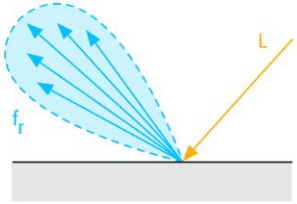
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

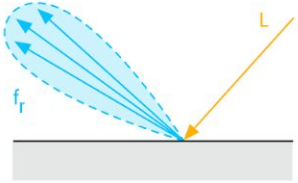
# Appearance: SVBRDF & materials

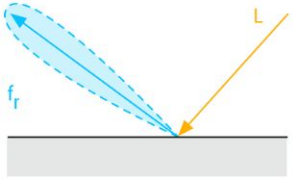


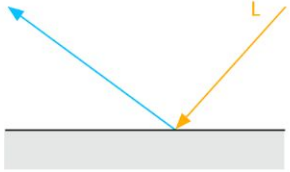

[Disney BRDF](): albedo, roughness, metallicity

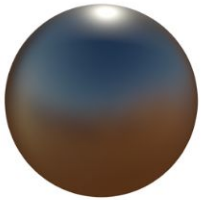


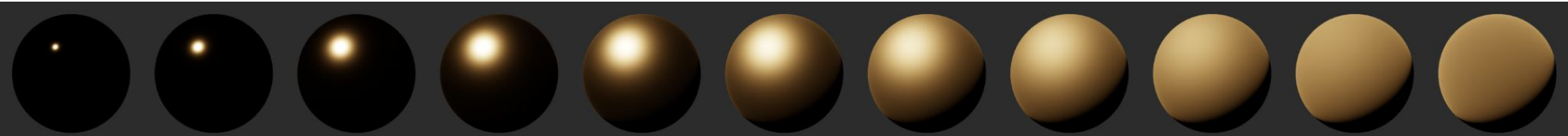



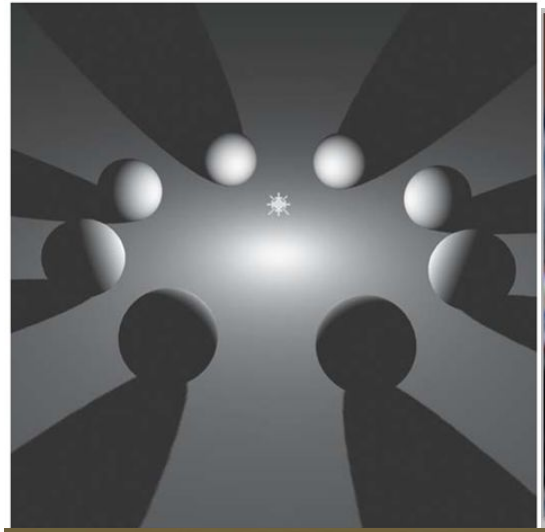





# Appearance: lighting

1. Environment maps
2. Spherical gaussians
3. Spherical harmonics
4. Point lights
5. Directional lights
6. Ambient light
7. ...



# Problems & papers

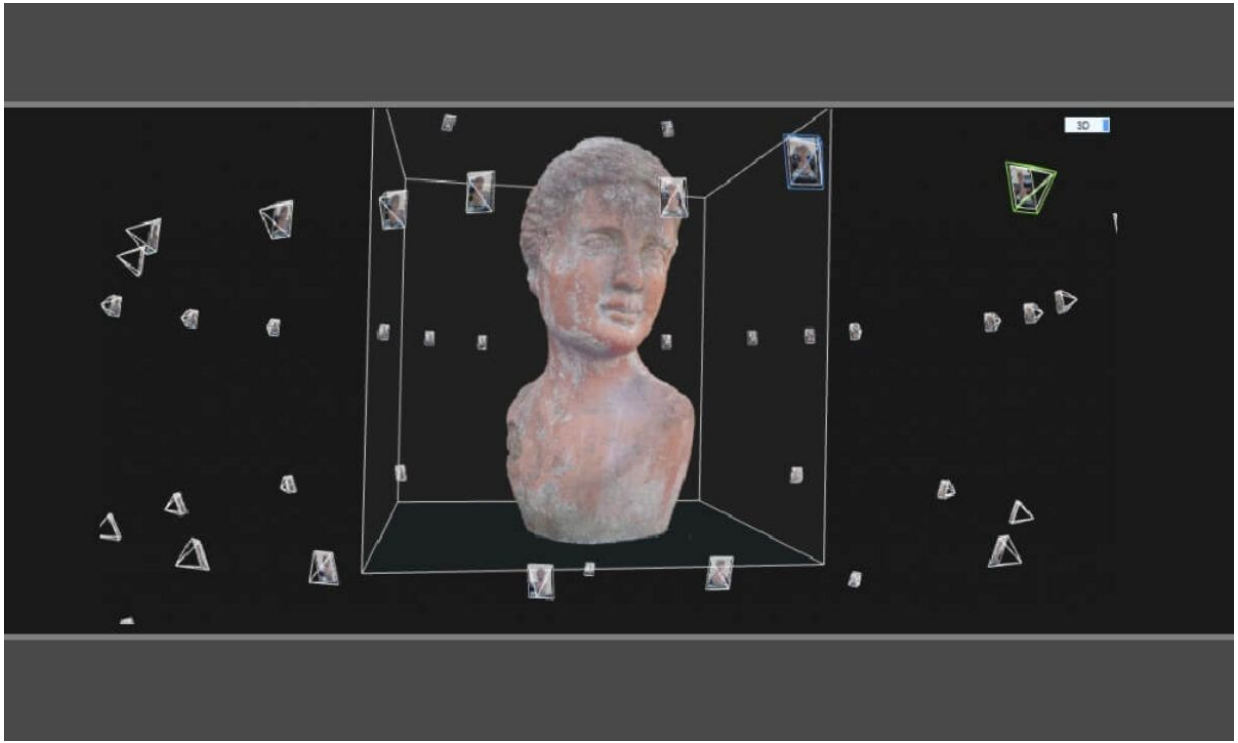
*Brace yourself.*

# Structure-from-Motion

**Goal:** reconstruct camera positions, intrinsics, mesh

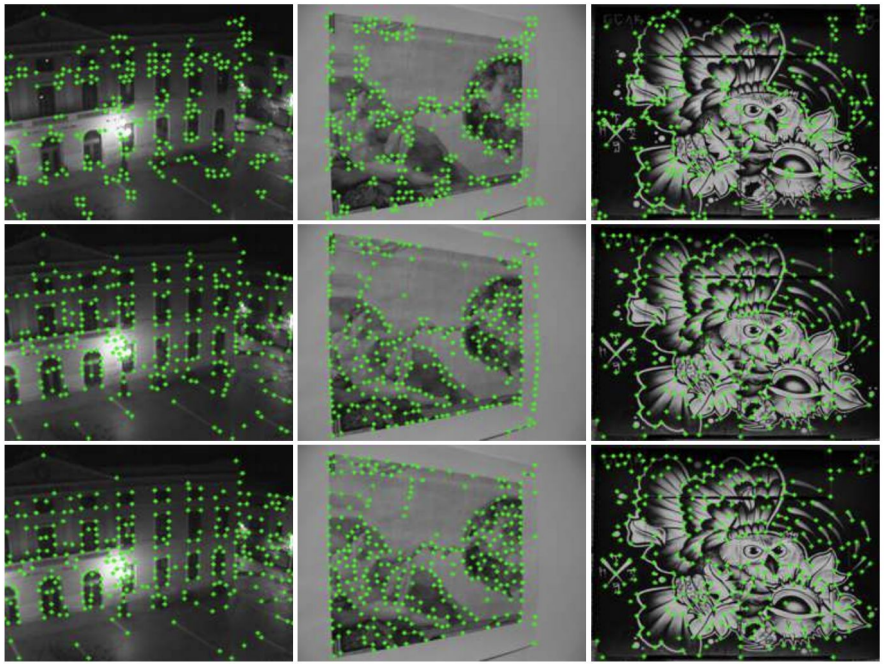
**Methods:**
– COLMAP
– Reality Capture
– pixsfm
– hloc

**Data:** >50 photos, typically 200-400



# Structure-from-Motion: steps

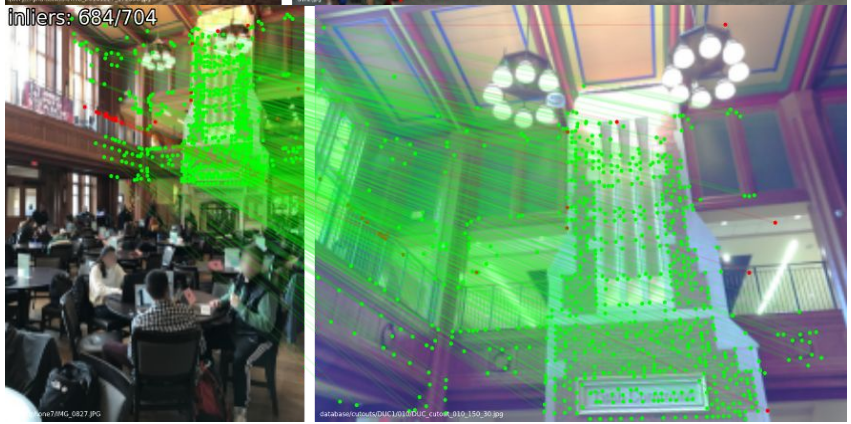
- **Extract keypoints from images** (SIFT, SuperPoint, SD2Net, …)



# Structure-from-Motion: steps

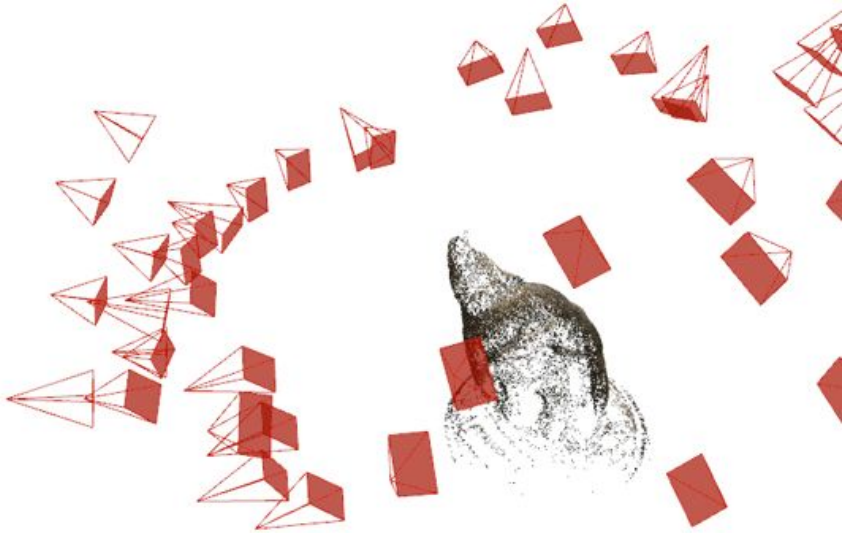
- Extract keypoints from images
- **Match images based on keypoints** (Exhaustive/Vocab/GD, SuperGlue)





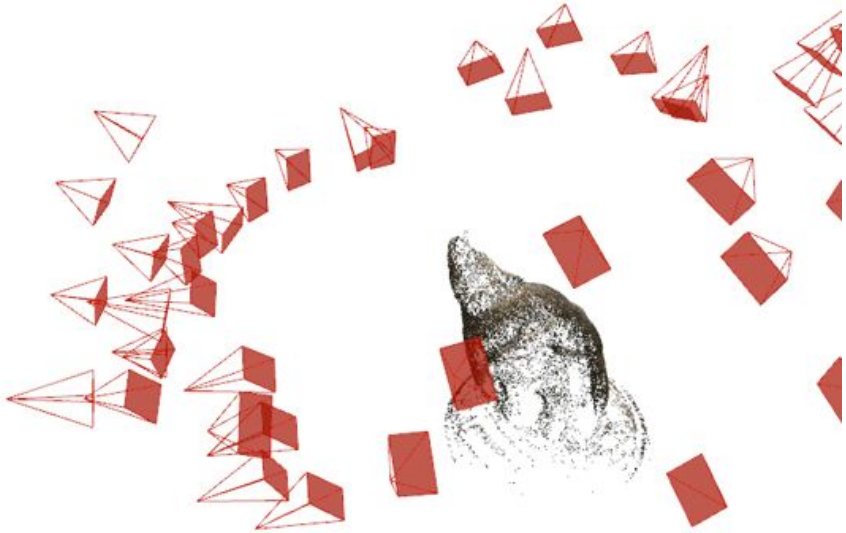
# Structure-from-Motion: steps

- Extract keypoints from images
- Match images based on keypoints
- **Iteratively add cameras and adjust them based on matches**
  (COLMAP / hloc / Reality Capture)



# Structure-from-Motion: steps

- Extract keypoints from images
- Match images based on keypoints
- Iteratively add cameras and adjust them based on matches
- **Sometimes bundle-adjust the entire scene & remove outliers**

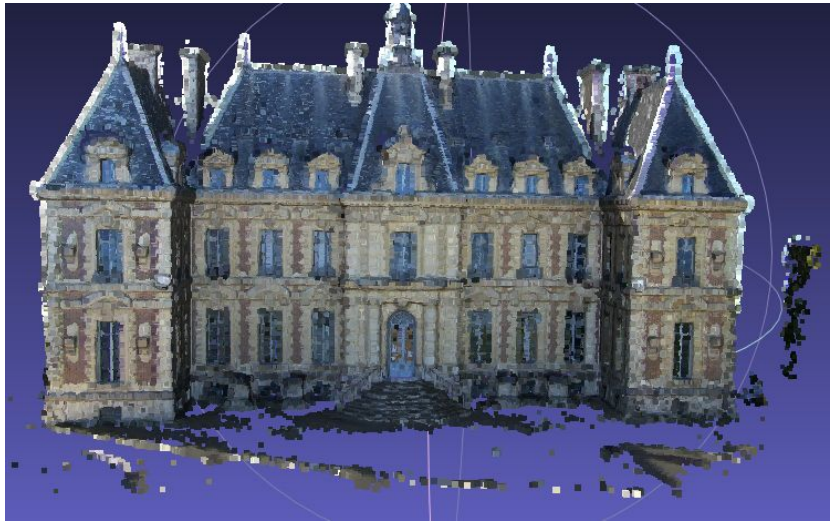


# Structure-from-Motion: steps

- Extract keypoints from images
- Match images based on keypoints
- Iteratively add cameras and adjust them based on matches
- Sometimes bundle-adjust the entire scene & remove outliers
- **Get dense point cloud**

# Structure-from-Motion: steps

- Extract keypoints from images
- Match images based on keypoints
- Iteratively add cameras and adjust them based on matches
- Sometimes bundle-adjust the entire scene & remove outliers
- Get dense point cloud
- **Triangulate mesh (poisson/delannay)**



# Novel View Synthesis (NVS)

**Goal:** learn scene representation and render it

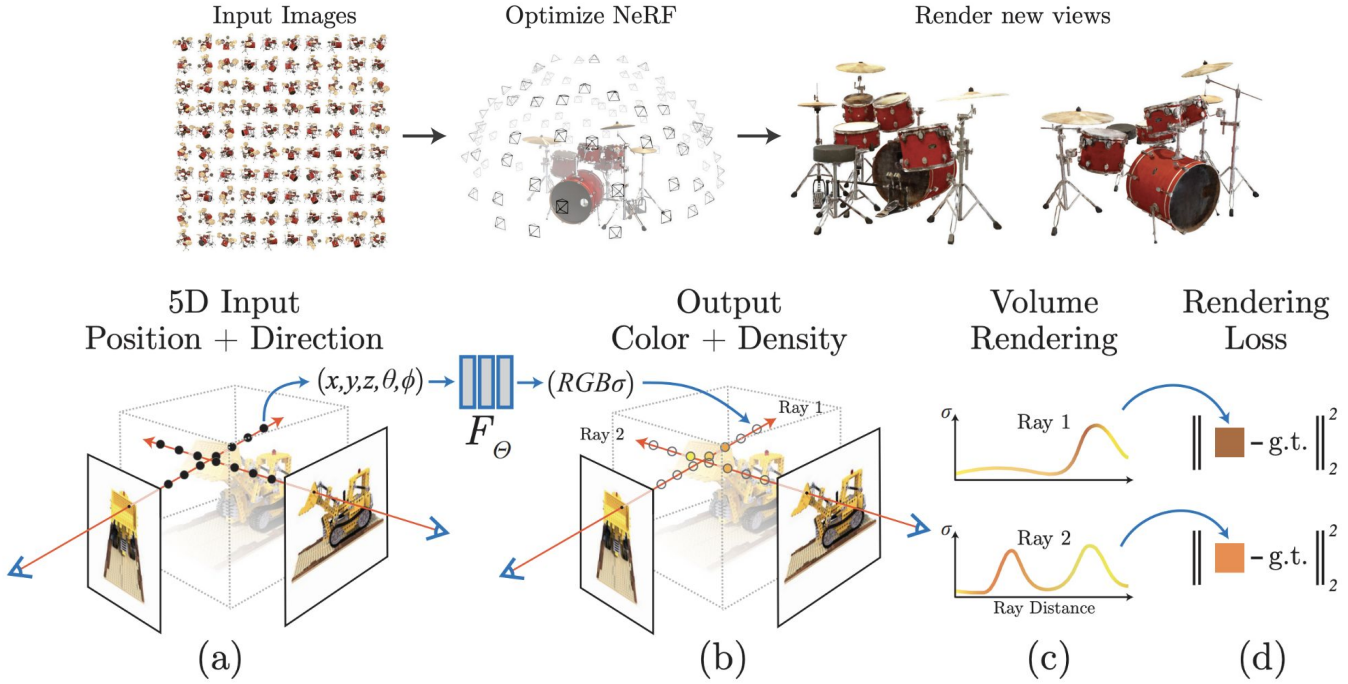
**Methods:**
– (mip-)NeRF
– instant-ngp
– NeuralMVS
and many more…

**Data:** typically ~100 photos, sometimes few shot

# NVS: NeRF

**Key takes**

1. Initially you have poses and images.
2. NeRF is trainable MLP predicting (sigma, color) in 3D point from a view (phi, theta). Sigma ~= density gradient.
3. To get a pixel color, trace a ray through it, evaluate (sigma, color) in N=128 points and compute integral by them.
4. Do a trick: sample uniformly in coarse network, then sample from sigma-based distribution in fine.
5. Train via pixel-wise MSE.




$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t}\sigma(\mathbf{r}(s))ds\right)$$

# NVS: NeRF tutorial

1. Capture ~100-120 photos sequentially.
    a. Be creative, closely capture high frequency details.
    b. Preserve lighting conditions and objects' placement on the scene.
    c. Fix camera parameters (exposure & stuff).
    d. Mb capture video and uniformly sample frames.
2. Run Colmap in exhaustive mode.
3. Train NeRF for a couple of days.
4. …
5. PROFIT

# ~~Daddy~~ NeRF issues

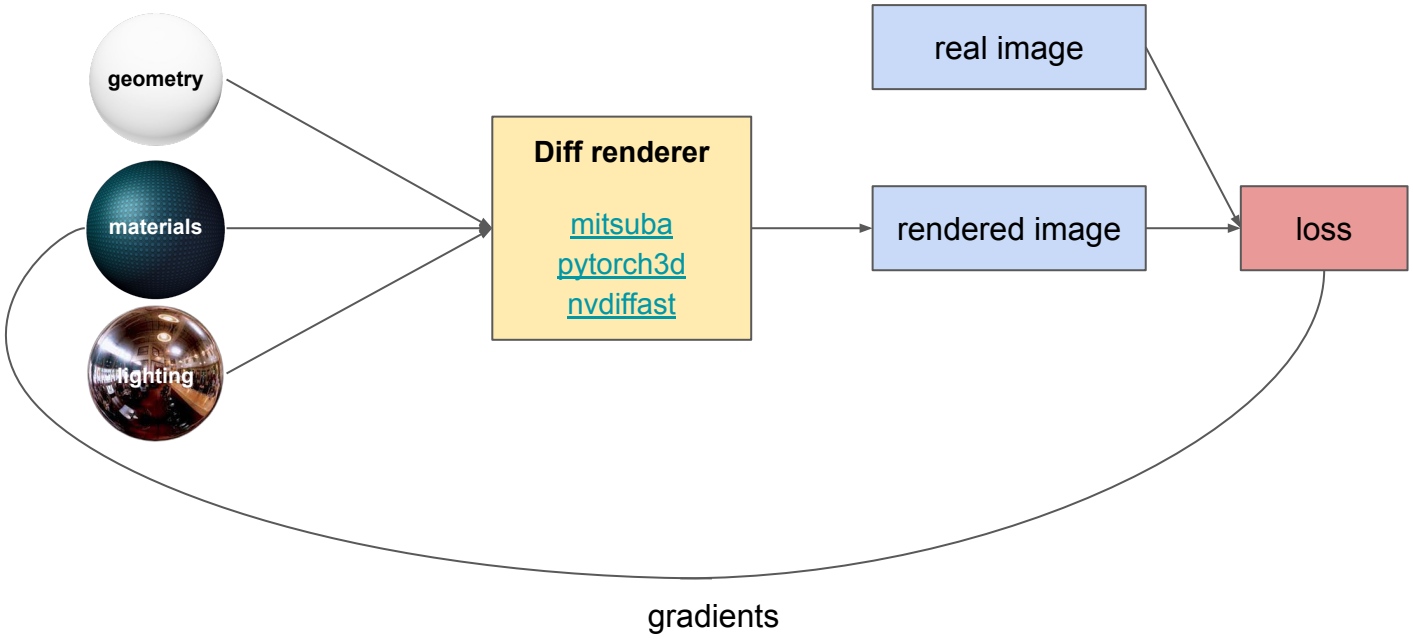
- 2+ days of training. (instance-ngp)
- Inaccurate camera poses worsen quality. (NeRF--, BaRF)
- Aliasing (mip-NeRF)
- No relighting. (NeRD, NeRV)
- Requires constant light setup while capturing. (NeRF-w)
- Typically can not train over entire image -> only pixel-wise losses. (instance-ngp)
- Slow inference. (instance-ngp)
- Is trainable per scene only. (PixelNeRF, NeuralMVS)

# Inverse rendering

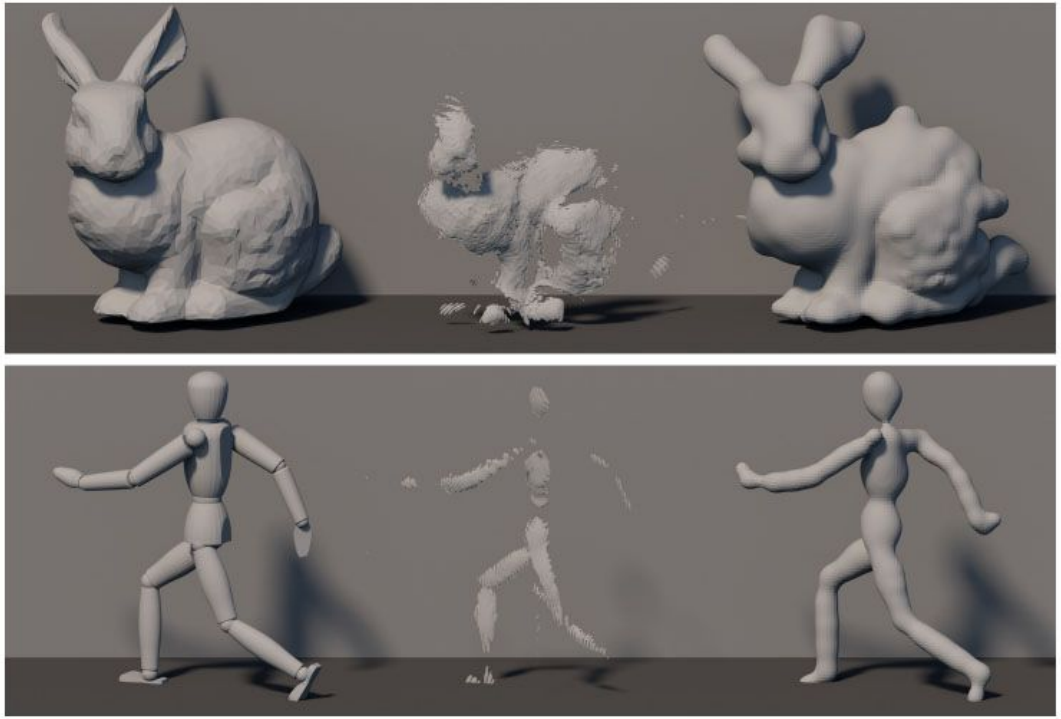
**Goal:** get a PBR scene from photos

**Methods:**
– mitsuba2
– nvdiffrast
– pytorch3d
– psdr-cuda

**Data:** typically ~100 photos

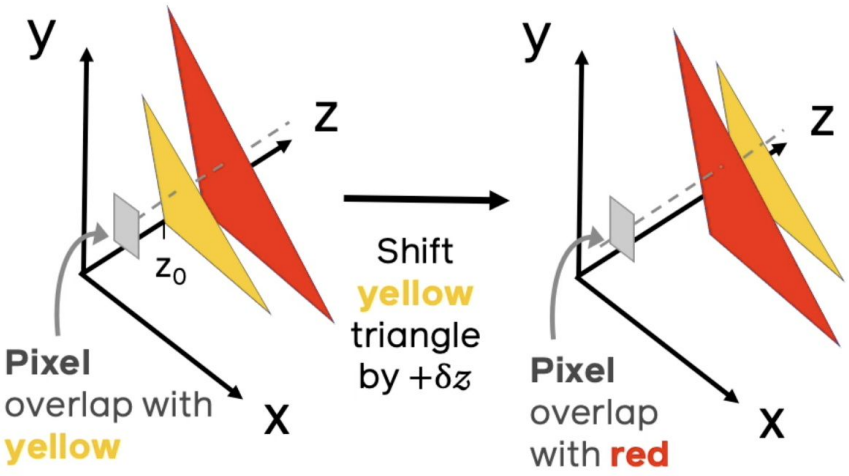


# Differentiable rendering is hard

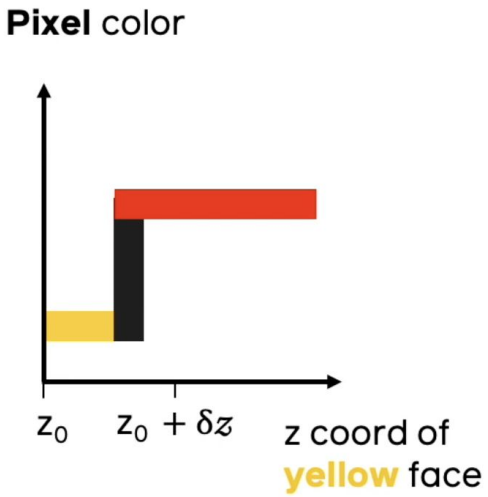


# Differentiable rendering is hard

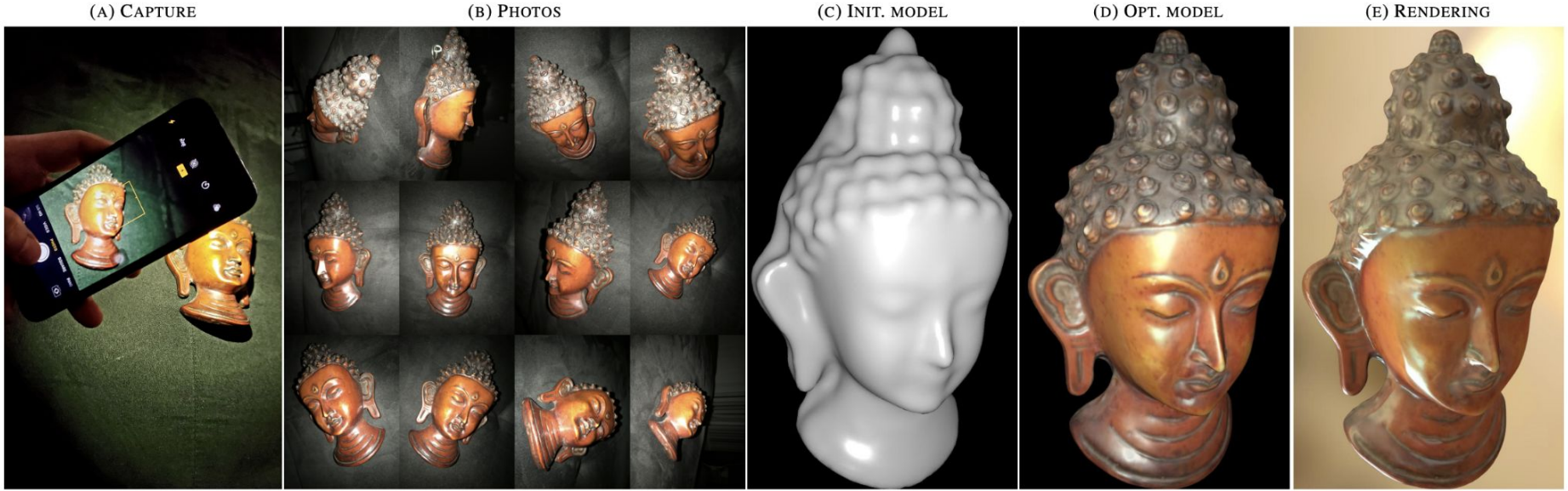
## Z discontinuity




## Step change in pixel color



# Inverse rendering




Luan et al

# Inverse rendering

$$\frac{\mathrm{d}I_j}{\mathrm{d}\boldsymbol{\xi}} = \frac{1}{|\mathcal{P}_j|}\left[\underbrace{\int_{\mathcal{P}_j} \frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{\xi}}(\boldsymbol{x})\,\mathrm{d}A(\boldsymbol{x})}_{\text{interior}} + \underbrace{\int_{\Delta\mathcal{P}_j} \left(\boldsymbol{n}(\boldsymbol{x}') \cdot \frac{\mathrm{d}\boldsymbol{x}'}{\mathrm{d}\boldsymbol{\xi}}\right) \Delta I(\boldsymbol{x}')\,\mathrm{d}\ell(\boldsymbol{x}')}_{\text{boundary}}\right],$$

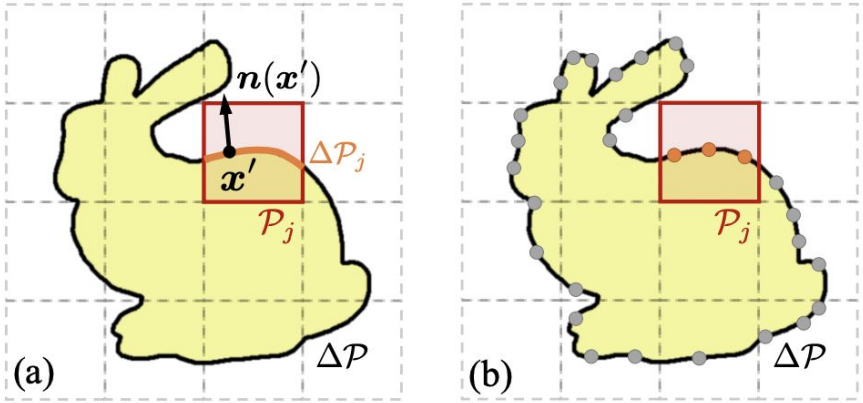
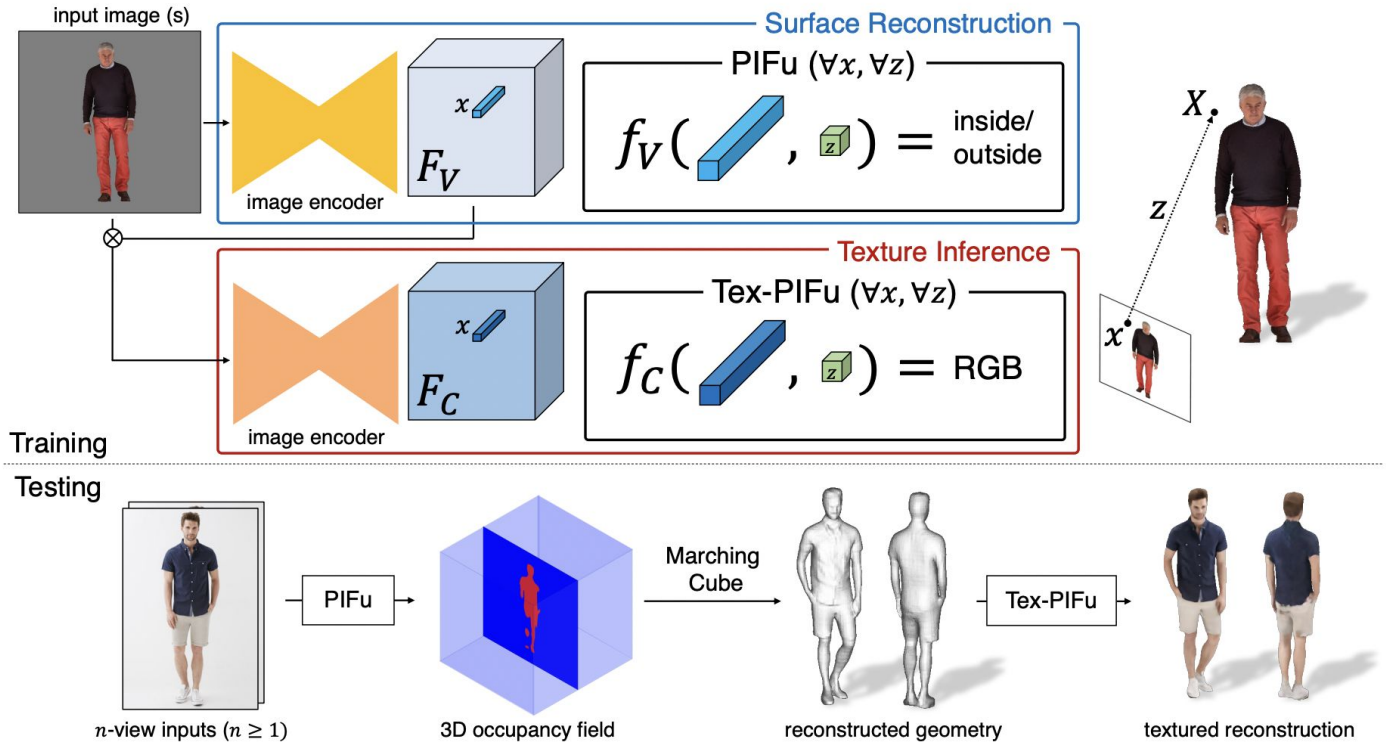



**Figure 2:** ***Differentiable rendering:*** *(a) To properly differentiate the intensity $I_j$ of a pixel $\mathcal{P}_j$ (illustrated as red squares) with respect to object geometry, a boundary integral over $\Delta\mathcal{P}_j$ (illustrated as the orange curve) needs to be calculated. (b) We perform Monte Carlo edge sampling [LADL18, ZWZ*19] by (i) sampling points (illustrated as small discs) from pre-computed discontinuity curves $\Delta\mathcal{P}$, and (ii) accumulating their contributions in the corresponding pixels (e.g., the orange samples contribute to the pixel $\mathcal{P}_j$).*

Luan et al

# Single-shot reconstruction: PiFu




https://arxiv.org/abs/1905.05172

# Going generative...

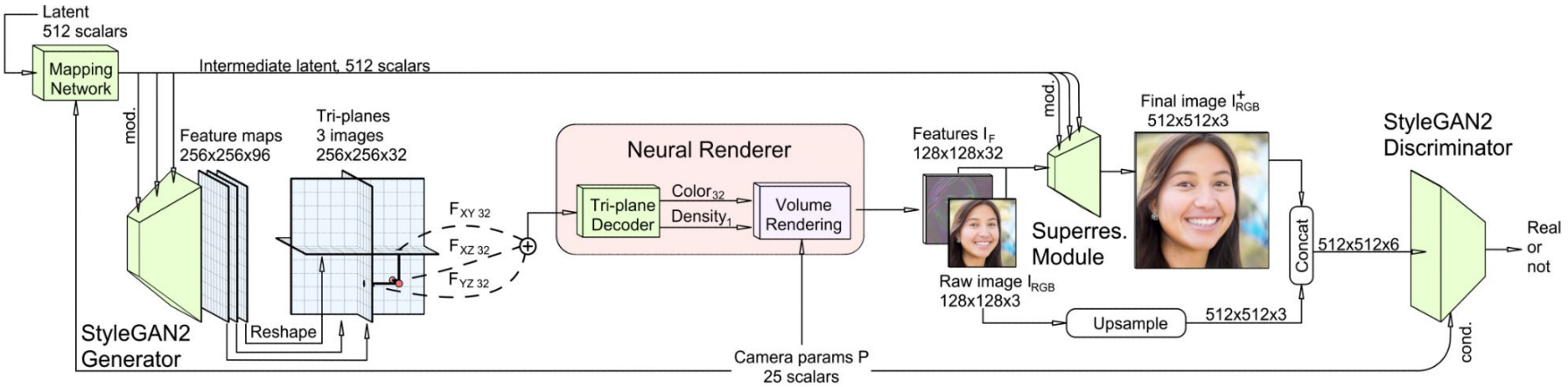
# 3D-aware generation

**Goal:** generate objects, support 3D manipulation

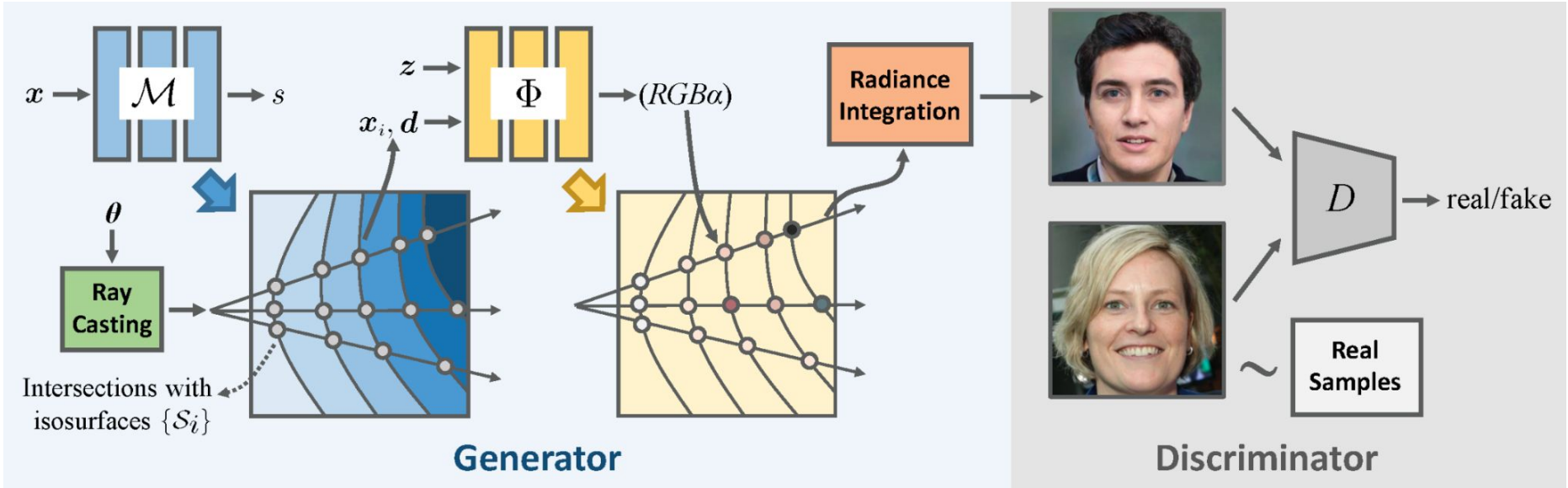
**Methods:**
– StyleNeRF
– EG3D
– GRAM
and many more…

**Data:** ~100k photos

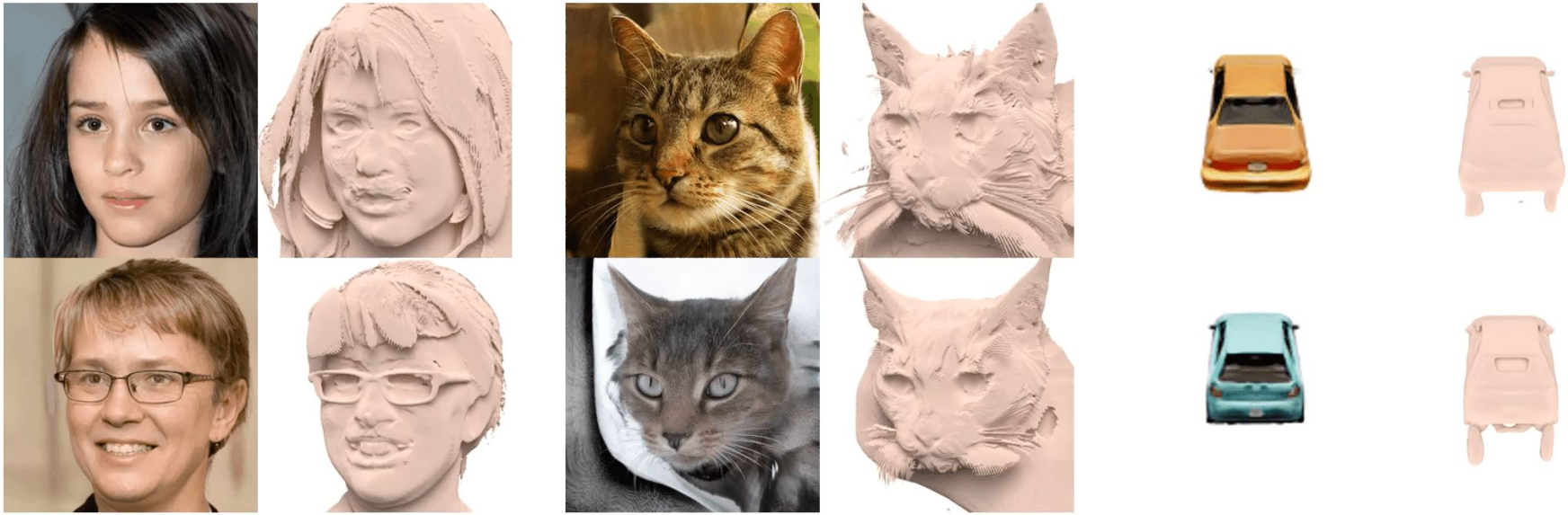
# 3D-aware generation: EG3D



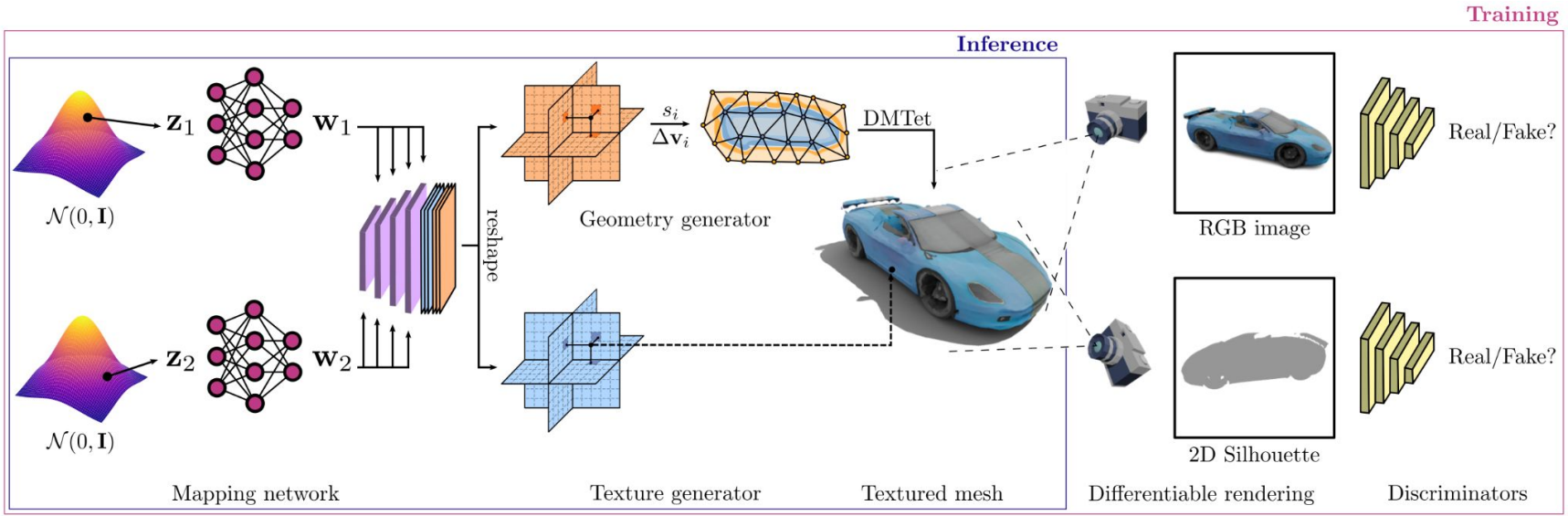

# 3D-aware generation: GRAM



# 3D-aware generation: GRAM



# 3D-aware generation: Get3D



# 3D-aware generation: Get3D

*CLIP conditioning*

**Мем.**

Ты

Кокои

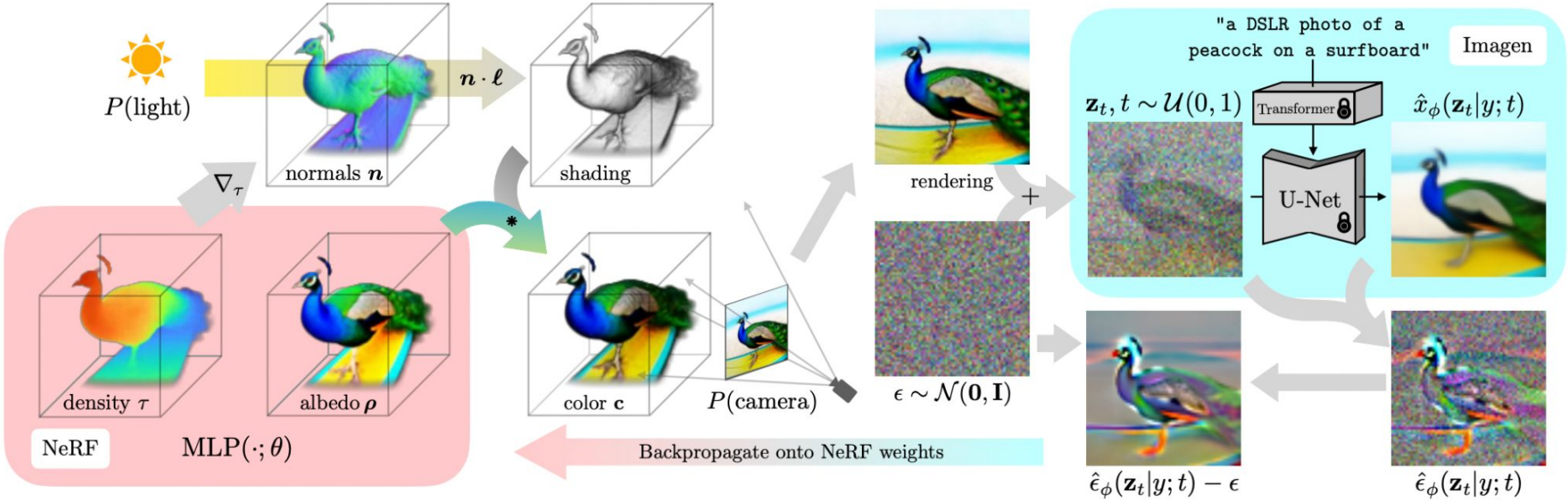
сегодн

я

тигр

¿¿

# 3D-aware generation: DreamFusion



# 3D-aware generation: DreamFusion

# Further reading

- 3D human pose: [1]
- Body mesh: [2], [3]
- Clothes transfer: PiFu, canonical pose transfer
- NeRF papers: NeuS, NeRF-w, NeRF++, NeRF--, NeuralMVS, MipNeRF, NeRD, instant-ngp … (ask me for more)
- Some stuff on 3D object detection, segmentation and depth maps

*the results are processed frame by frame.