# EECE5698
# Networked XR Systems

# Lecture Outline for Today

- Advances in novel view synthesis
  - NeRF
  - Gaussian Splatting
- Final Quiz
- Summary of the course

# Novel View Synthesis

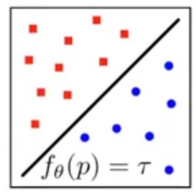
- Given a set of sparse images that are captured from different directions, compute a continuous scene

# NeRF

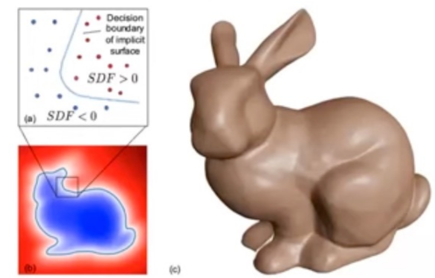
https://www.matthewtancik.com/nerf

# NeRF

**Occupancy Networks**
(Mescheder et al. 2019)
$(x, y, z) \rightarrow$ occupancy

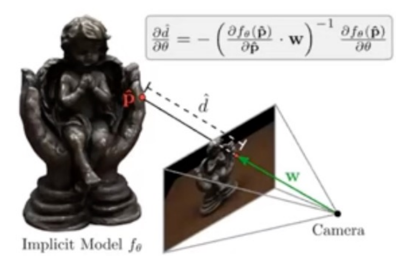



**DeepSDF**
(Park et al. 2019)
$(x, y, z) \rightarrow$ distance



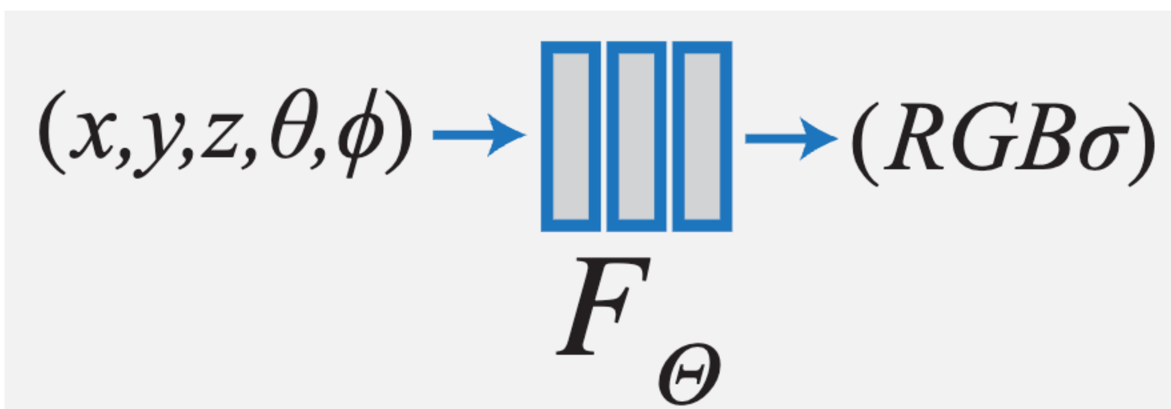

**Scene Representation Networks**
(Sitzmann et al. 2019)
$(x, y, z) \rightarrow$ latent vec. (color, dist.)

**Differentiable Volumetric Rendering**
(Niemeyer et al. 2020)
$(x, y, z) \rightarrow$ color, occ.



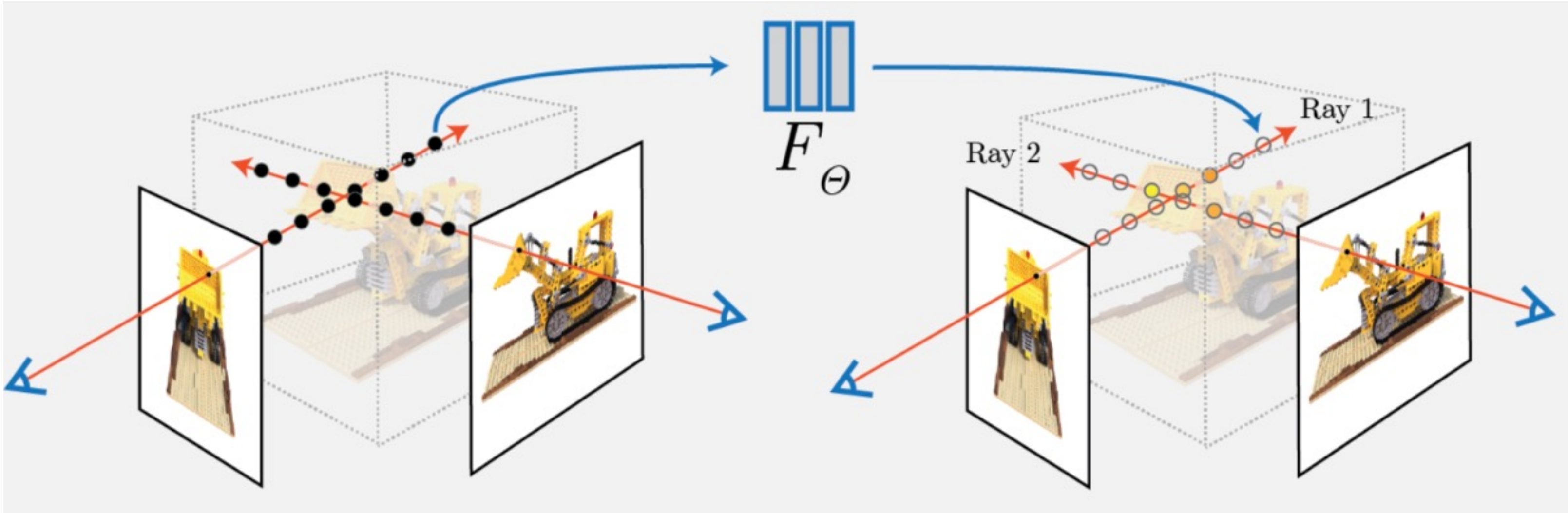
# NeRF

- NeRF
- Input: spatial location (x, y, z) and viewing direction (θ, φ)
- Output: volume density and view-dependent emitted radiance



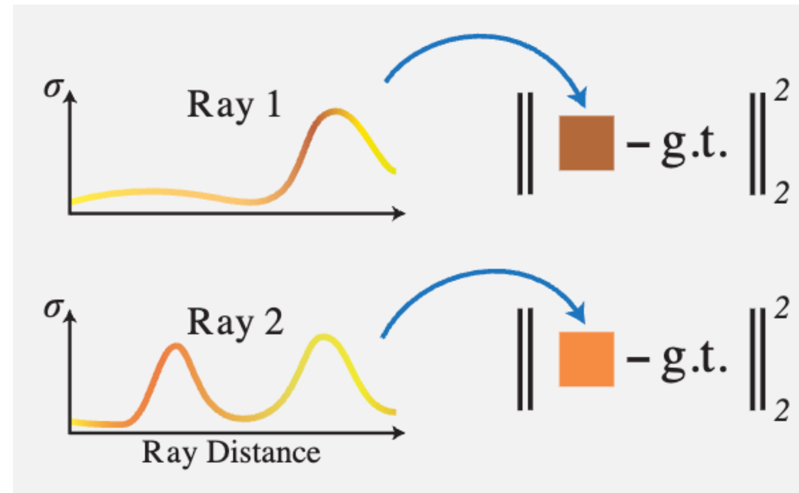
# NeRF

Query 5D coordinates along camera rays – ray tracing

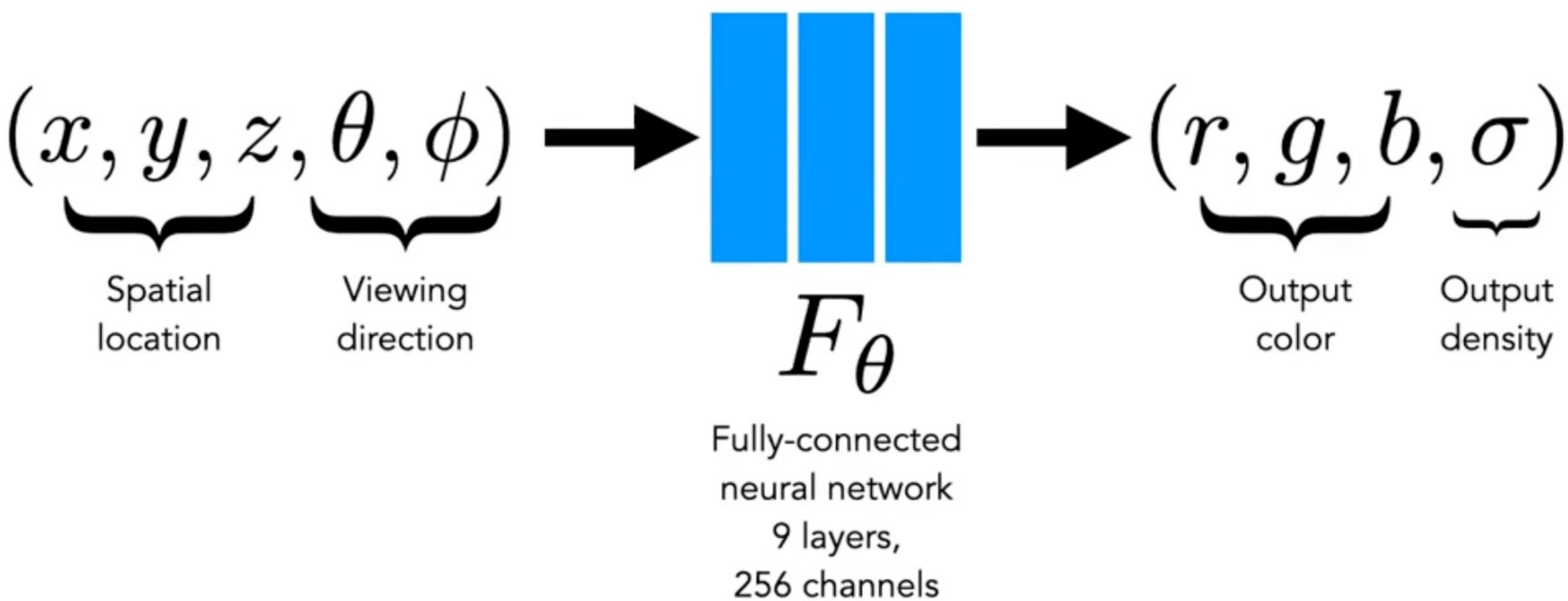


# NeRF

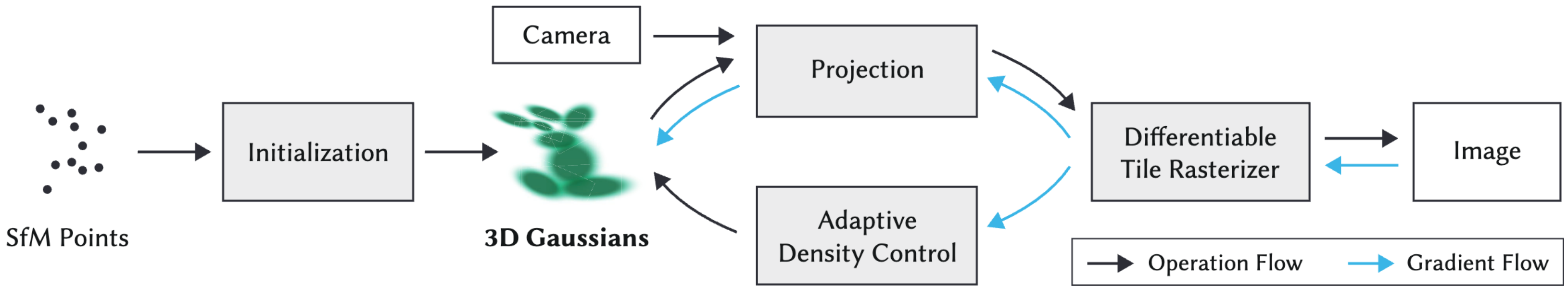
- Memorize the scene
- Weights are the scene




Volume Rendering Rendering Loss

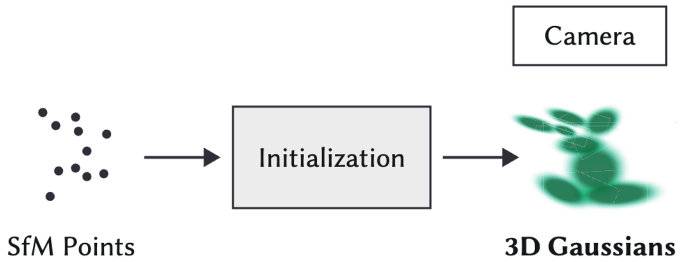


# Gaussian Splatting



# Gaussian Splatting

- Starting from sparse points produced during camera calibration, SFM
- From these points, we create a set of 3D Gaussians defined by a position (mean), covariance matrix, and opacity $\alpha$



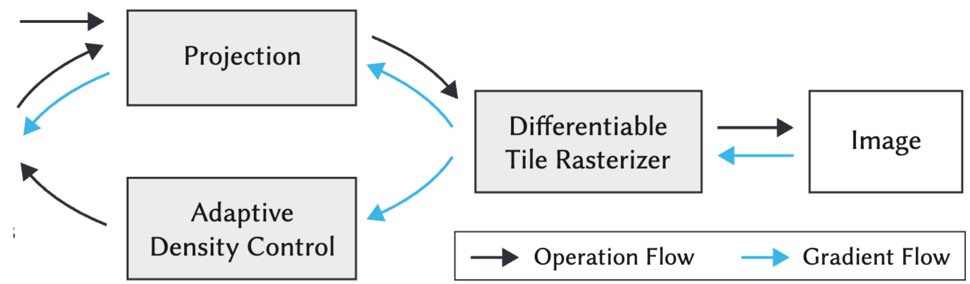

Step1

# Gaussian Splatting

**Algorithm 2** GPU software rasterization of 3D Gaussians
$w, h$: width and height of the image to rasterize
$M, S$: Gaussian means and covariances in world space
$C, A$: Gaussian colors and opacities
$V$: view configuration of current camera

**function** RASTERIZE($w, h, M, S, C, A, V$)
    CullGaussian($p, V$) ▷ Frustum Culling
    $M', S' \leftarrow$ ScreenspaceGaussians($M, S, V$) ▷ Transform
    $T \leftarrow$ CreateTiles($w, h$)
    $L, K \leftarrow$ DuplicateWithKeys($M', T$) ▷ Indices and Keys
    SortByKeys($K, L$) ▷ Globally Sort
    $R \leftarrow$ IdentifyTileRanges($T, K$)
    $I \leftarrow \mathbf{0}$ ▷ Init Canvas
    **for all** Tiles $t$ **in** $I$ **do**
        **for all** Pixels $i$ **in** $t$ **do**
            $r \leftarrow$ GetTileRange($R, t$)
            $I[i] \leftarrow$ BlendInOrder($i, L, r, K, M', S', C, A$)
        **end for**
    **end for**
    **return** $I$
**end function**

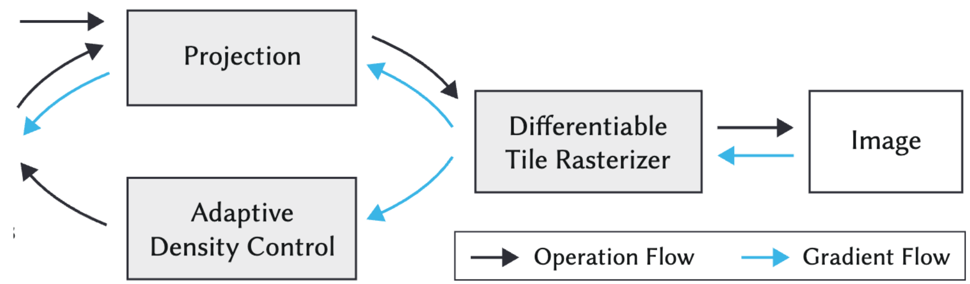
Step 2




1. Collect the Gaussians in the view frustum
2. Project and split the space into 16xx16 tiles
3. Associate tiles with IDs, and sort them with Radix sort
4. Alpha-blending

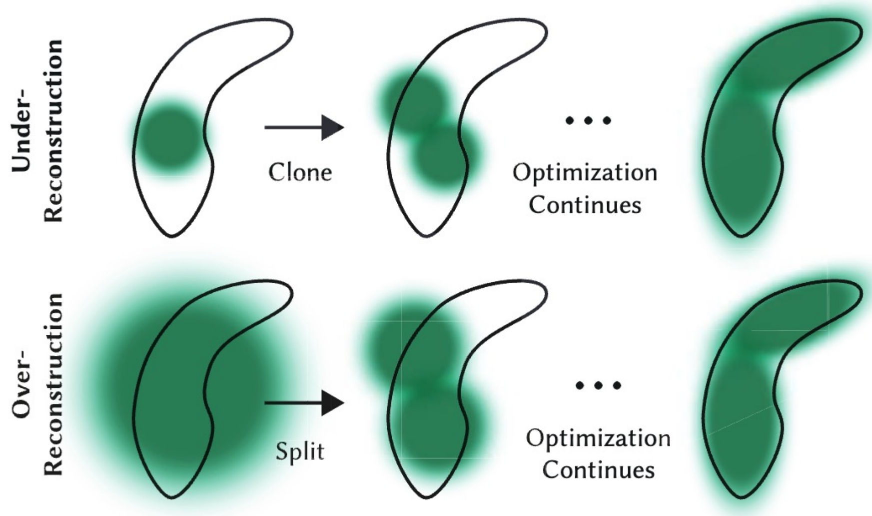
# Gaussian Splatting

Step 3

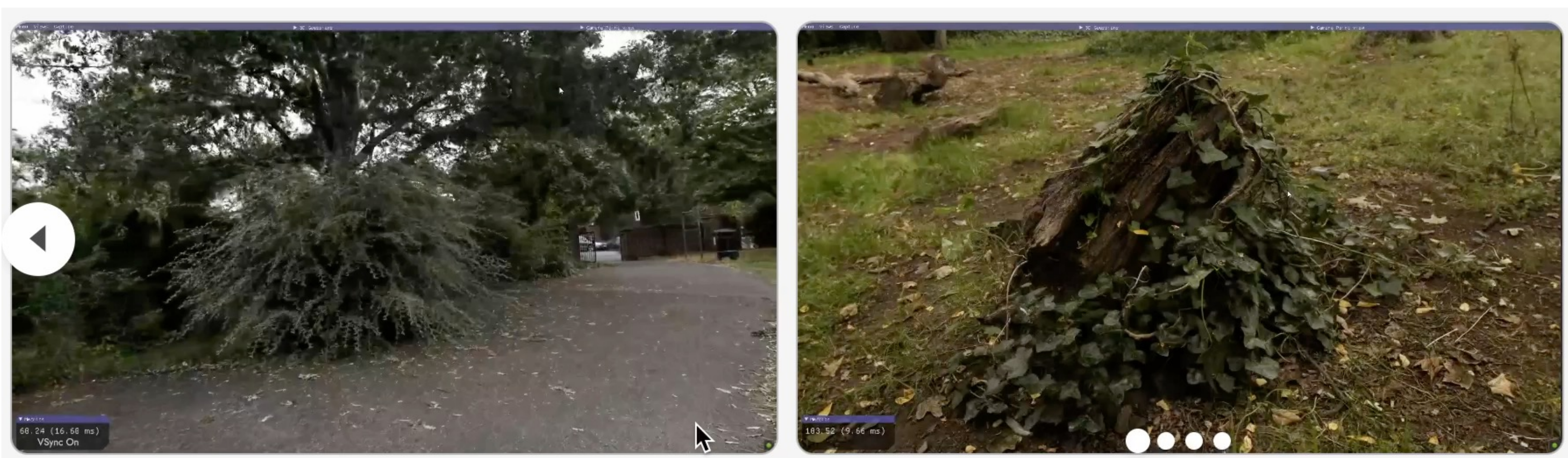



Adaptive density control

1. Under-reconstruction – Clone
2. Over-reconstruction - Split



# Gaussian Splatting



# Quiz

# Summary of the Course

- Fundamental problems of networked applications
- XR content representations
- 2D, Flat 360, 3D/Volumetric videos (RGB-D, point cloud, mesh, NeRF)
- Monocular, stereoscopic, and multiview videos
- Acquiring XR content for network delivery
- Compression algorithms for RGB and depth videos
- Compression algorithms for point cloud and mesh sequences
- Multiview compression algorithms
- Streaming fundamentals
- Stored, live, and interactive streaming protocols
- Streaming XR content (videos, point clouds, meshes, holograms, spaces)
- Content delivery networks
- Local streaming via WiFi, mmWave and optical wireless links
- Remote and hybrid rendering
- Visual and wireless sensing for tracking