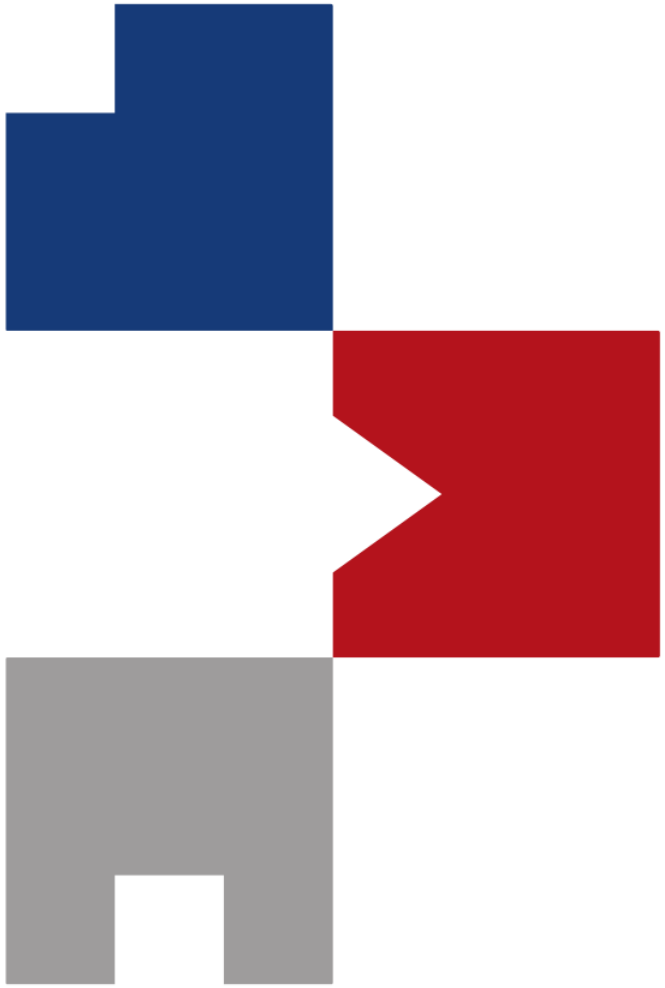
Slides and example codes are available:
https://github.com/mint-lab/cv_tutorial



3D Vision

Sunglok Choi, Assistant Professor, Ph.D.
Computer Science and Engineering Department, SEOULTECH
sunglok@seoultech.ac.kr | https://mint-lab.github.io/

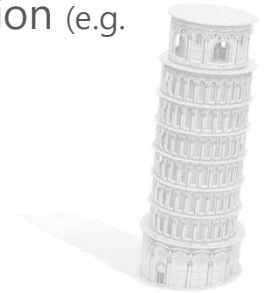
What is Computer Vision?

Computer Vision

What is it?

- Label (e.g. Tower of Pisa)
- Shape (e.g. )

Where am I?

- Place (e.g. Piazza del Duomo, Pisa, Italy)
- Location (e.g. )

(84, 10, 18) [m]

What is 3D Vision?

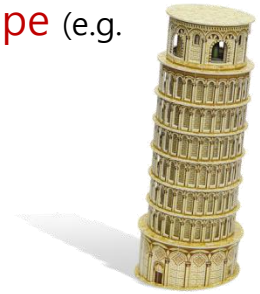
Visual Geometry

Multiple View Geometry

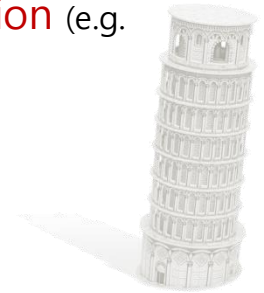
Geometric Vision

Computer Vision

What is it?

- Label (e.g. Tower of Pisa)
- Shape (e.g. )

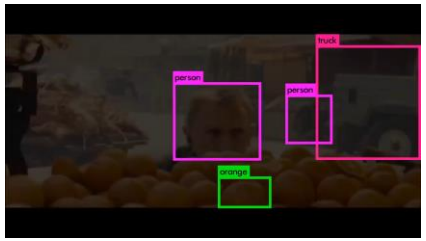
Where am I?

- Place (e.g. Piazza del Duomo, Pisa, Italy)
- Location (e.g. )

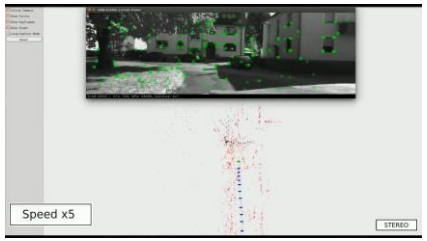
(84, 10, 18) [m]

Recognition Problems **v.s.** **Reconstruction Problems**

Stanford CS231n:
[CNN for Visual Recognition]



YOLO v2 (2016)

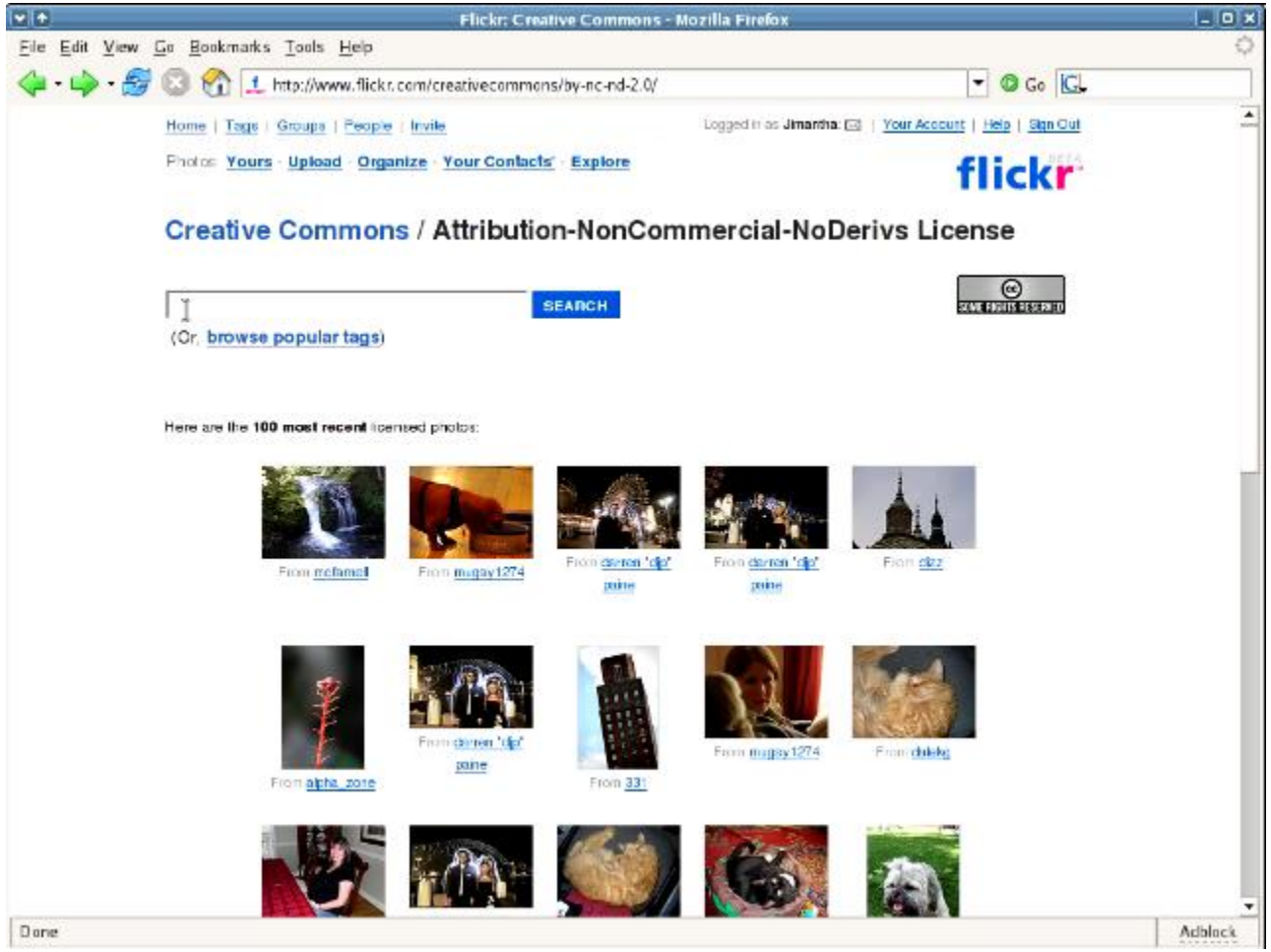


ORB-SLAM2 (2016)

Stanford CS231A:
[Computer Vision,
From 3D Reconstruction to Recognition]

Applications) Photo Browsing

- Photo Tourism (2006)

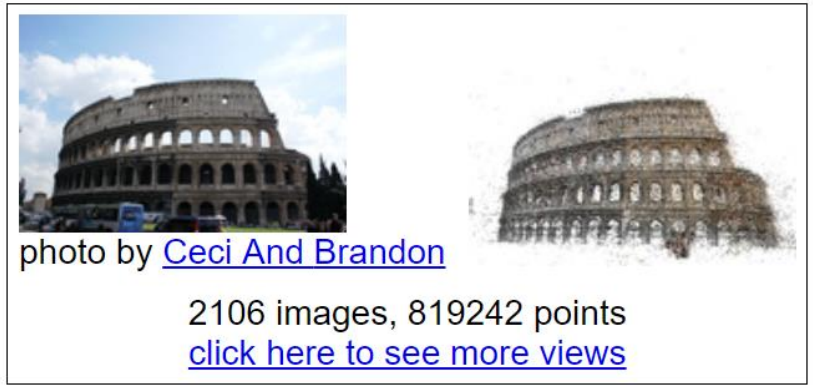


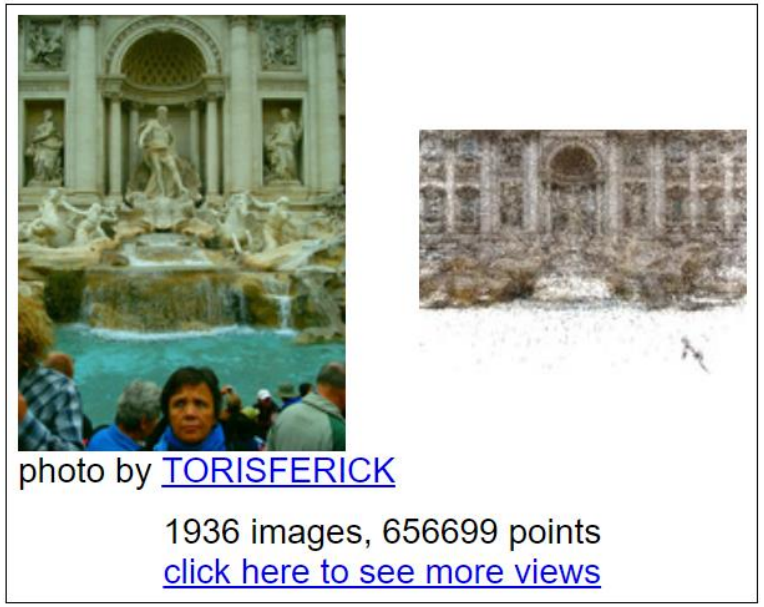
YouTube

Reference: Snavely et al., **Photo Tourism: Exploring Photo Collections in 3D**, SIGGRAPH, 2006

Applications) 3D Reconstruction

- Building Rome in a Day (2009)



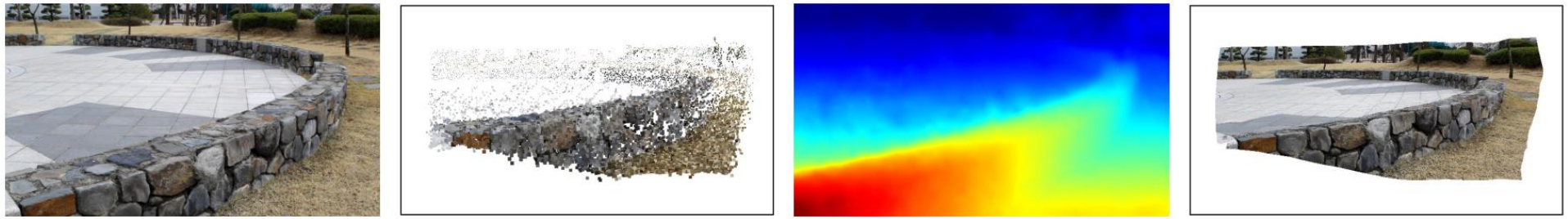


YouTube

Reference: Agarwal et al., **Building Rome in a Day**, ICCV, 2009

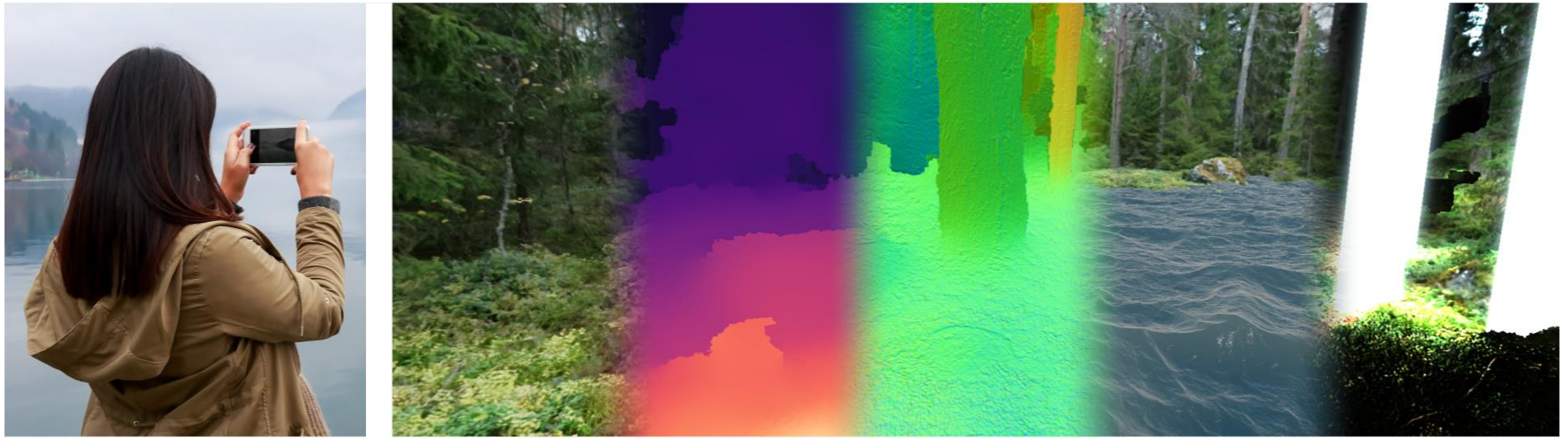
Applications) Depth Estimation from Cellular Phones

- Structure from Small Motion (SfSM; 2015)



(a) Reference images (b) SfSM results (c) Depth maps (d) Our 3D meshes

- Casual 3D Photography (2017)

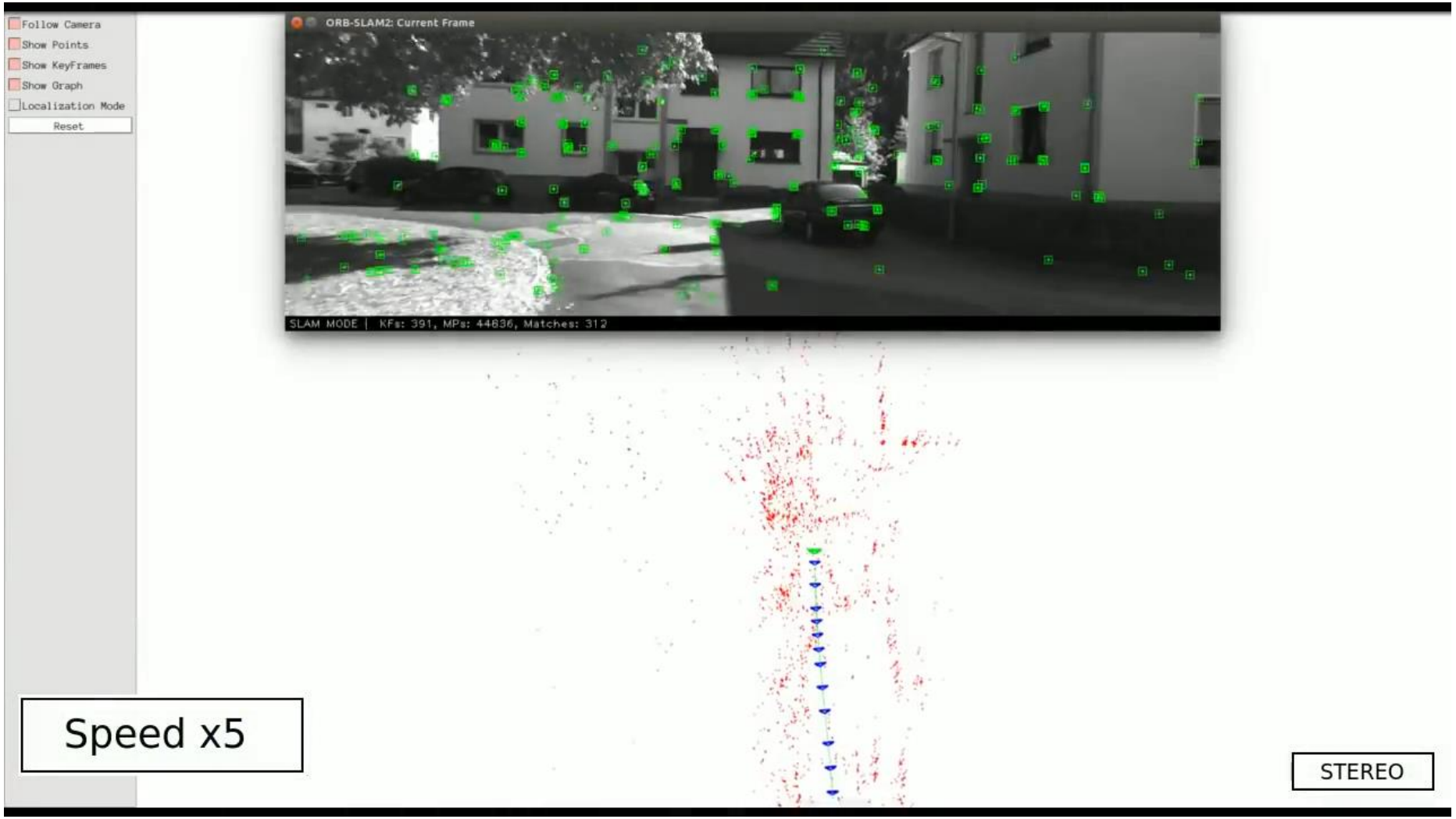


Casual 3D photo capture

Reference: Im et al., **High Quality Structure from Small Motion for Rolling Shutter Cameras**, ICCV, 2015
Reference: Hedman et al., **Casual 3D Photography**, SIGGRAPH Asia, 2017

Applications) Real-time Visual SLAM

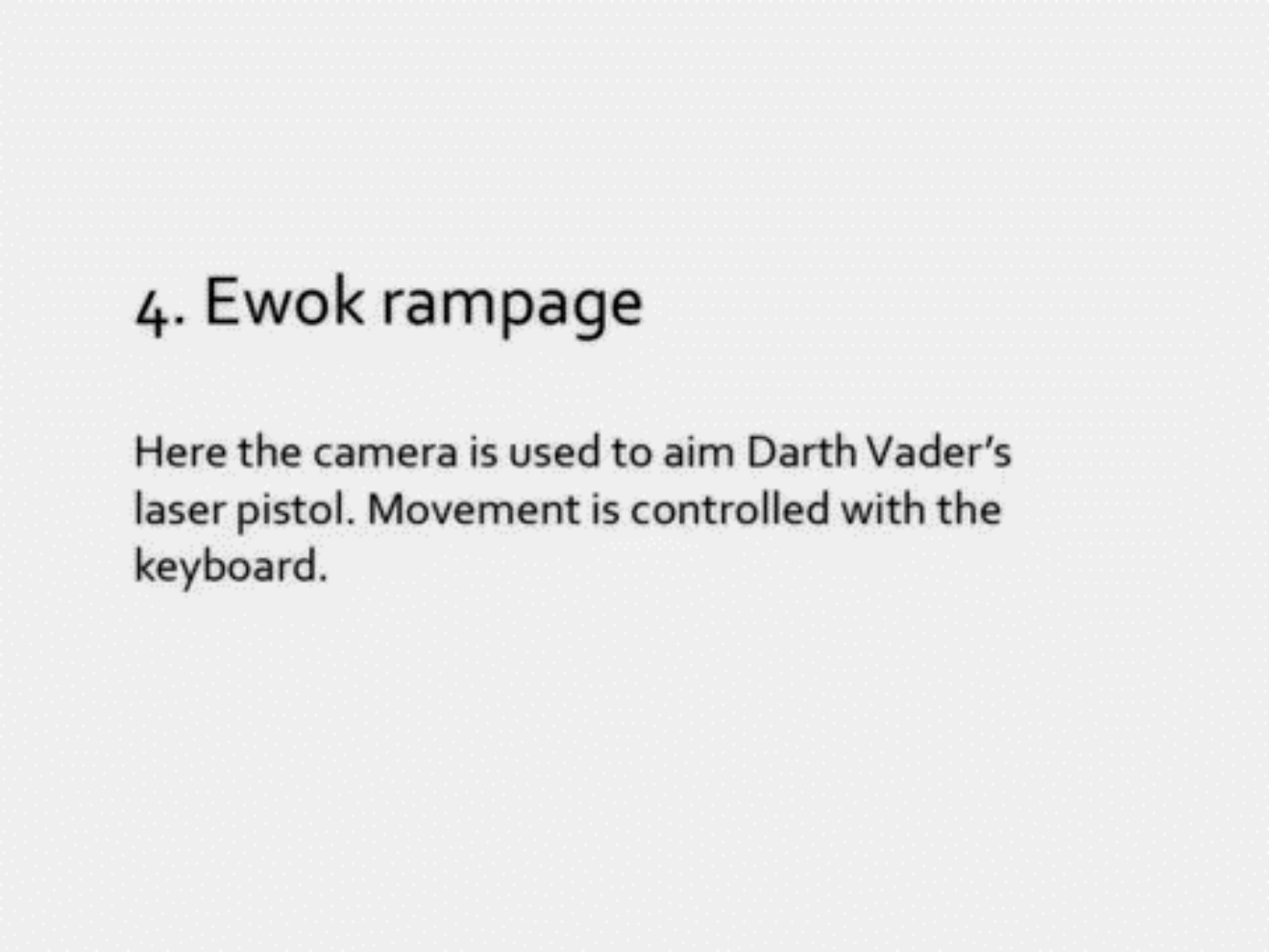
- ORB-SLAM (2014)



Reference: Mur-Artal et al., **ORB-SLAM: A Versatile and Accurate Monocular SLAM System**, T-RO, 2015

Applications) Augmented Reality

- PTAM: Parallel Tracking and Mapping (2007)



Reference: Klein and Murray, **Parallel Tracking and Mapping for Small AR Workspaces**, ISMAR, 2007

Applications) Virtual Reality

- Oculus Quest (2019)

Image: TechSpot

Applications) Mixed Reality

- Microsoft Hololens 2 (2019)
  - Head tracking: 4 x visible light cameras

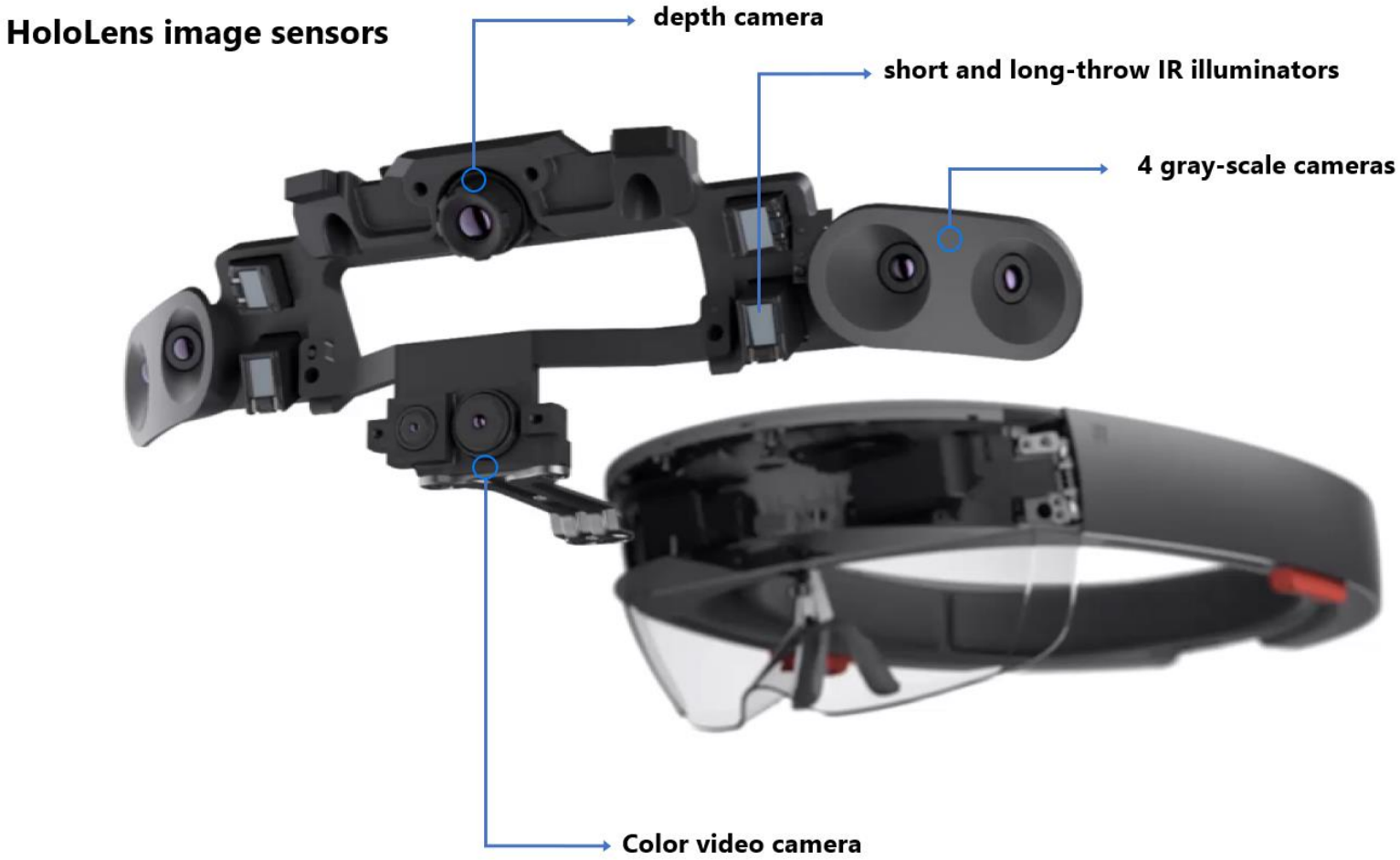


Image: SlashGear

Table of Contents

Structure-from-Motion

- **Structure-from-Motion** (shortly SfM)
  - Unknown: 3D *structures* + each camera's relative pose (roughly $3n + 6m$ DOF)
  - Given: 2D images from different viewpoints (a.k.a. *motion*; but unknown)
  - Solution: Feature extraction and matching + bundle adjustment (shortly BA)

- **Structure-from-Motion tools**
  - Bundler, COLMAP, MVE, Theia, openMVG, OpenSfM, Toy SfM / VisualSFM (GUI, binary only)

Image: Bundler

Structure-from-Motion

- Example) **Structure-from-Motion (batch version)**
  - 0) Load images and extract features
  - 0) Match features and find good matches (which has enough inliers)

Structure-from-Motion

- Example) **Structure-from-Motion (batch version)**

  0) Load images and extract features

  0) Match features and find good matches (which has enough inliers)

  1) Initialize camera parameters

$$\mathrm{K} = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix}, \mathrm{R}_i = \mathrm{I}_{3\times 3}, \text{ and } \mathbf{t}_i = [0,0,0]^\intercal$$

  2) Initialize 3D points and build a visibility graph

$$\mathrm{X}_i = [0,0,2]^\intercal$$

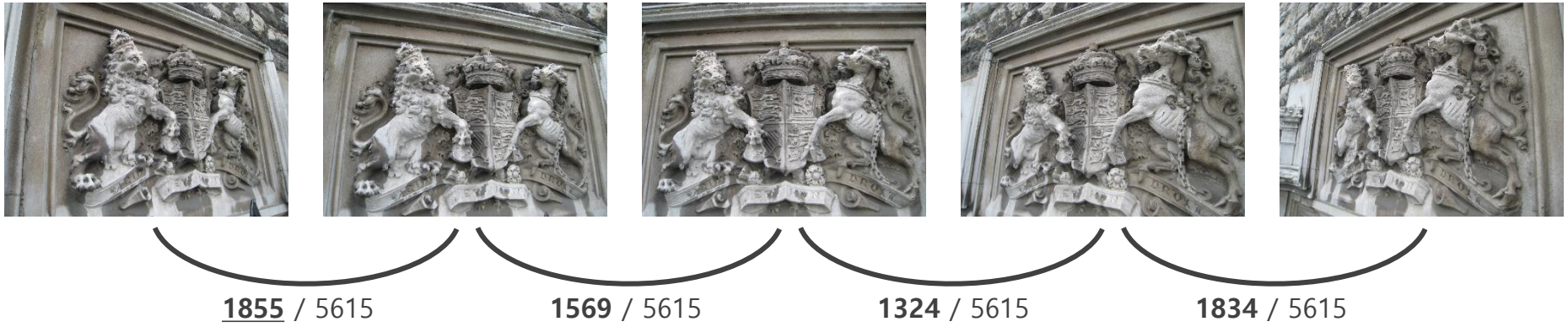
  3) Optimize camera poses and 3D points together **(BA)**

Structure-from-Motion

- Example) **Structure-from-Motion (incremental version)**

  0) Load images and extract features

  0) Build a viewing graph (well-matched pairs) while finding inliers



  1) Select the best pair

  2) Estimate relative pose from the best two views **(epipolar geometry)**

  3) Reconstruct 3D points of the best two views **(triangulation)**

  4) Select the next best image to add

     Separate points into known and unknown for PnP (known) and triangulation (unknown)

  5) Estimate relative pose of the next best view using (known) 3D points **(PnP)**

  6) Reconstruct newly observed (unknown) 3D points **(triangulation)**

  7) Optimize camera poses and 3D points together **(BA)**

Until there is no image left

COLMAP (2016)

- **The state-of-the-art incremental SfM and more**
  - The author won the PAMI Mark Everingham Prize (2020) for COLMAP SfM and MVS.

- **References**
  - Code repositories
    - COLMAP (BSD license): Homepage, Documentation, Github
    - Note) COLMAP binary files are also available for rapid deployment.
  - Papers
    - Schonberger et al., *Structure-from-Motion Revisited*, CVPR, 2016 PDF DOI
    - Schonberger et al., *Pixelwise View Selection for Unstructured Multi-view Stereo*, ECCV, 2016 PDF DOI
    - Schonberger et al., *A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval*, ACCV, 2016 PDF DOI

COLMAP (2016)

- **The state-of-the-art incremental SfM and more**
  - The best performance in the view of **accuracy** and **time/space complexity** [Bianco et al., 2018]

Table 3. SfM camera pose evaluation results. N_c is the percentage of used cameras. $\bar{x}$ is the mean distance from ground truth of reconstructed camera positions and s_x its standard deviation. $\bar{r}$ is the mean rotation difference from ground truth of reconstructed camera orientations and s_r its standard deviation. n.a. means measure not available.

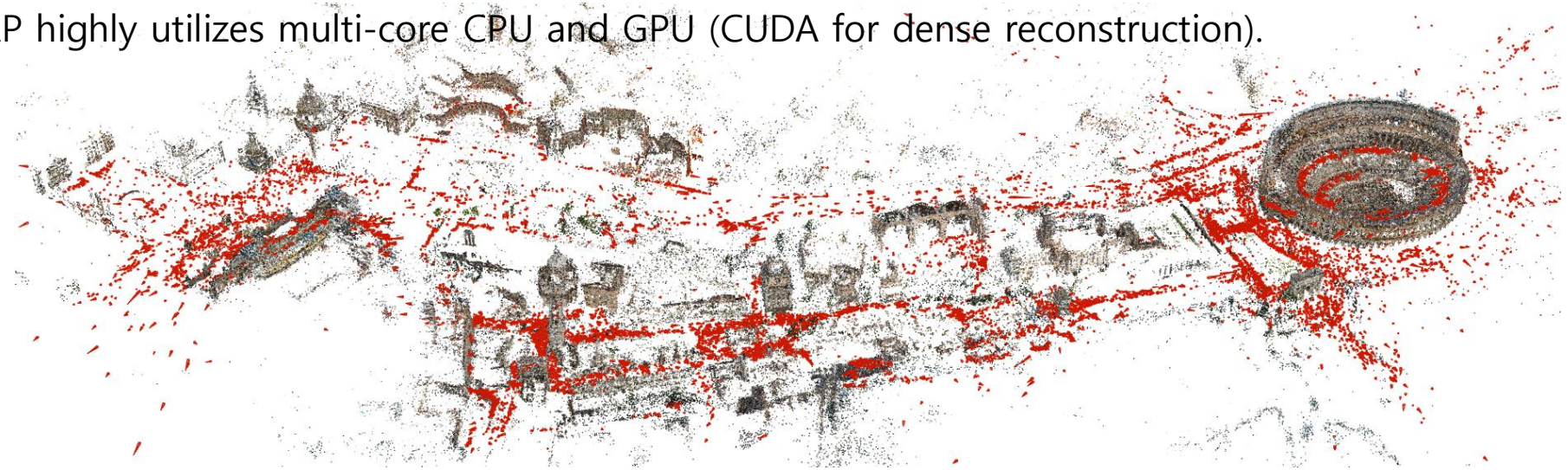
Model	COLMAP					OpenMVG					Theia					VisualSFM				
	N_c	$\bar{x}$ [m]	s_x [m]	$\bar{r}$ [°]	s_r [m]	N_c	$\bar{x}$ [m]	s_x [m]	$\bar{r}$ [°]	s_r [m]	N_c	$\bar{x}$ [m]	s_x [m]	$\bar{r}$ [°]	s_r [m]	N_c	$\bar{x}$ [m]	s_x [m]	$\bar{r}$ [°]	s_r [m]
Statue	100	0.08	0.01	0.04	0.05	100	0.27	0.03	0.47	0.22	100	1.86	0.09	0.45	0.22	100	1.45	0.91	3.55	2.88
E. Vase	100	0.01	0.01	0.51	0.05	83	0.78	1.62	32.19	64.89	100	0.13	0.07	0.91	0.35	94	0.15	0.14	4.91	5.07
Bicycle	88	0.04	0.02	0.25	0.19	94	0.60	1.09	7.00	12.53	37	0.60	0.03	1.10	0.31	47	0.27	0.14	1.32	1.03
Hydrant	82	2.63	2.09	72.28	64.98	3	–	–	–	–	6	3.49	0.29	174.27	0.51	80	2.43	1.76	66.29	58.90
Jeep	63	0.04	0.02	0.26	0.11	92	0.24	1.32	4.80	26.33	95	0.43	0.42	1.33	5.67	83	1.02	2.79	9.68	22.84
Ignatius	100	n.a.	n.a.	n.a.	n.a.	100	n.a.	n.a.	n.a.	n.a.	100	n.a.	n.a.	n.a.	n.a.	100	n.a.	n.a.	n.a.	n.a.

Table 5. Pipelines execution times in seconds and memory usage in MB.

Model	COLMAP				OpenMVG				Theia				VisualSFM			
	SfM		MVS		SfM		MVS		SfM		MVS		SfM		MVS	
	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*	t [s]	*RAM*
Statue	59	897	86	1062	43	1359	115	1300	196	1984	98	1249	86	1406	144	1452
Empire Vase	53	897	154	2101	28	628	130	1734	129	1988	134	1926	62	1226	159	2095
Bicycle	98	896	117	1356	57	1467	146	1720	63	1722	58	548	64	1226	68	641
Hydrant	19	894	55	793	16	1547	–	–	17	2048	3	1249	36	997	56	1452
Jeep	38	897	121	1812	69	1550	275	3209	213	2083	280	3293	109	1406	254	3078
Ignatius	1225	1825	430	5082	401	1555	494	5926	992	2588	484	5626	1639	2381	345	4742

COLMAP (2016)

- **The state-of-the-art incremental SfM and <u>more</u>**
  - COLMAP includes not only **sparse reconstruction**, but also **dense reconstruction** and **mesh reconstruction**.
  - COLMAP highly utilizes multi-core CPU and GPU (CUDA for dense reconstruction).

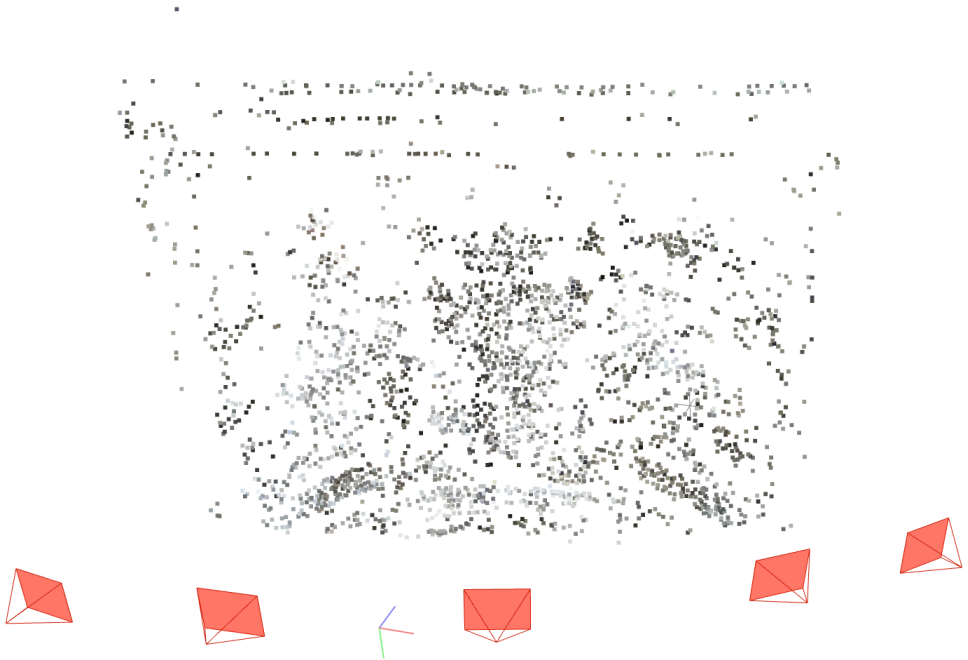


A sparse model of *central Rome* using 21K photos produced by COLMAP's SfM pipeline

Dense models of several landmarks produced by COLMAP's MVS pipeline

COLMAP (2016)

Sparse Reconstruction (SfM)



# of points: 2,889

Dense Reconstruction (MVS)

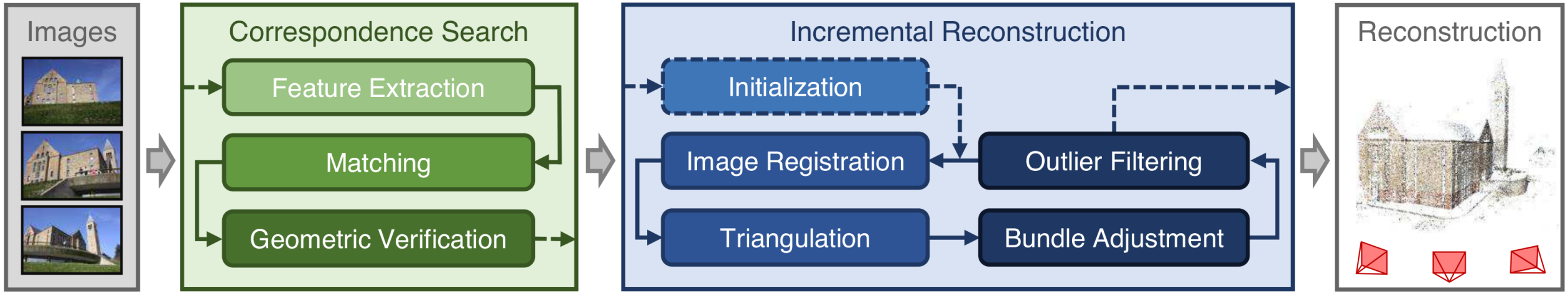
# of points: 336,223

Mesh Reconstruction

Viewer: [CloudCompare]

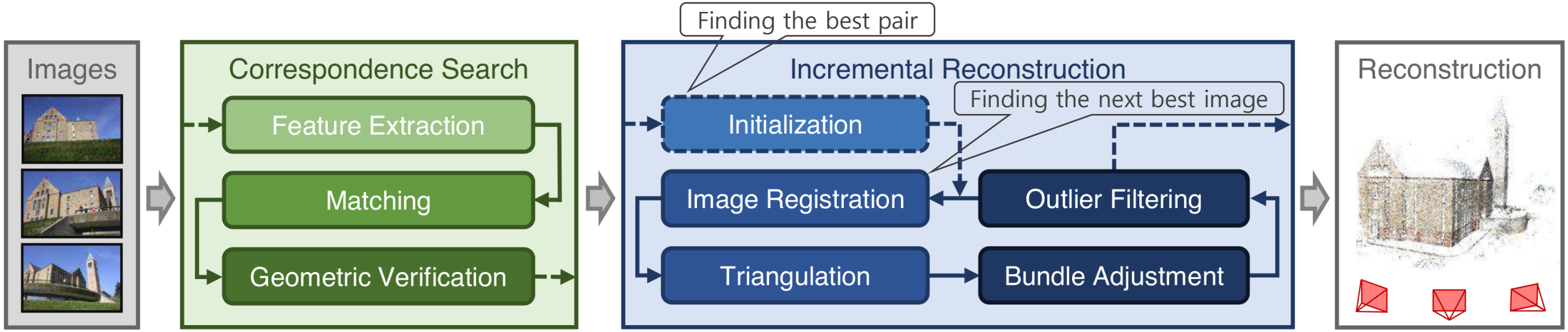
# of vertices: 102,694
# of faces: 205,633

Review) Incremental Structure-from-Motion



- **Correspondence Search**
  - **Feature extraction**
    - The features should be invariant under radiometric and geometric changes.
    - **SIFT** and its derivatives, learned features vs. binary features (better efficiency with reduced robustness)
  - **Matching** → Feature correspondence
    - Exhaustive search (time complexity: $O(N_I^2\, N_F^2)$) vs. more efficient search (for scalability)
  - **Geometric verification** → Scene graph (node: images, edge: feature/geometric relationship)
    - Geometric relations: H (for pure rotation and a planar scene) vs. F and E (for moving camera in a general scene)
    - Issues) Outliers: **RANSAC** / Model selection: **GRIC**, QDEGSAC

Review) Incremental Structure-from-Motion



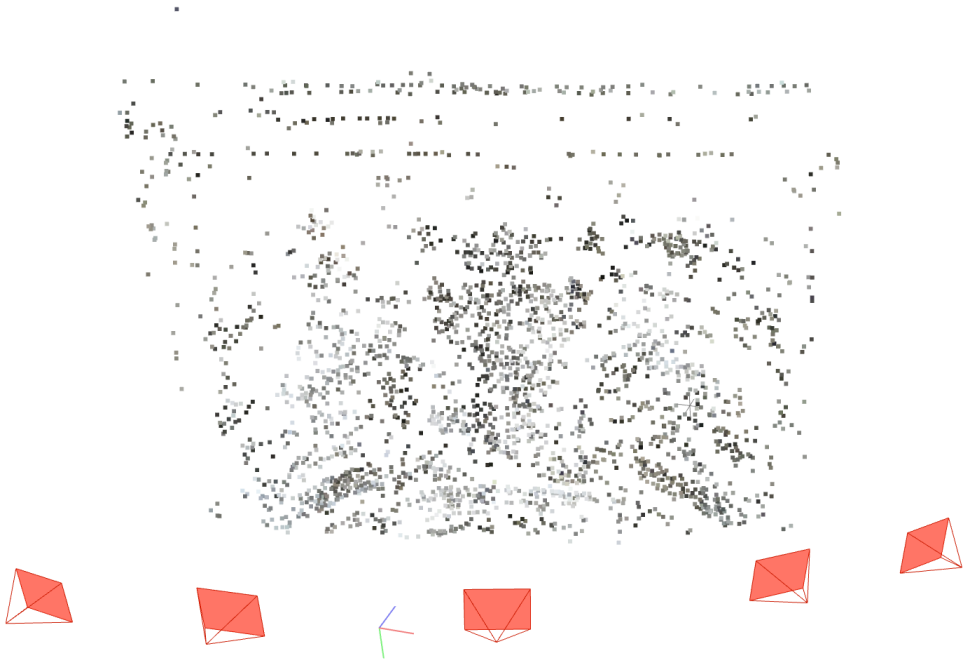
- **Incremental Reconstruction**
  - **Initialization**: Two-view reconstruction from the best pair of images [critical]
  - **Image Registration**: PnP (+ intrinsic parameters) with 2D-3D correspondences and RANSAC
  - **Triangulation**: Adding more 2D-3D correspondences
  - **Bundle Adjustment (BA)**
    - Motivation) SfM usually drifts quickly without additional refinement to image registration and triangulation.
    - The joint non-linear ~~minimization~~ refinement of camera parameters $\mathbf{P}_c$ and 3D points $\mathbf{X}_k$

      $$E = \sum_j \rho_j \left(\left\| \pi(\mathbf{X}_k; \mathbf{P}_c) - \mathbf{x}_j \right\|_2^2 \right)$$ with a loss function ρ_j (against outliers)
    - Choices) Solvers: LM, …, exact vs. inexact methods / Error functions: Direct vs. sparse direct vs. **indirect**

COLMAP (2016): Getting Started with COLMAP GUI

- Example) COLMAP with the relief dataset

Sparse Reconstruction (SfM)



# of points: 2,889

Dense Reconstruction (MVS)

# of points: 336,223

Mesh Reconstruction

Viewer: CloudCompare

# of vertices: 102,694
# of faces: 205,633

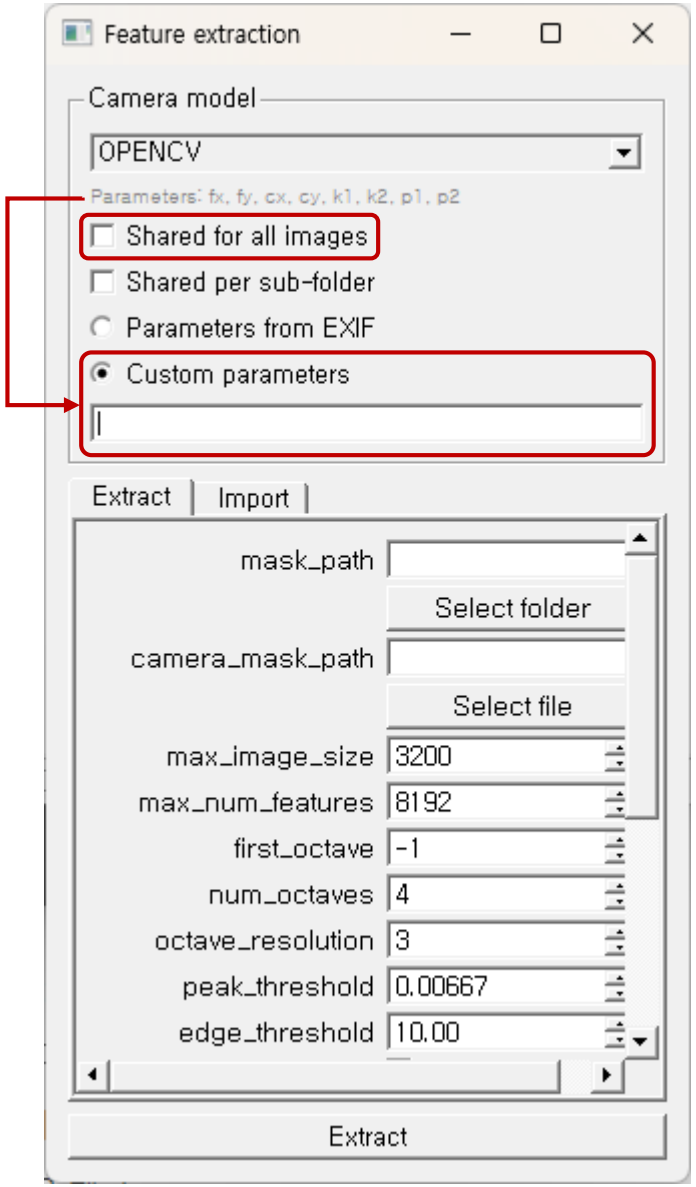
COLMAP (2016): Getting Started with COLMAP GUI

- **Download COLMAP Binaries**
  - Download the pre-release version available on the official website.
    - Note) Please download the version labeled with `-cuda` if you want to use an NVIDIA GPU for dense reconstruction.
  - For more efficient feature matching, download the vocabulary trees available on the COLMAP website.
- **Running COLMAP GUI @ Windows**
  - Execute `COLMAP.bat` file
  - Note) Please refer the official document for detailed mouse/keyboard usages.
- **COLMAP SfM/MVS Step-by-Step**
  1. Create a Project and Load Input Images
  2. Feature Extraction
  3. Feature Matching
  4. Sparse Reconstruction (SfM)
  5. Dense Reconstruction (MVS)
  6. Mesh Reconstruction

COLMAP (2016): Getting Started with COLMAP GUI

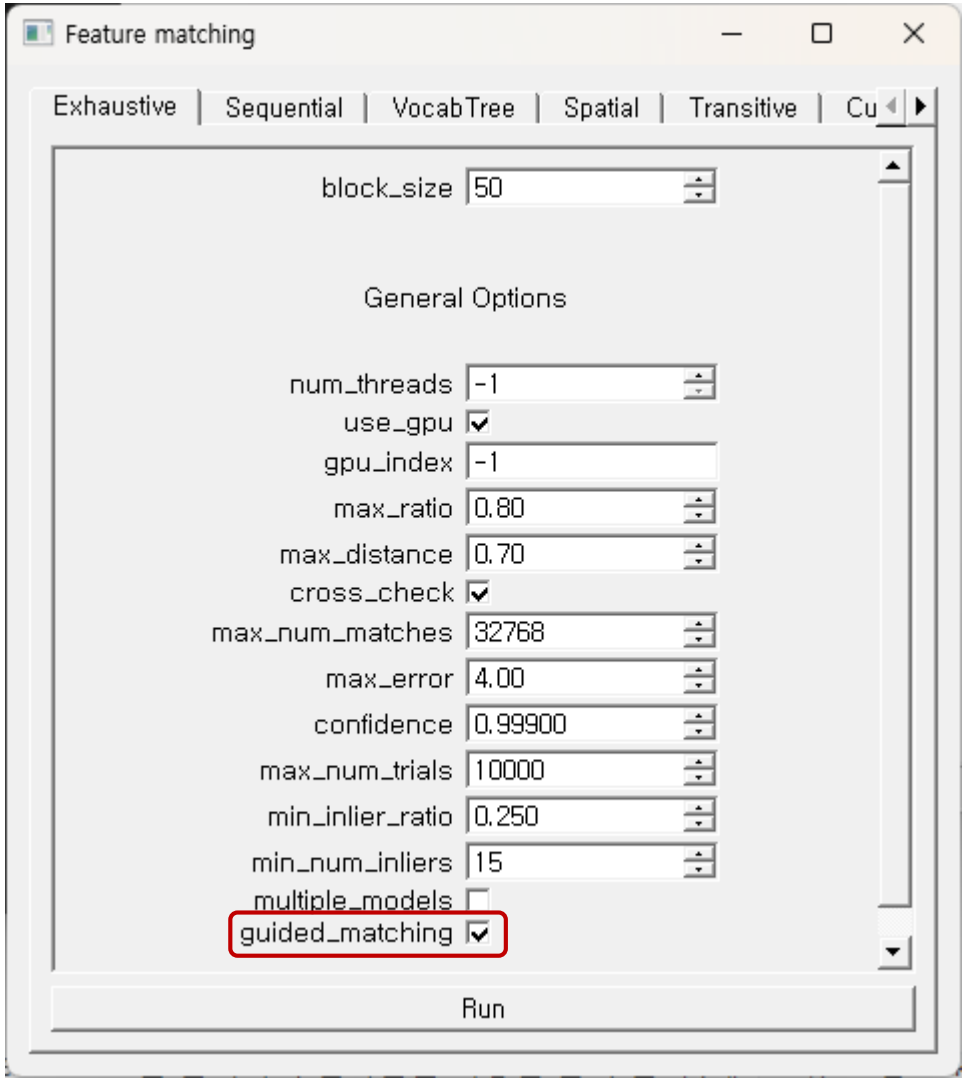
- **COLMAP SfM/MVS Step-by-Step (Cont'd)**

1. **Create a Project and Load Input Images**: Menu > `File` > `New project`
   - **For Database**: Click `New` to set the name of the database where the SfM results will be stored.
     - e.g. `C:\your_workspace\database.db`
   - **For Image**: Click `Select` to set the directory containing the images for SfM.
     - e.g. `C:\your_workspace\images`
   - Click the `Save` button to save the settings.
2. **Feature Extraction**: Menu > `Processing` > `Feature extraction`
   - Tip) If all images have the same camera parameters, check the `Shared for all images` option.
   - Tip) If you already know the internal parameters of the camera, select the `Custom parameters` option and fill the values (the order of values is shown in gray under the selected `Camera model`).



COLMAP (2016): Getting Started with COLMAP GUI

- **COLMAP SfM/MVS Step-by-Step (Cont'd)**
  1. Create a Project and Load Input Images
  2. Feature Extraction
  3. **Feature Matching**: Menu > `Processing` > `Feature matching`
     - Various matching methods are available.
       - By default, use `Exhaustive` to match all combinations.
       - For ordered images like SLAM data, use `Sequential` for faster matching (vocabulary tree setup needed for loop closing).
     - Tip) If your matching is not successful or SfM does not converge, try to check the `guided_matching` option.



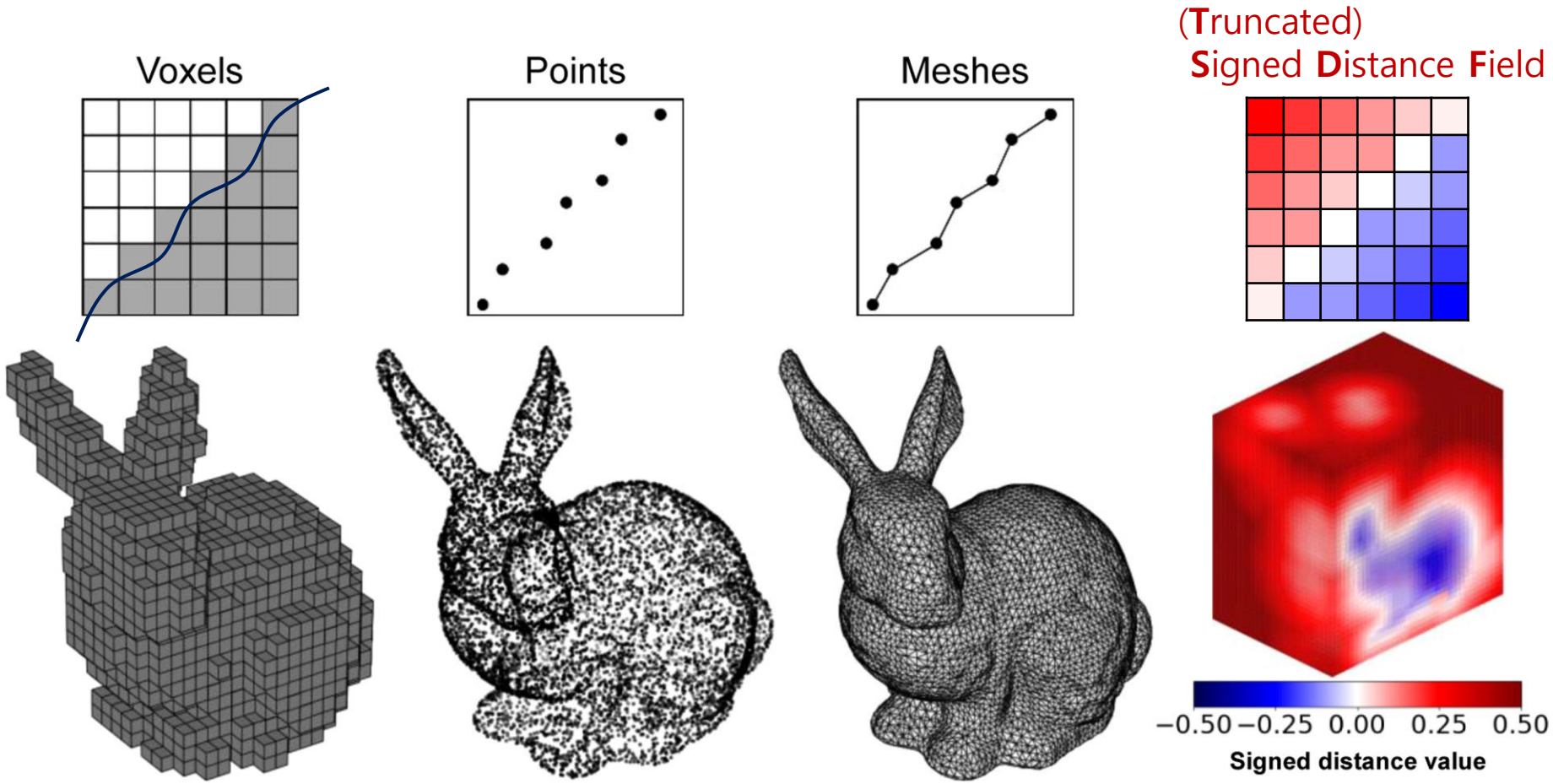
COLMAP (2016): Getting Started with COLMAP GUI

- **COLMAP SfM/MVS Step-by-Step (Cont'd)**
  1. Create a Project and Load Input Images
  2. Feature Extraction
  3. Feature Matching
  4. **Sparse Reconstruction (SfM)**: Menu > `Reconstruction` > `Start reconstruction`.
  5. **Dense Reconstruction (MVS)**: Menu > `Reconstruction` > `Dense reconstruction`.
     - Set the directory to save the results by clicking `Select`.
       - e.g. `C:\your_workspace\dense`
     - Follow the next steps, `Undistort` → `Stereo` → `Fusion`.
  6. **Mesh Reconstruction**
     - Follow the more steps, `Poisson` → `Delaunay`, after running the above dense reconstruction.

Table of Contents

Motivation) 3D Shape Representations

- Explicit vs. Implicit representations



Why not a neural network?

Image: Attar et al., **"Implicit Neural Representations of Sheet Stamping Geometries with Small-Scale Features"**, EAAI, 2023 DOI

NeRF (Neural Radiance Field; 2020)

- **NeRF** is a **multi-layer perceptron** for rendering an image from a new viewpoint.
- Network: **11 fully-connected (shortly FC) layers**
  - Input: Spatial location $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (d_x, d_y, d_z)$ on a unit sphere
  - Output: RGB color (RGB) and density (σ)

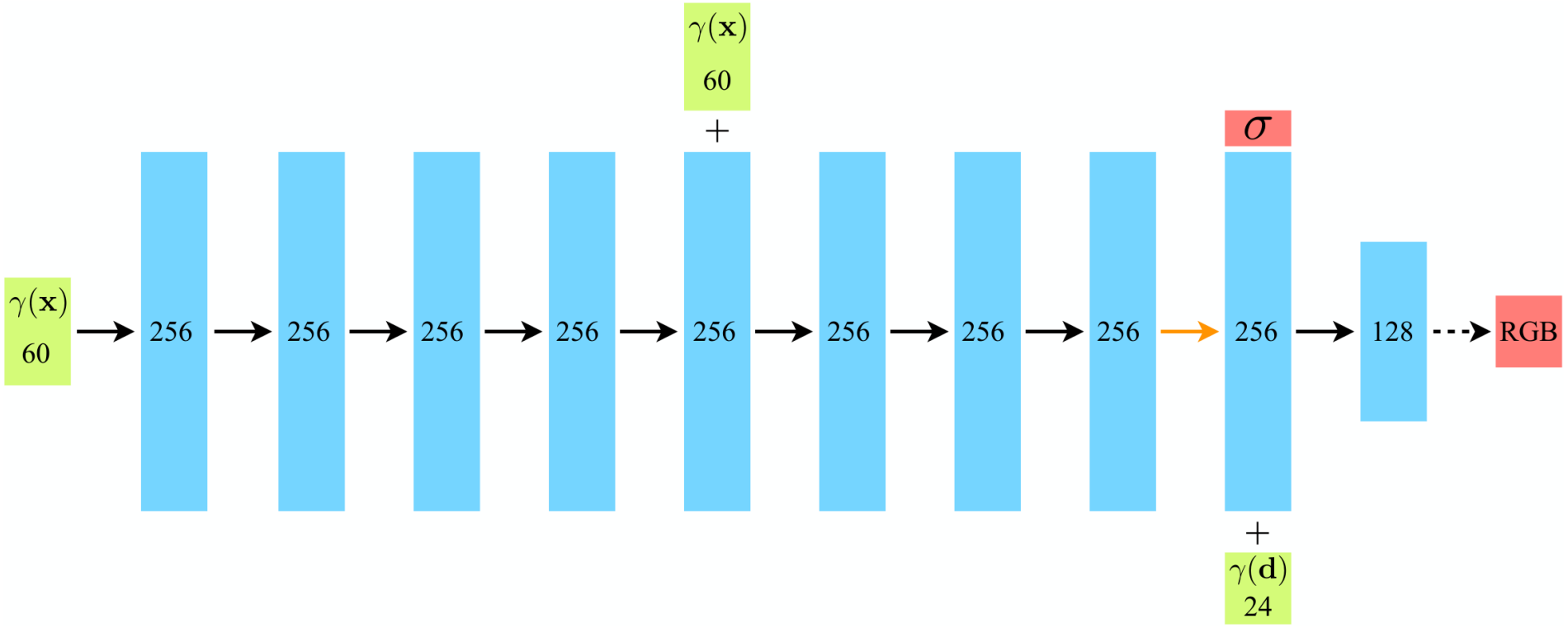
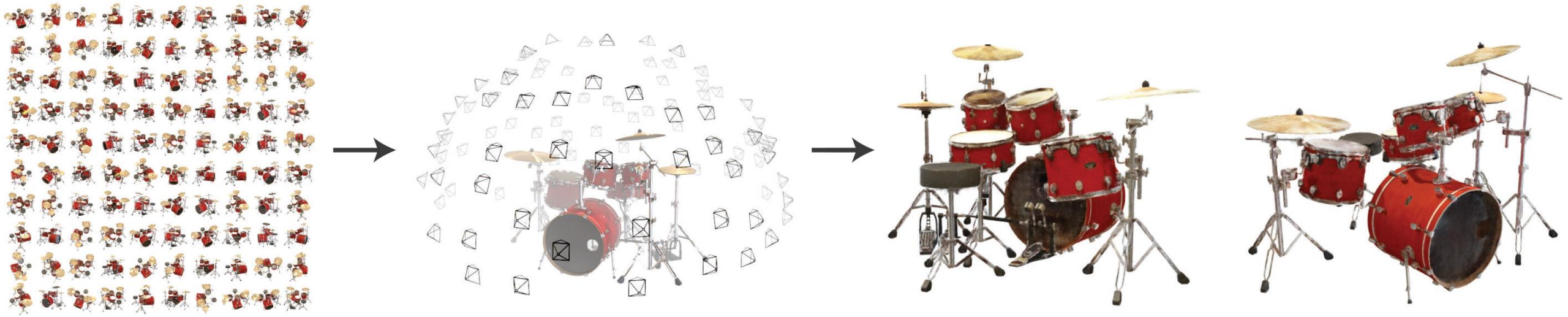


Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

implicitly representing a 3D scene.

- **NeRF** is a multi-layer perceptron for ~~**rendering an image from a new viewpoint**~~.
- **Training**: Learning the 3D scene
  - Input: Images with their 3D viewpoints (R_j, $\mathbf{t}_j$)
    - Note) 3D viewpoints can be retrieved by SfM (e.g. COLMAP).
- **Inference**: Synthesizing a 2D image with a *new* viewpoint
  - Input: A new camera viewpoint (R_n, $\mathbf{t}_n$)



Input images | Training the NeRF network | Rendering new views

Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

- **NeRF** is a multi-layer perceptron for **rendering an image from a new viewpoint**.
- **Inference**: Synthesizing a 2D image with a *new* viewpoint
  - Input: A new camera viewpoint $(\mathrm{R}_n, \mathbf{t}_n)$

Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

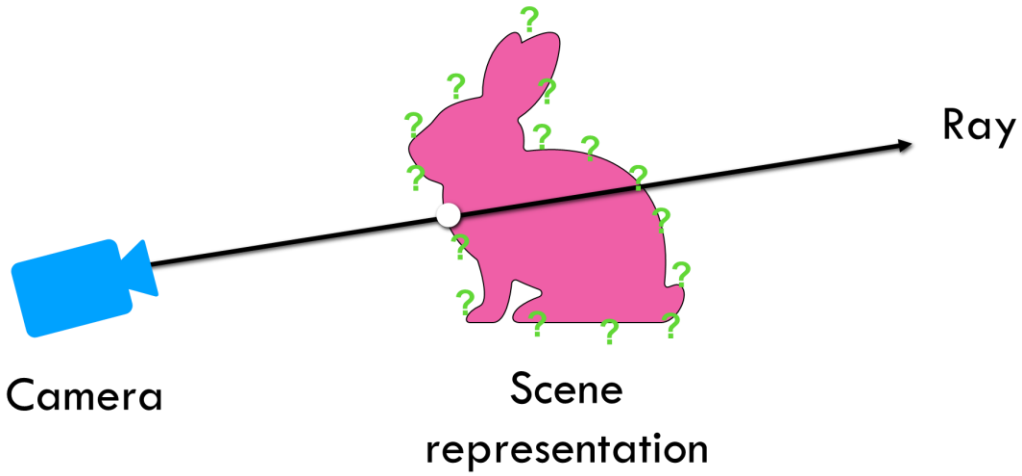
- Key idea: **Neural Volumetric Rendering**
  - + Continuous rendering
  - + Differentiable rendering
  - + Model without concrete ray/surface intersections

Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

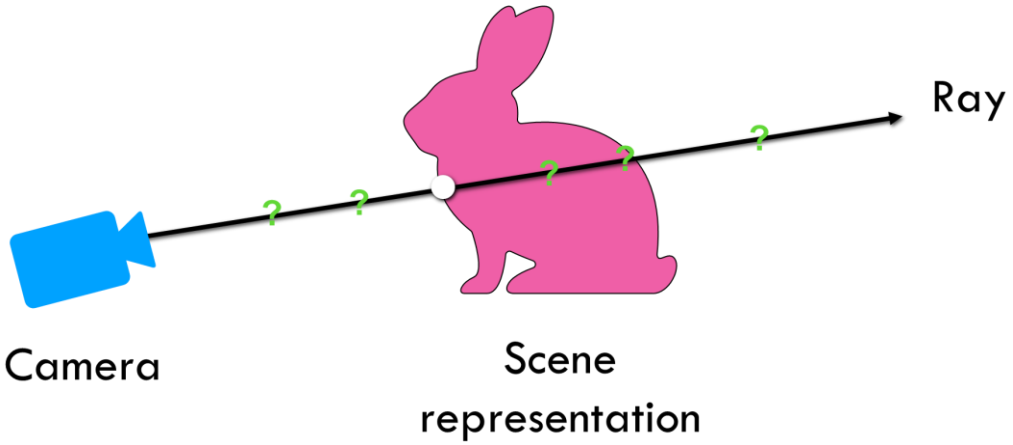
NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**

Surface rendering
(loop over geometry, check for ray hits)



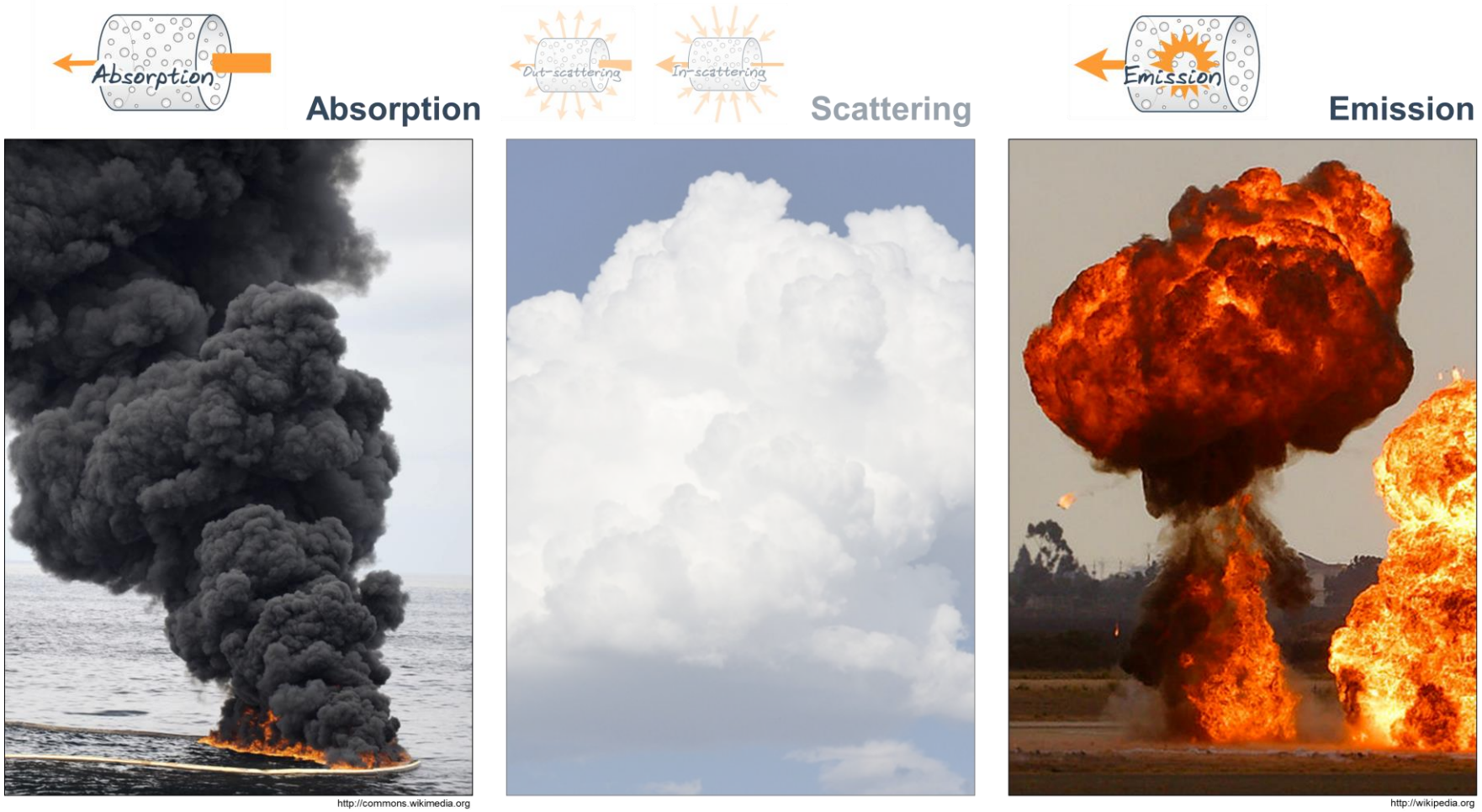
Volume rendering
(loop over ray points, query geometry)



Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

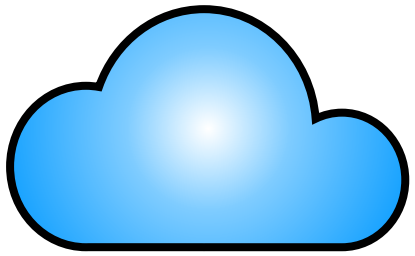
- Key idea: **Neural Volumetric Rendering**
  - Based on the simplified physics (ignoring **scattering**)



Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - The scene is a cloud composed of tiny colored particles.

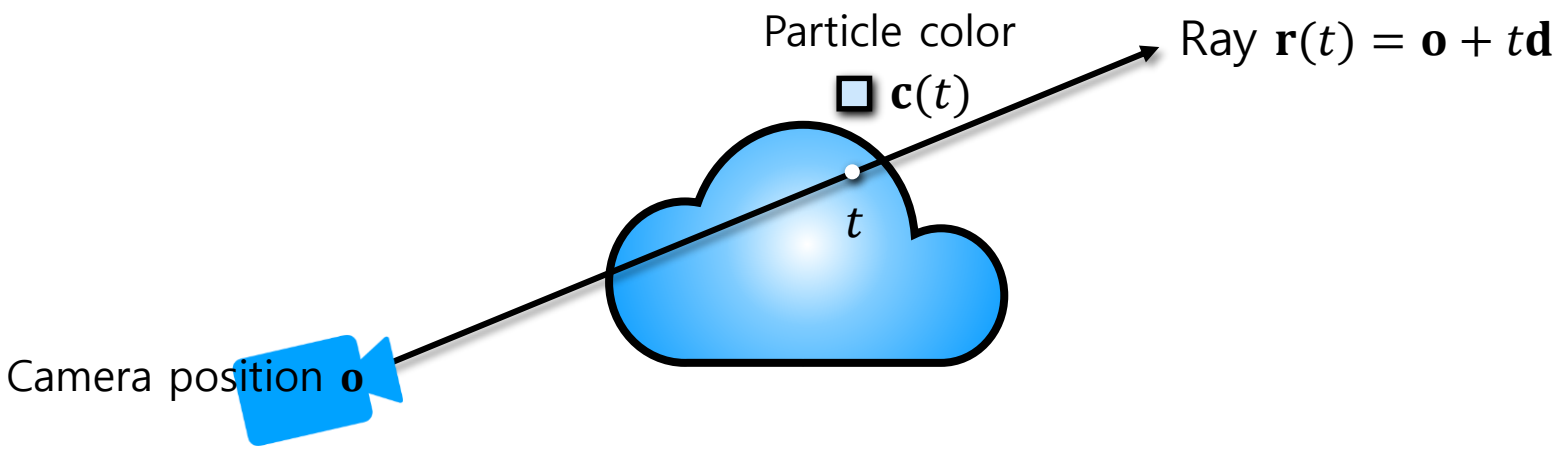


3D volume

Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

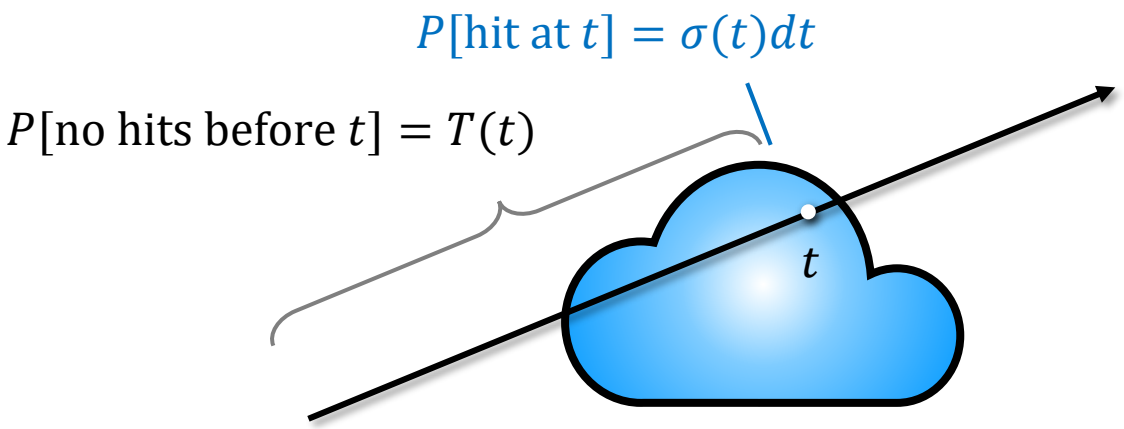
- Key idea: **Neural Volumetric Rendering**
  - If a **ray** (traveling through the scene) hits a **particle** at distance t (along the ray), we can retrieve its color $\mathbf{c}(t)$.



Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - The product of these probabilities tells us how much you see the particles at t:
    - $P[\text{first hit at } t] = P[\text{no hit before } t] \times P[\text{hit at } t] = T(t)\sigma(t)dt$
    - $T(t)$: Transmittance (the probability that the ray doesn't hit any particles earlier)

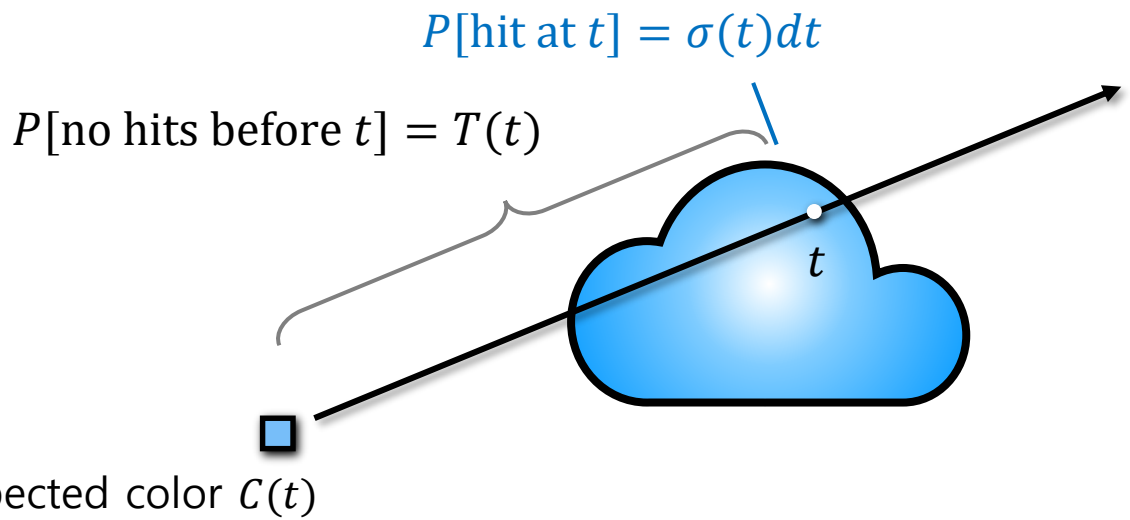


Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - The product of these probabilities tells us how much you see the particles at t:
    - $P[\text{first hit at } t] = P[\text{no hit before } t] \times P[\text{hit at } t] = T(t)\sigma(t)dt$
  - Expected color by a ray $\mathbf{r}$

$$C(\mathbf{r}) = \int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{N} T_i \mathbf{c}_i \left(1 - \exp(-\sigma_i \delta_i)\right) \quad \text{where} \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

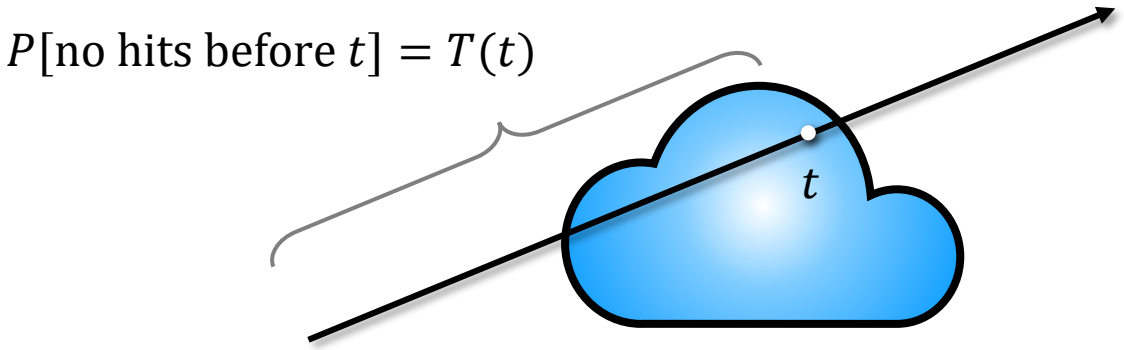


Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - Q) Can we represent $T(t)$ using the (known) density function $\sigma(t)$?
    - A recursive form: $P[\text{no hit before } t+dt] = P[\text{no hit before } t] \times P[\text{no hit at } t]$

$$T(t+dt) = T(t)(1-\sigma(t)dt) \;\rightarrow\; T(t) = \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right)$$



Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - Q) Can we represent $T(t)$ using the (known) density function $\sigma(t)$?
    - A recursive form: $P[\text{no hit before } t+dt] = P[\text{no hit before } t] \times P[\text{no hit at } t]$

$$T(t+dt) = T(t)(1-\sigma(t)dt) \;\rightarrow\; T(t) = \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right)$$

  - Solving the differential equation

$$T(t+dt) = T(t)(1-\sigma(t)dt)$$

$$T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt \quad \text{(Taylor expansion for } T\text{)}$$

$$\frac{T'(t)}{T(t)}dt = -\sigma(t)dt \quad \text{(Rearrange)}$$

$$\log T(t) = -\int_{t_0}^{t}\sigma(s)ds \quad \text{(Integrate from } t_0 \text{ to } t\text{)}$$

$$T(t) = \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right) \quad \text{(Exponentiate)}$$

Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - Q) How can we integrate the color equation?

$$C(\mathbf{r}) = \int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{N} T_i \mathbf{c}_i \left(1 - \exp(-\sigma_i \delta_i)\right)$$

  - Using the quadrature rule (구분구적법 in Korean)

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt \qquad \text{(Quadrature rule)}$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp(-\sigma_i (t - t_i)) dt \qquad \text{(Substitute)}$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \frac{\exp(-\sigma_i (t_{i+1} - t_i)) - 1}{-\sigma_i} \qquad \text{(Integrate)}$$

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i (1 - \exp(-\sigma_i \delta_i)) \qquad \text{(Cancel } \sigma_i \text{ and substitute } \delta_i = t_{i+1} - t_i\text{)}$$

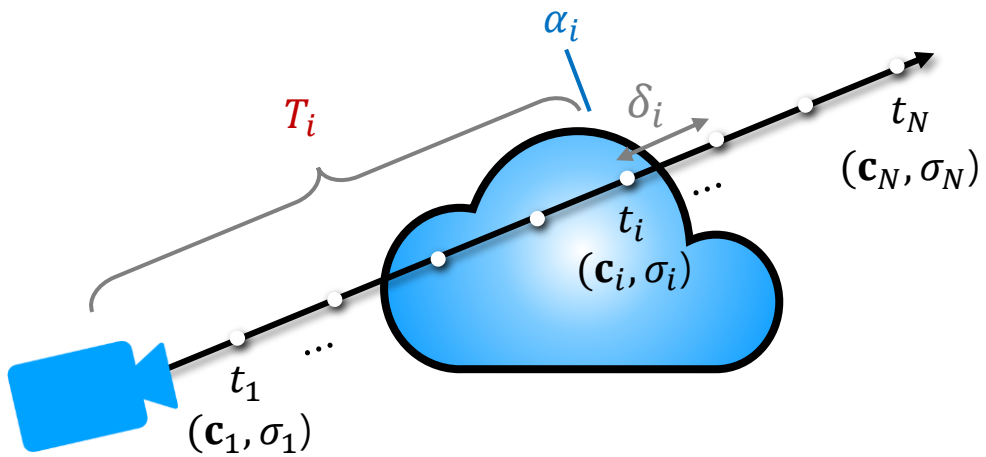
Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Neural Volumetric Rendering**
  - Rendering an image for a ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$

$$C(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i \quad \text{where} \quad \alpha_i = 1 - \exp(-\sigma_i \delta_i) \quad \text{and} \quad T_i = \prod_{j=1}^{i=1} (1 - \alpha_j)$$

  - T_i: How much light is blocked before ray segment i?
  - α_i: How much light is contributed by ray segment i?



Slide: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- **Inference**: Synthesizing a 2D image with a *new* viewpoint
  - Input: A new camera viewpoint (R_n, $\mathbf{t}_n$)
- **Training**: Learning the 3D scene
  - Input: Images with their 3D viewpoints (R_j, $\mathbf{t}_j$)
    - Note) 3D viewpoints can be retrieved by SfM (e.g. COLMAP).
  - Loss function: Rendering loss (MSE) between the *input* and *synthesized* images at each (R_j, $\mathbf{t}_j$)
    - The *synthesized* images are generated by the neural volumetric rendering.

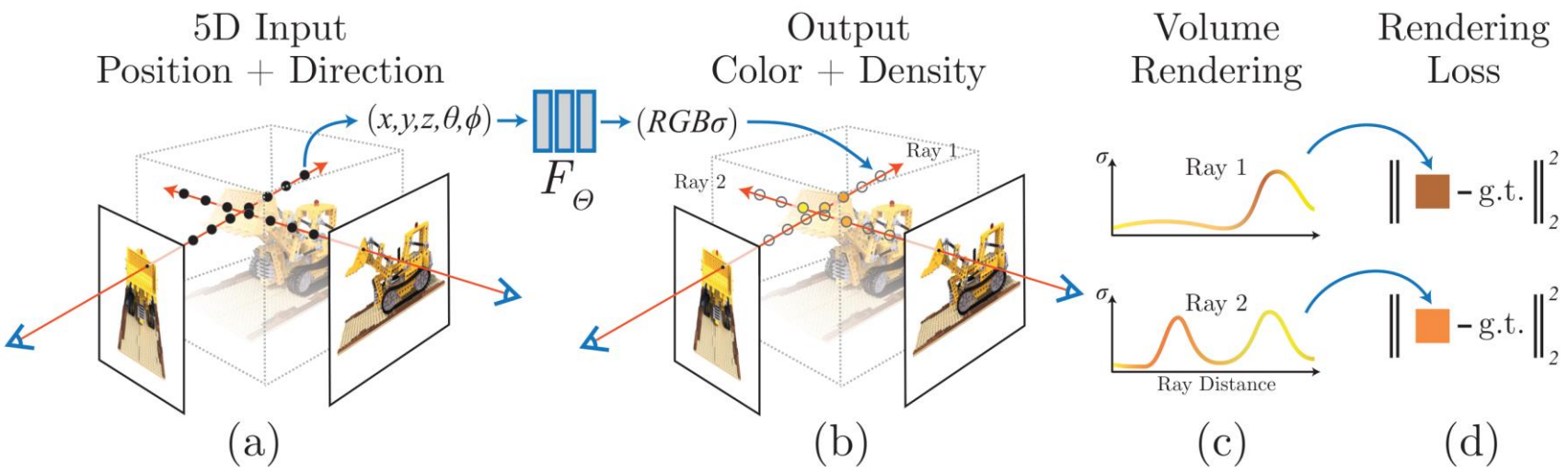


Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

- Network: **11 fully-connected (shortly FC) layers**
  - Input: Spatial location $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (d_x, d_y, d_z)$ on a unit sphere
    - Q) What is a function γ? Why are the input dimensions 60 and 24?
  - Output: RGB color (RGB) and density (σ)

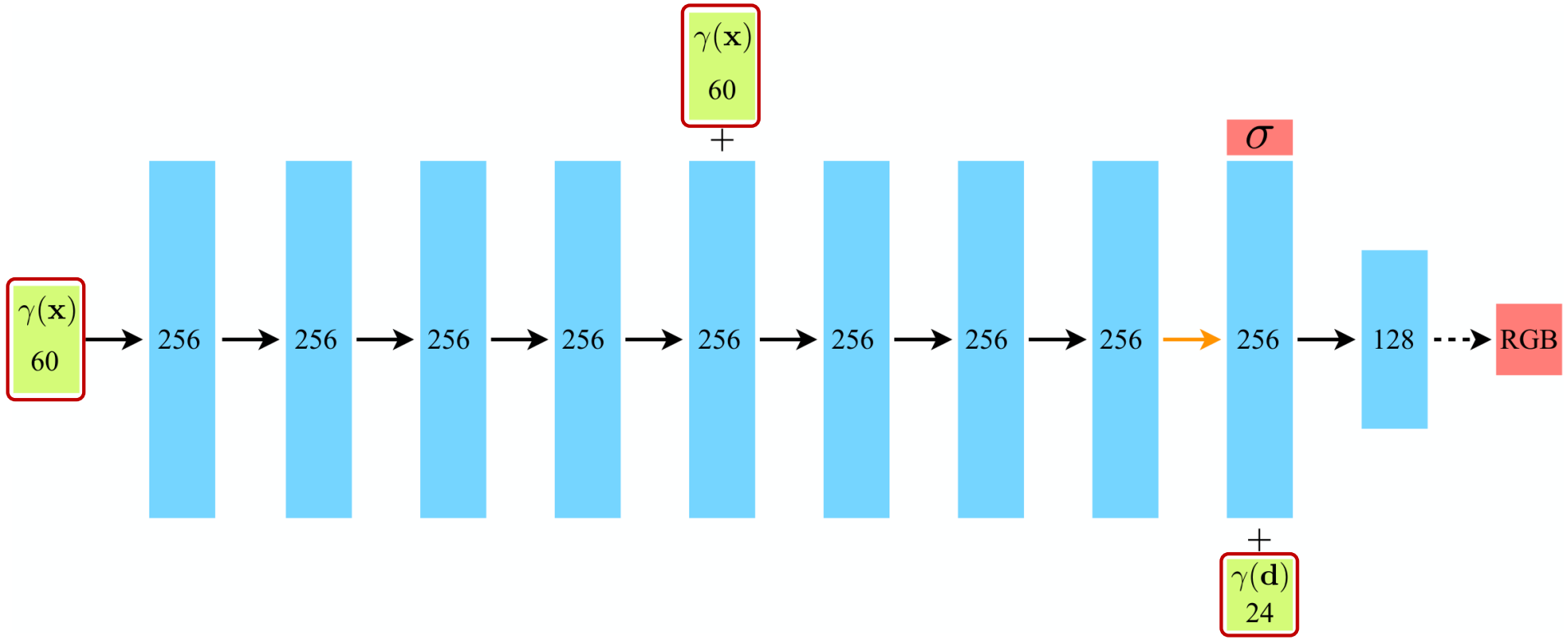


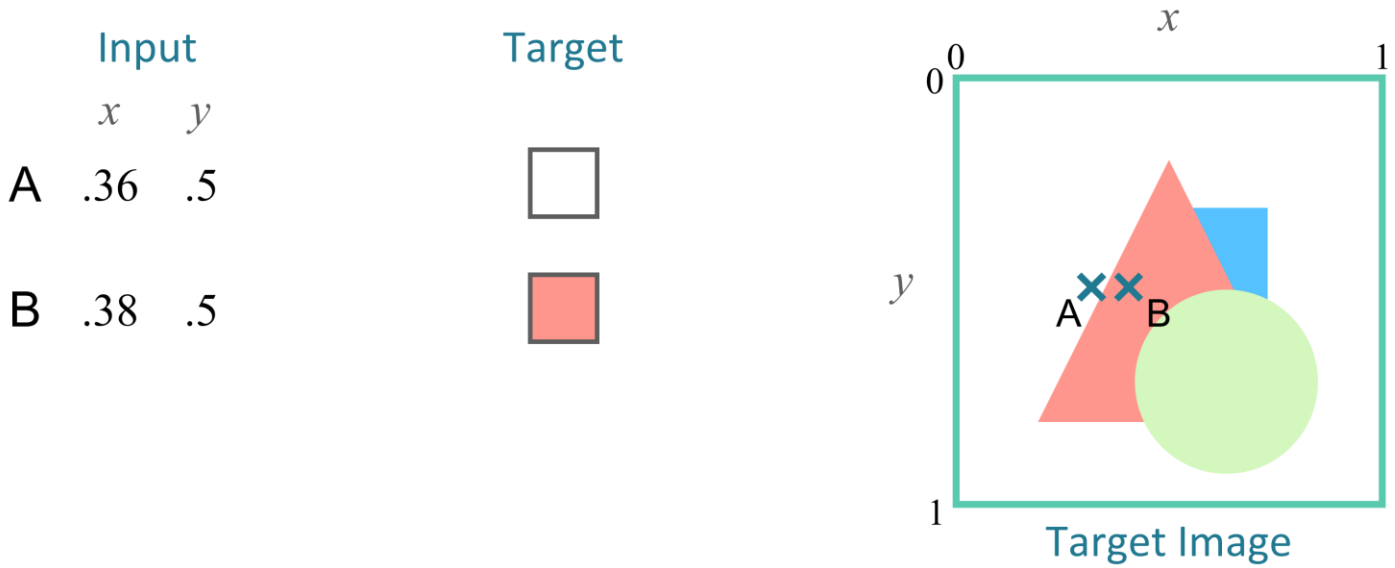
Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

- Key idea: **Positional encoding** transforms a real number $p \rightarrow 2L$-dimensional real numbers.

$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \cdots \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right)$$

  - Q) Why? 3.1415 and 3.1414 may generate similar values.



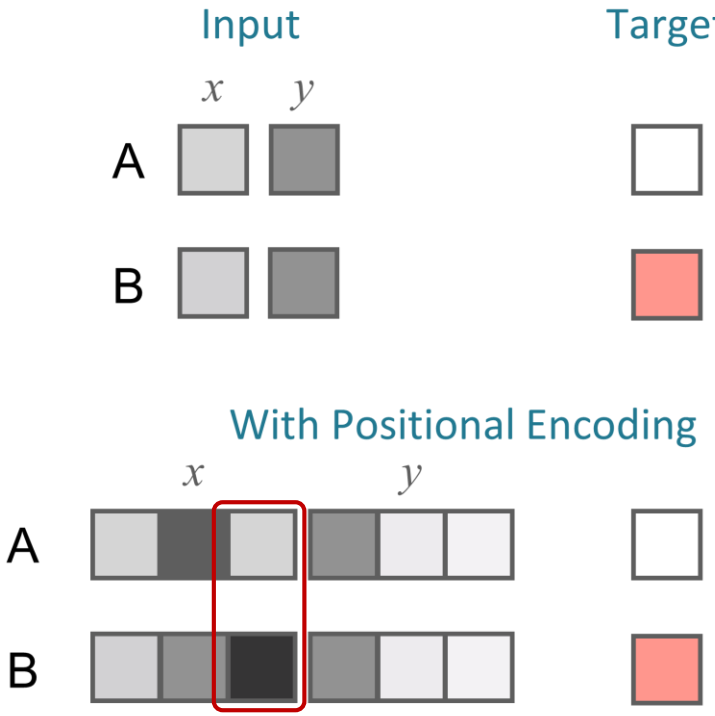
Slide: Matt Tancik, **"Encoding and Representing 3D Volumes"**, NeRF Tutorial, ECCV 2022

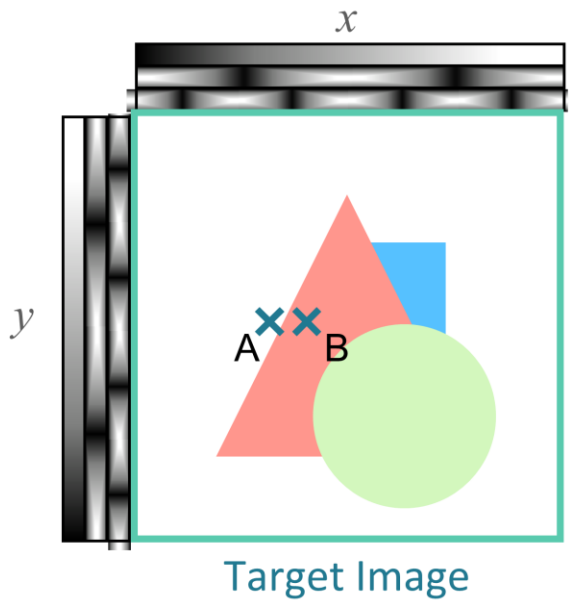
NeRF (Neural Radiance Field; 2020)

- Key idea: **Positional encoding** transforms a real number $p \rightarrow 2L$-dimensional real numbers.

$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \cdots \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right)$$

  - Q) Why? 3.1415 and 3.1414 may generate similar values.





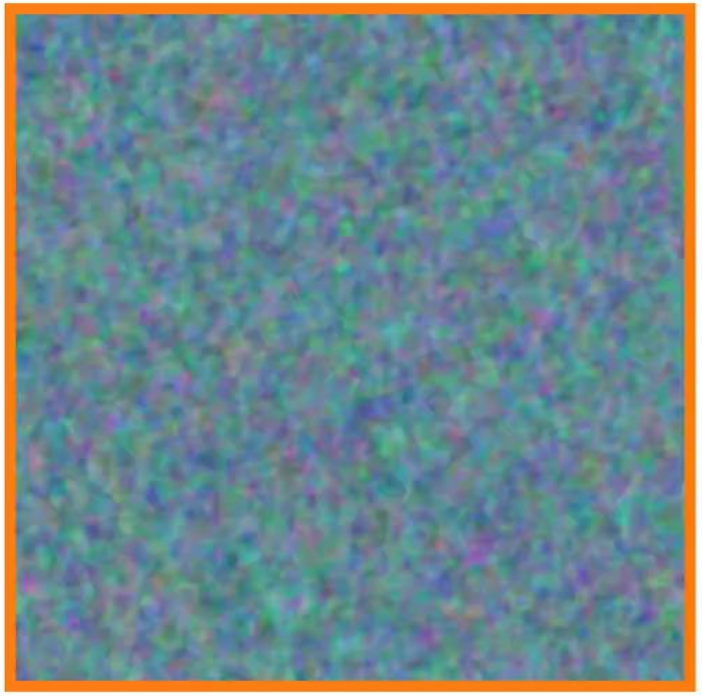
Slide: Matt Tancik, **"Encoding and Representing 3D Volumes"**, NeRF Tutorial, ECCV 2022

NeRF (Neural Radiance Field; 2020)

- Key idea: **Positional encoding** transforms a real number $p \rightarrow 2L$-dimensional real numbers.

$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \cdots \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right)$$

  - Q) Why? 3.1415 and 3.1414 may generate similar values.
  - A) Positional encoding highlights not only large values but also small fraction numbers.
  - Note) $L = 10$ for $\gamma(\mathbf{x})$ and $L = 4$ for $\gamma(\mathbf{d})$



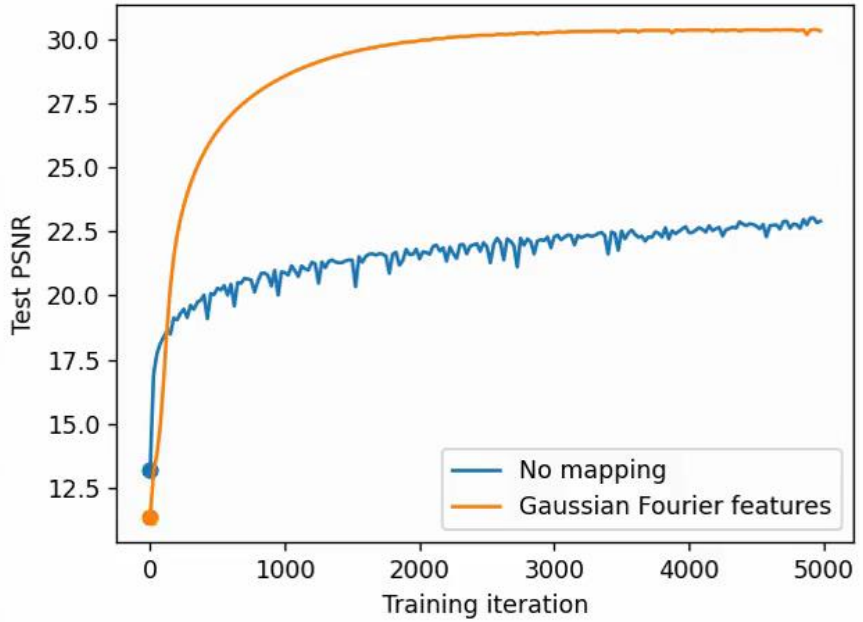
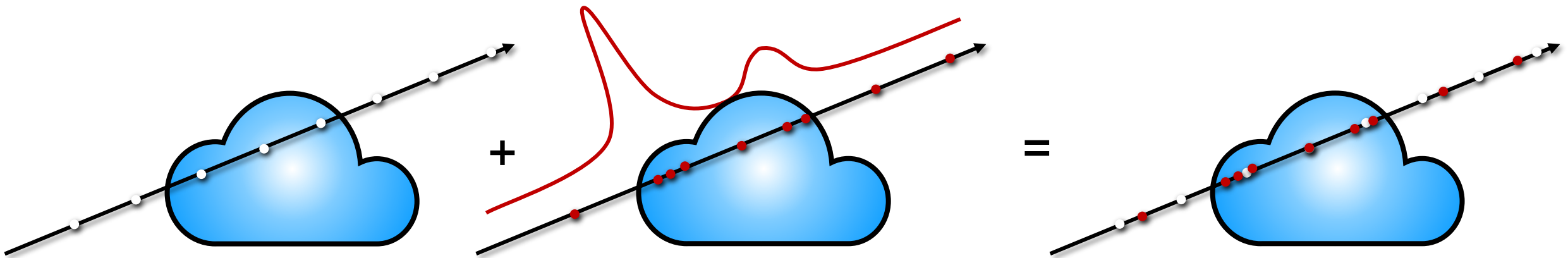


Image: Mildenhall et al., **"Representing Scenes as Neural Radiance Fields for View Synthesis"**, ECCV, 2020 Homepage

NeRF (Neural Radiance Field; 2020)

- Key idea: **Hierarchical volume sampling** selects points according to uniform and non-uniform weights.
  - Q) How to assign non-uniform weights?

$$C(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i = \sum_{i=1}^{N} w_i \mathbf{c}_i \quad \rightarrow \quad \hat{w}_i = \frac{w_i}{\sum w_i}$$



Uniform sampling Weighted sampling

NeRF (Neural Radiance Field; 2020)

- **Inference**: Synthesizing a **2D image** with a *new* viewpoint
  - Input: A new camera viewpoint (R_n, $\mathbf{t}_n$)
  - Neural volumetric rendering
- **Inference**: Retrieval of a **3D model** from density values
  - e.g. Normal vectors from analytic gradient of density

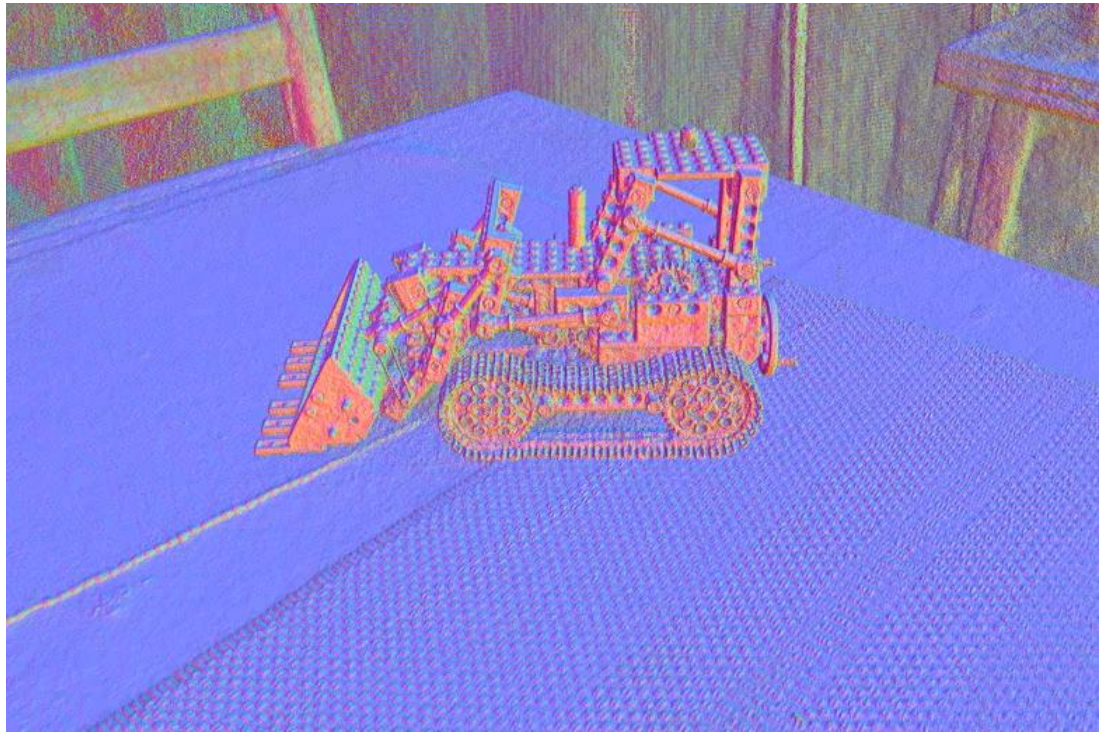


Image: Ben Mildenhall, **"Deep Dive into the Volumetric Rendering Function"**, NeRF Tutorial, ECCV 2022

Summary

- **What is 3D Vision?**
- **Visual Geometry**
- **3D Reconstruction**
  - **Structure-from-Motion (SfM)**
    - The *flower* of visual geometry
  - **COLMAP (2016)**
    - The state-of-the-art incremental SfM and more
- **3D Representations**
  - **NeRF (Neural Radiance Field; 2020)**
    - Network: 11 fully-connected (shortly FC) layers
    - Inference (new view synthesis): Neural **volumetric rendering**
    - Training (learning the 3D scene)
      - Loss function: Rendering loss (MSE) between input images and syn
    - More ideas: **Positional encoding**, **hierarchical volume sampling**