EECE5512
Networked XR Systems

Previous class

- Tracking Fundamentals
  - Eyes
  - Face
  - Gestures
  - Hands
  - Head

- Sensors and algorithms

Lecture Outline for Today

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

Tracking in XR - Recap

- What is Tracking?
  - The process of continuously determining the position and orientation of a user's device or body parts within a given space, such as hands, face, or eyes.

Tracking in XR - Recap

- Why do we need Tracking?
  - Essential for creating an immersive and interactive experience, as it allows the virtual environment to respond dynamically to the user’s movements.
  - E.g., hand tracking in AVP eliminates the need for controllers

Hand Tracking

- A system to detect, track, and interpret the movements and positions of a user's hands and fingers in real-time.
- Why?
  - Enables users to interact with digital environments and interfaces in a natural and intuitive way, using their hands and gestures directly, without the need for physical controllers or input devices.

Hand Tracking

Applications

Hand Tracking

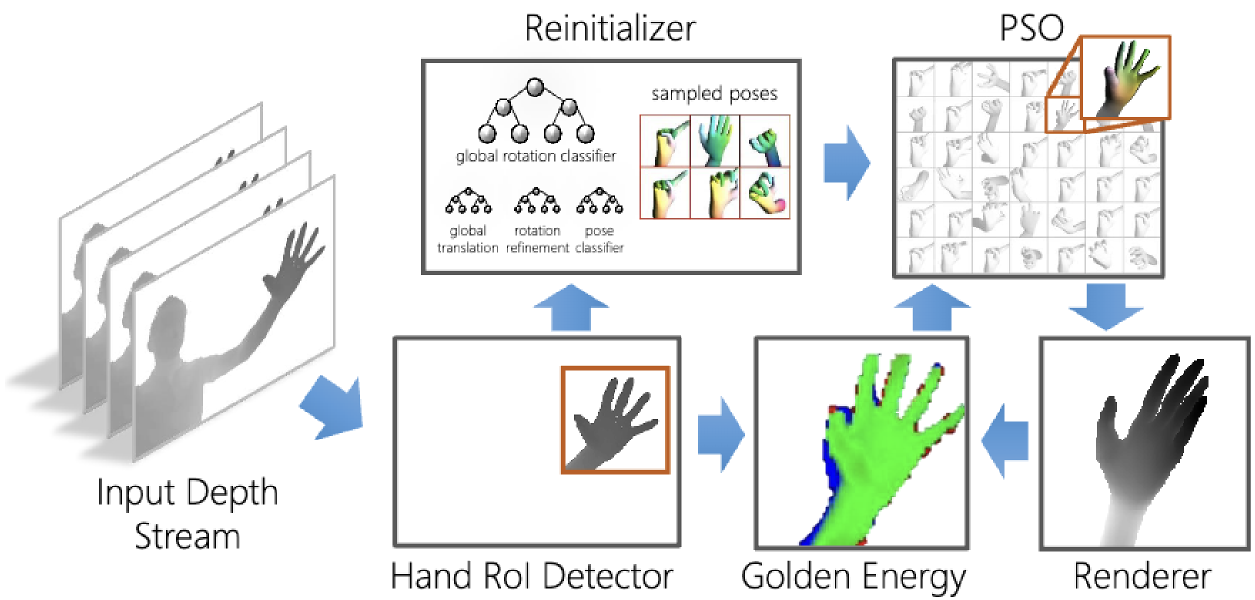
- Early Developments: Hand tracking roots in the 1960s with simple gesture recognition systems.
- 1990s to 2000s: Evolution from wired gloves to marker-based optical systems.
- Leap in Technology: Leap Motion (2010) and Microsoft Kinect (2010) popularized hand tracking with advanced depth sensors and computer vision.
- Recent Advances: Integration in VR/AR headsets, e.g., Oculus Quest's hand tracking feature (2019).

Hand Tracking

- **How It Works:**
  - Cameras and sensors capture the hand's position and movements.
  - Software analyzes images and sensor data to identify hand shapes and gestures.
- **Tracking Methods:**
  - Optical Tracking: Uses cameras to detect hand position and movement.
  - Inertial Tracking: Employs accelerometers and gyroscopes to measure motion.
  - Electromagnetic Tracking: Uses magnetic fields to detect hand position and orientation.

Hand Tracking

- Case Study: Microsoft's hand tracking



Sharp et.al, Accurate, Robust, and Flexible Real-time Hand Tracking, CHI'15

Hand Tracking

- Case Study: Microsoft's hand tracking

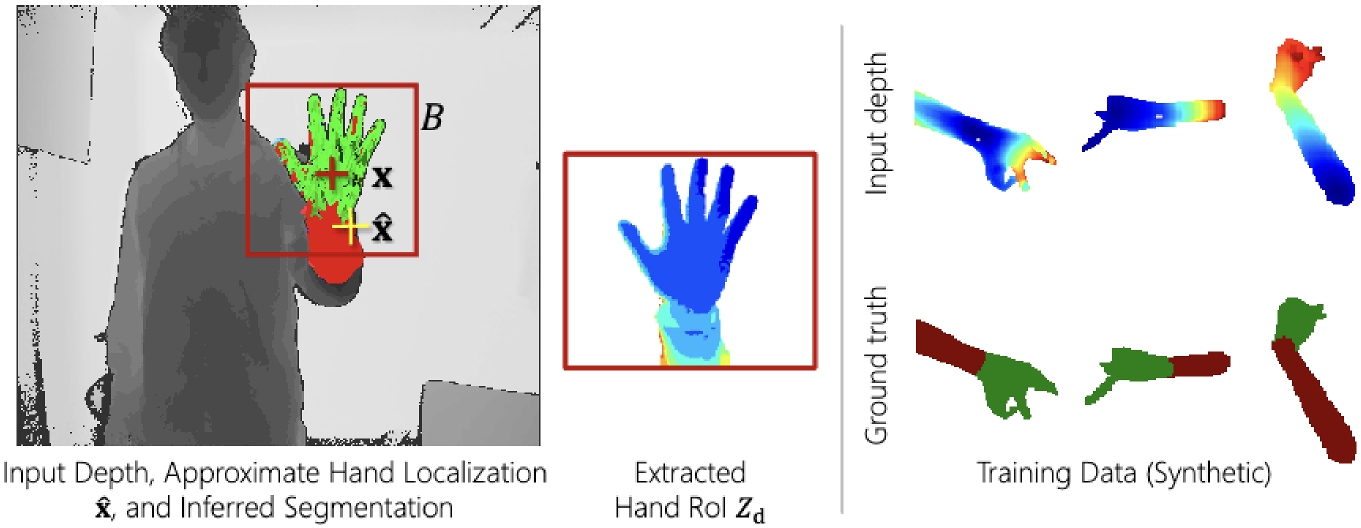
- **Hand RoI extraction**: Identify a square region of interest (RoI) around the hand and segment hand from background.
- **Reinitialization**: Infer a hierarchical distribution over hand poses with a layered discriminative model applied to the RoI.
- **Model fitting**: Optimize a ‘population’ of hand pose hypotheses (‘particles’) using a stochastic optimizer based on particle swarm optimization (PSO)

Hand Tracking

- 3D Hand Model
  - human hand as a 3D model, represented by a detailed mesh of triangles and vertices. The 3D positions of the M mesh vertices are represented as columns in a $3 \times M$ matrix V that defines the hand shape in a ‘base’ (rest) pose.
  - Includes wrist, finger, and thumb joints – e.g., rotations and translation matrices for all the joints (i.e., 3 joints per finger) and wrist, etc., comprising pose vector ($\boldsymbol{\theta}$)
  - Input: depth image
  - Output: $\phi(\boldsymbol{\theta}, V)$ i.e., hand mesh in a given pose

Hand Tracking

- RoI extraction



Input Depth, Approximate Hand Localization $\hat{\mathbf{x}}$, and Inferred Segmentation

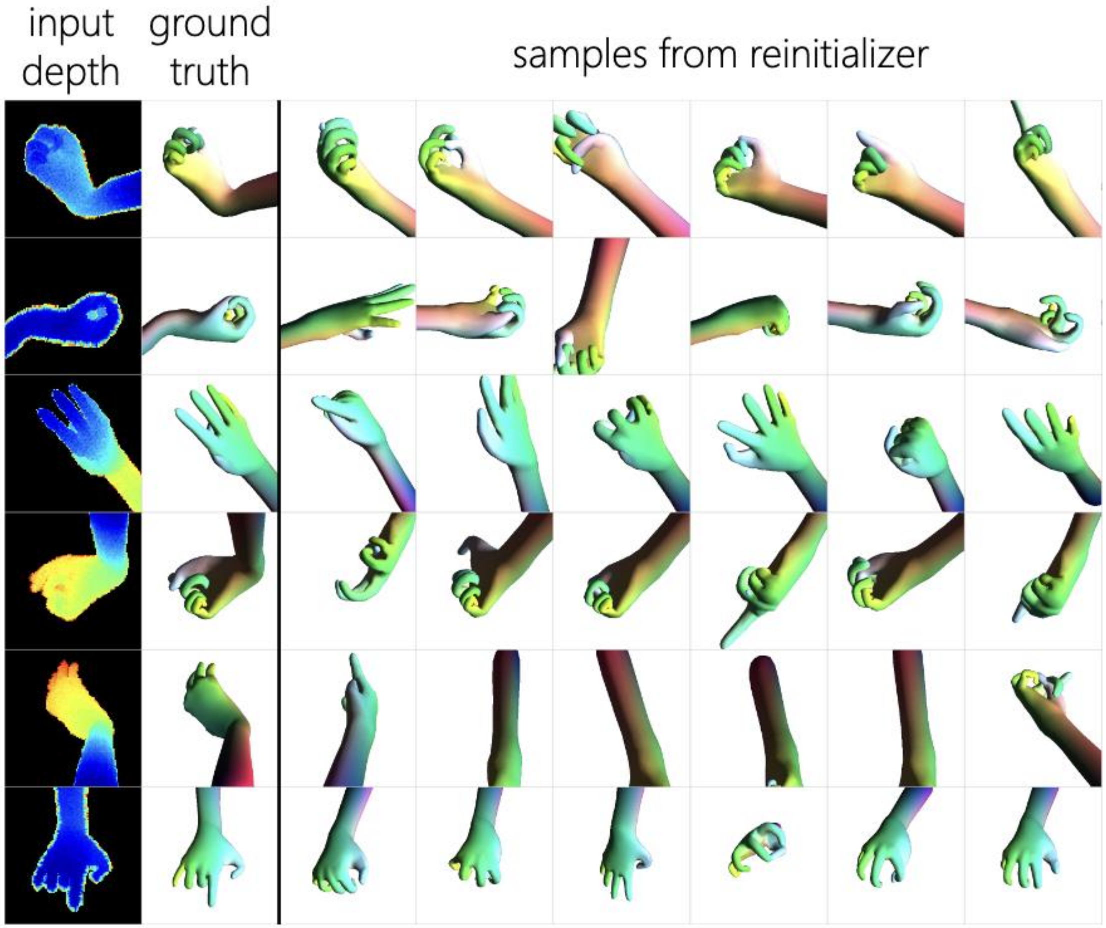
Extracted Hand RoI $Z_{\mathbf{d}}$

Hand Tracking

- Reinitialization
  - Output a pool of hypotheses of the full hand pose by observing just the current input depth image
  - It is difficult to predict a single good pose solution
  - Instead, predict a distribution over poses, and fit a model that will quickly sample as many poses as desired and use the golden energy to disambiguate the good from the bad candidate

Hand Tracking

- Sample poses output from the reinitializer



Hand Tracking

- Model fitting
  - PSO optimizes the following scoring function

$$E^{\mathrm{Au}}(Z_{\mathrm{roi}}, R_{\mathrm{roi}}) = \sum_{ij} \rho(\bar{z}_{ij} - r_{ij})$$

1.Evaluation and Scoring: The algorithm evaluates a scoring function across the particle population in parallel on the GPU, with each evaluation determining the hand pose's "energy."

2.Particle Randomization and Updates:

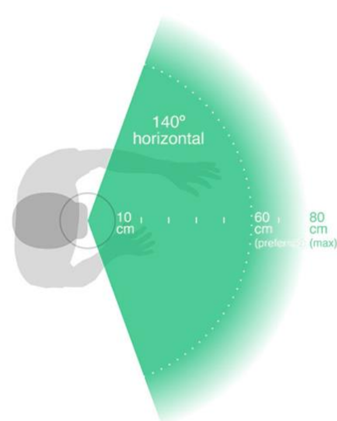
1. Regular randomization of particles to prevent stagnation: per-generation adjustments for fingers and every-third-generation for broader pose variations.
2. Updates include standard PSO dynamics with added mechanisms for local minima attraction and momentum, plus custom extensions for better performance.

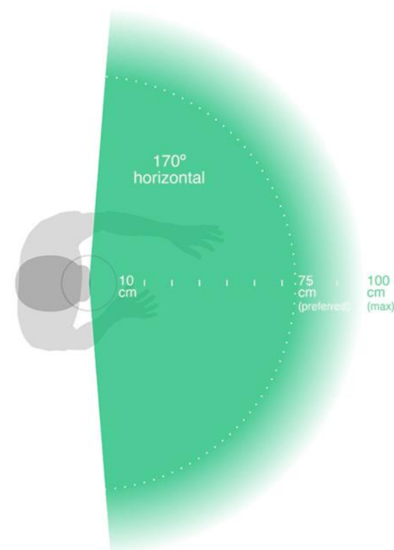
Hand Tracking

- Failure scenarios – why?

Hand Tracking

- Leap motion – ultra leap
  - Two cameras and some infrared LEDs. These track infrared light at a wavelength of 850 nanometers, which is outside the visible light spectrum.
  - Wide angle lenses are used to create a large interaction zone within which a user’s hands can be detected.



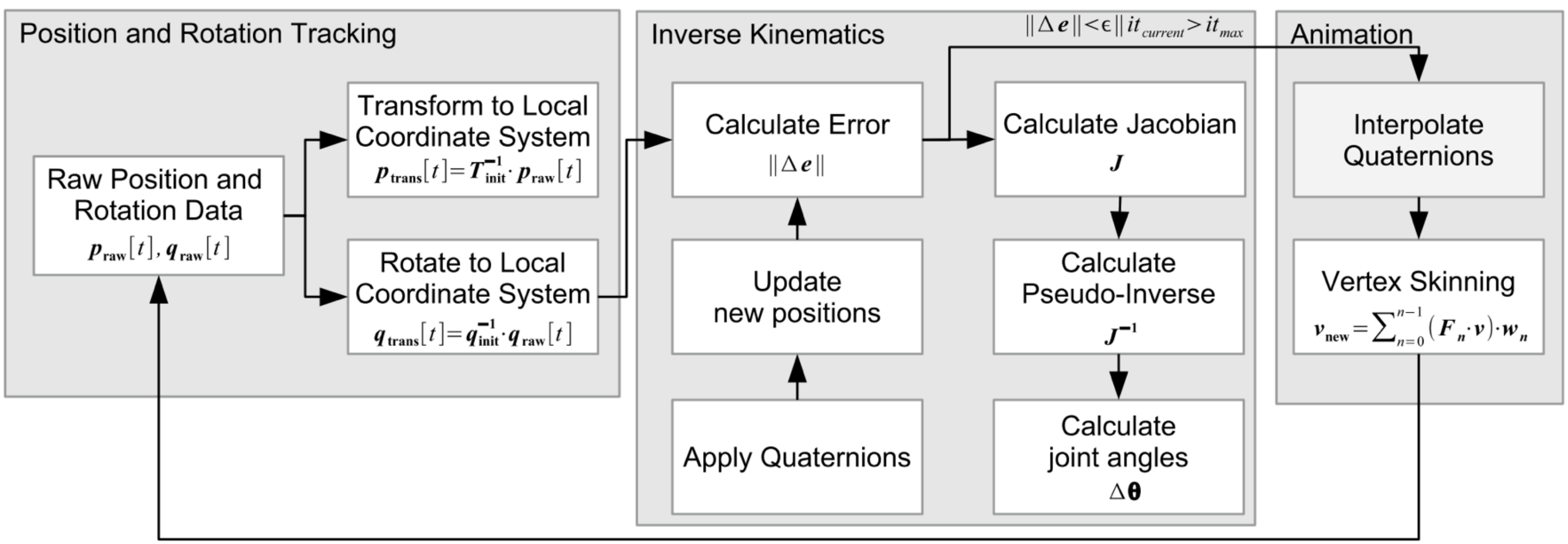


Full Body Tracking

- Capture or track the movements of a person's entire body in real-time.
- Applications

Full Body Tracking

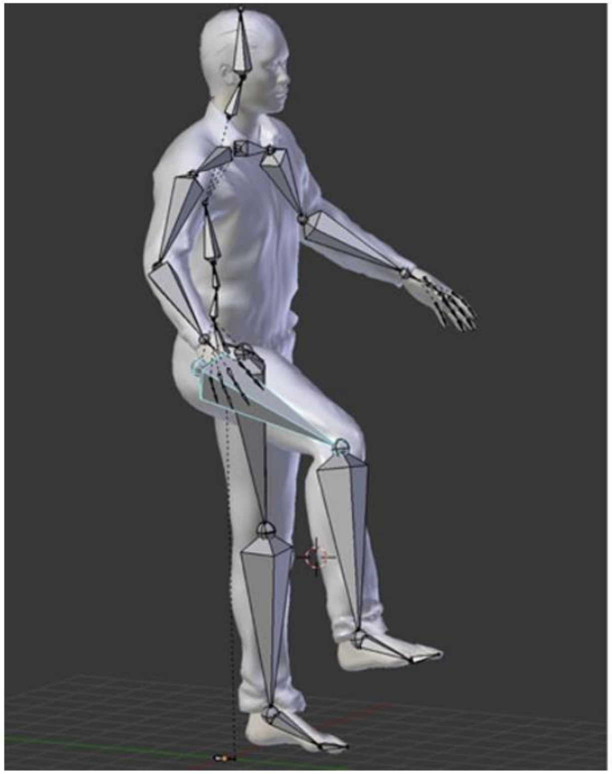
- Inverse Kinematics
  - Find positional and orientational constraints of each specific joint

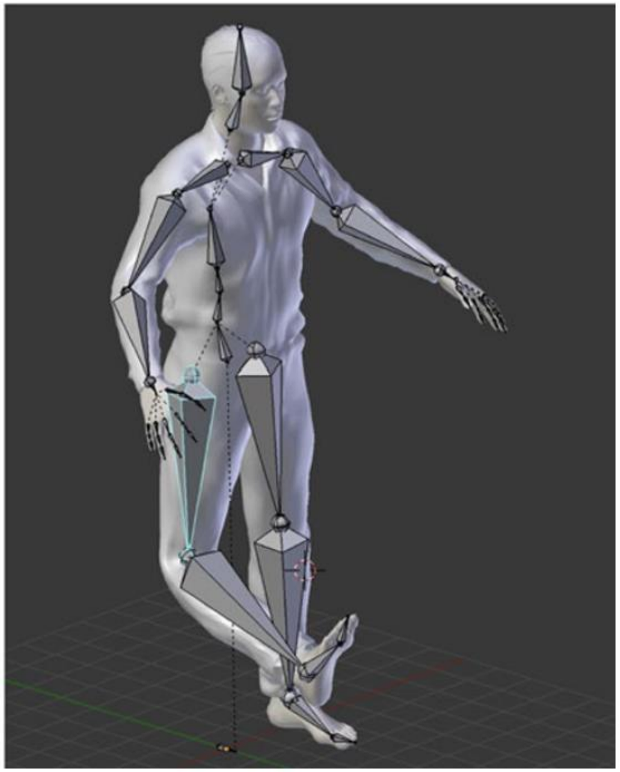


Caserman et.al, Real-time body tracking in virtual reality using a Vive tracker, virtual reality 2019

Full Body Tracking

- Natural body pose





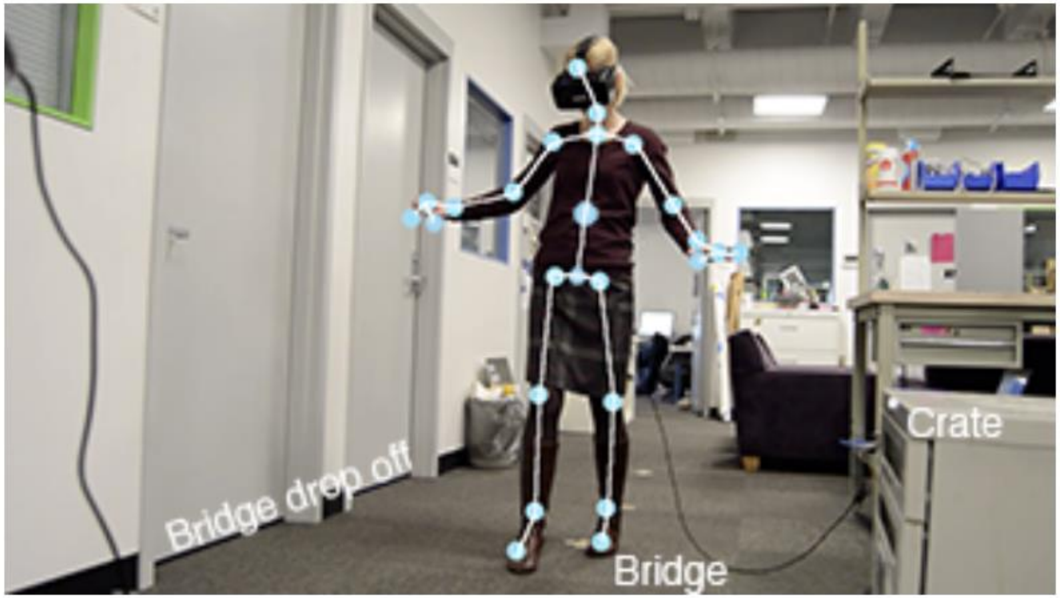
Full Body Tracking

- Algorithm
  1. Calculate error between desired and actual position as well as rotation
  2. Check for convergence
  3. Calculate Jacobian
  4. Calculate Pseudo-Inverse
  5. Calculate joint angles for each bone joint
  6. Apply quaternions to the transformation matrix
  7. Update new positions

Caserman et.al, Real-time body tracking in virtual reality using a Vive tracker, virtual reality 2019

Full Body Tracking

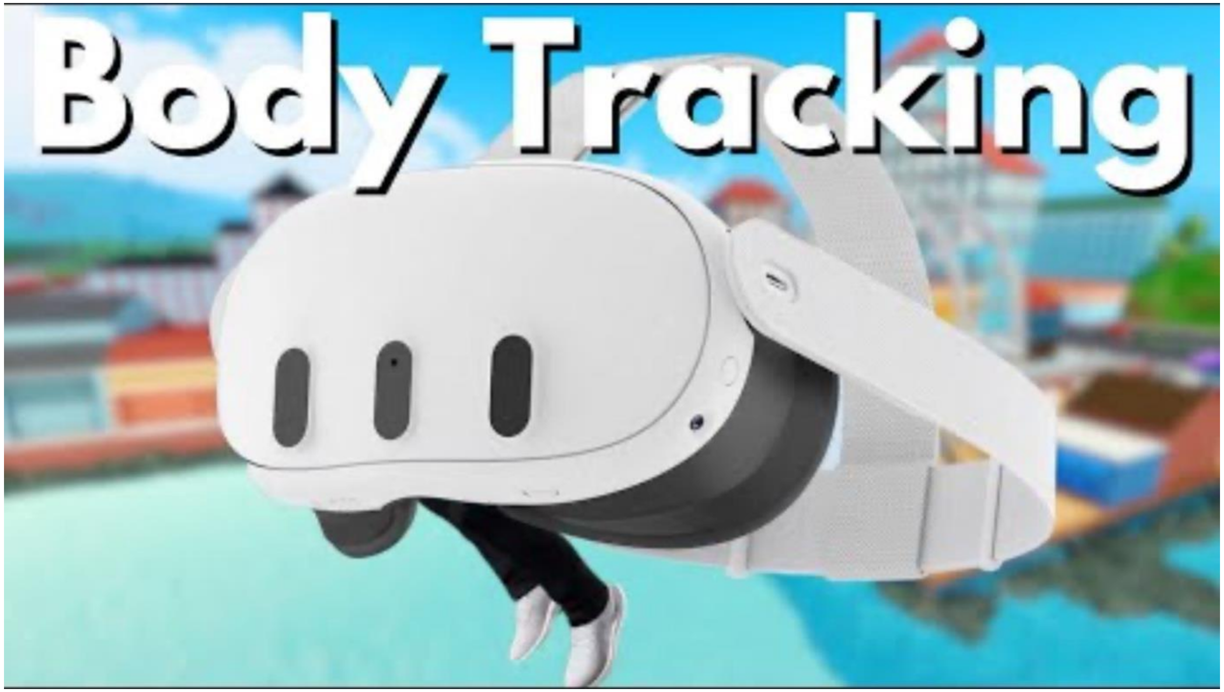
- Kinect sensor



Sra et.al, https://arxiv.org/pdf/1512.02922.pdf

Full Body Tracking

- Meta Quest3



Face Tracking

- Identify and monitor the movements and expressions of a face in real-time.
  - It involves detecting key facial features, such as the eyes, nose, mouth, and jawline, and tracking these features' movements and changes in expression.

Face Tracking

- **Cameras**: 2D and 3D depth cameras are crucial for capturing detailed facial features and movements. 3D cameras provide depth information, essential for accurate tracking in three-dimensional space.
- **Infrared Sensors**: Used in environments with variable lighting to capture the thermal signature of the face, enhancing accuracy in feature detection.

Face Tracking

- **Facial Landmark Detection**: The system identifies key points on the face, such as the corners of the eyes, nose, and mouth, establishing a base for tracking movements.
- **Initial Calibration**: Importance of the initial setup where the system learns the neutral state of the user's face for more accurate tracking.

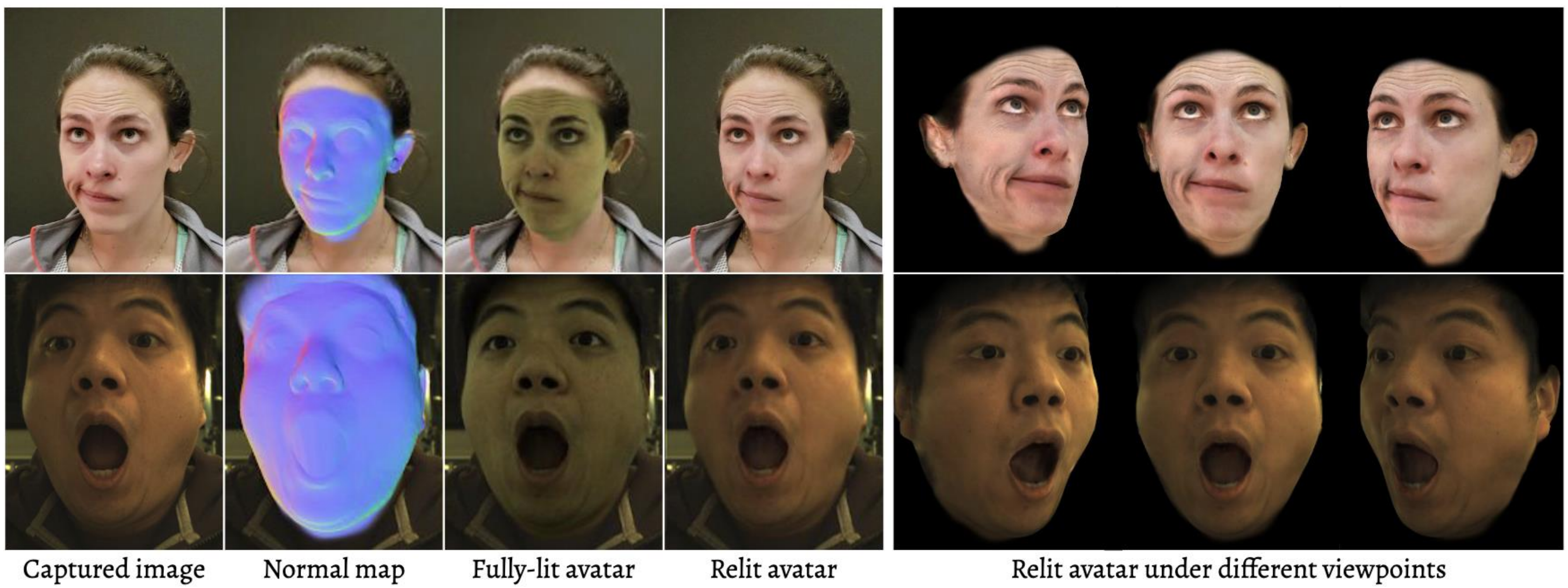
Face Tracking

- **Feature-to-Model Mapping**: Details on how detected facial landmarks are mapped onto a digital model, allowing the system to understand and replicate facial movements.
- **Expression and Gesture Interpretation**: How different facial expressions and head movements are interpreted and translated into digital actions or reactions.

Face Tracking

- **Avatar Animation**: Use of face tracking data to animate avatars in real-time, reflecting the user's expressions and movements in the virtual environment.
- **Realistic Interactions**: Enhancing XR experiences by enabling natural and intuitive interactions, such as nodding, winking, or smiling, to control or influence the digital environment.

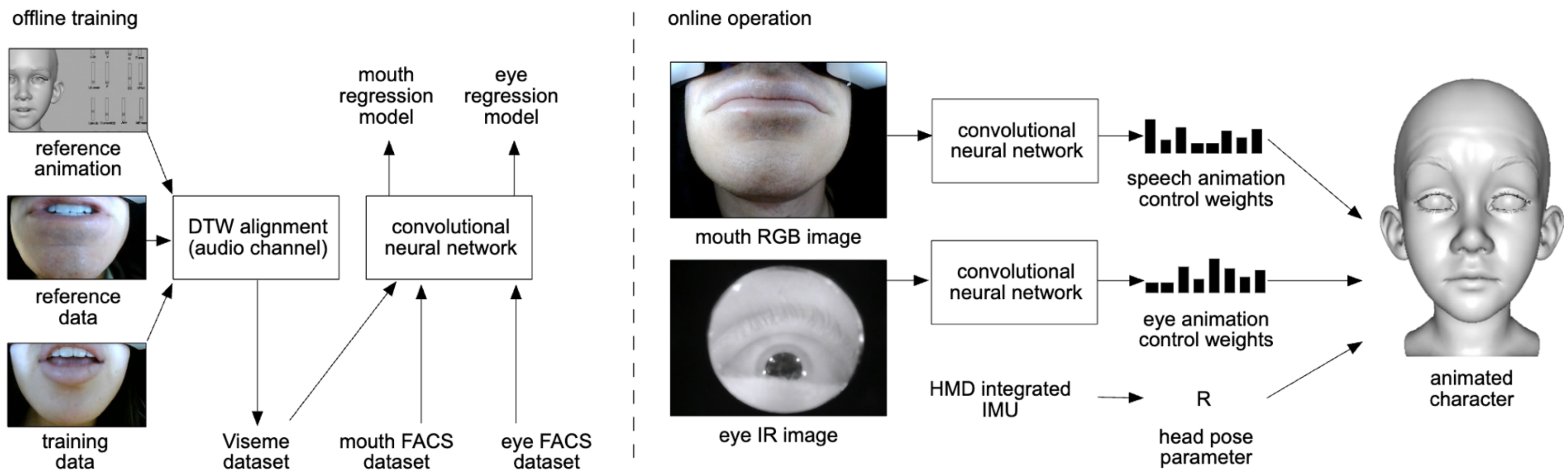
Face Tracking



Meta

Face Tracking

Face and speech animation



K. Olszewski et al

Lecture Outline for Today

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

Novel View Synthesis

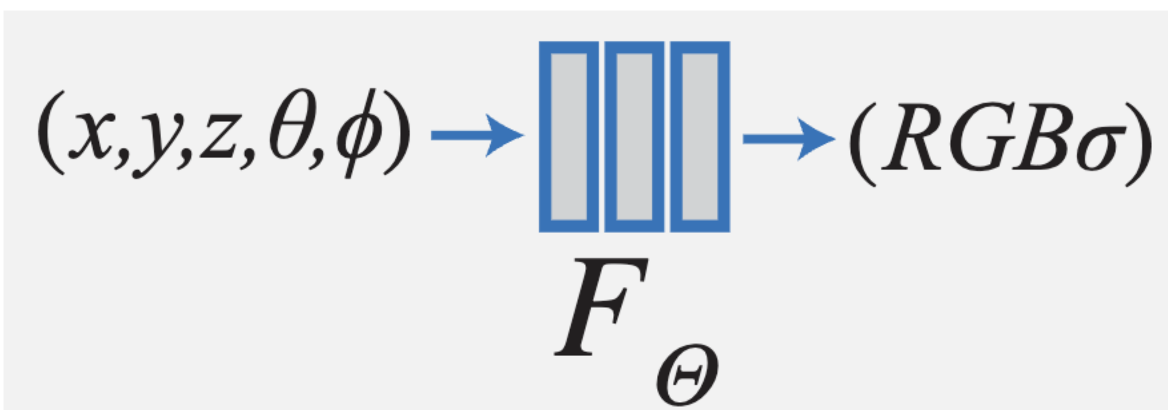
- Given a set of sparse images that are captured from different directions, compute a continuous scene

NeRF

https://www.matthewtancik.com/nerf

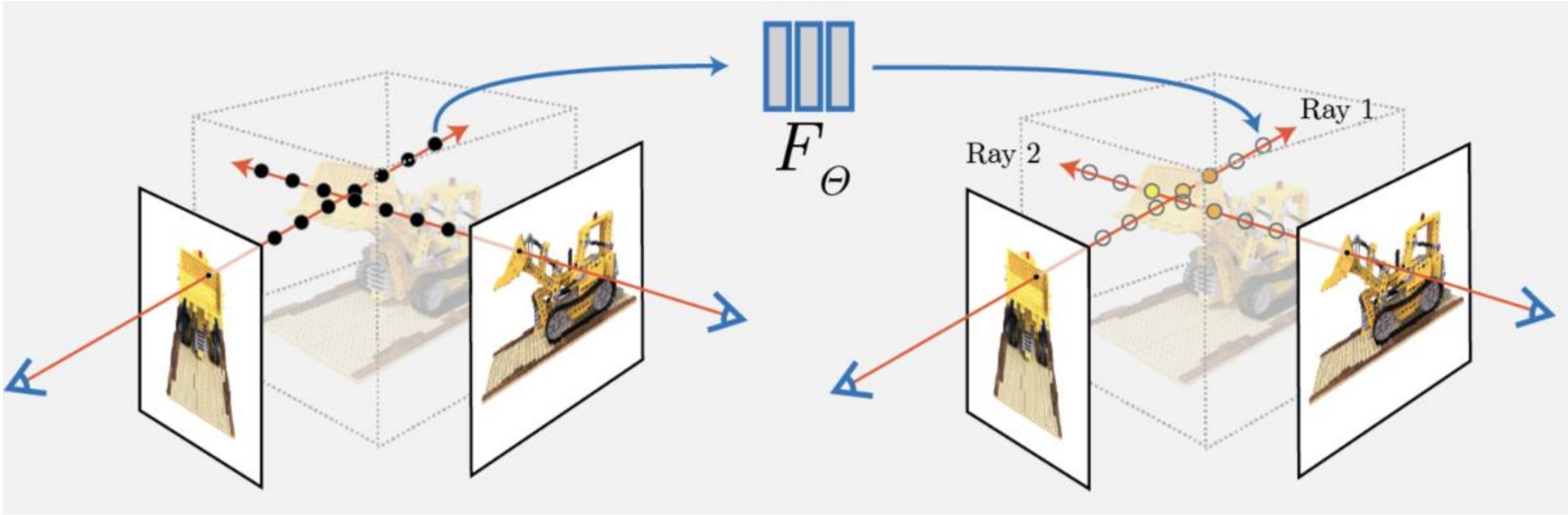
NeRF

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



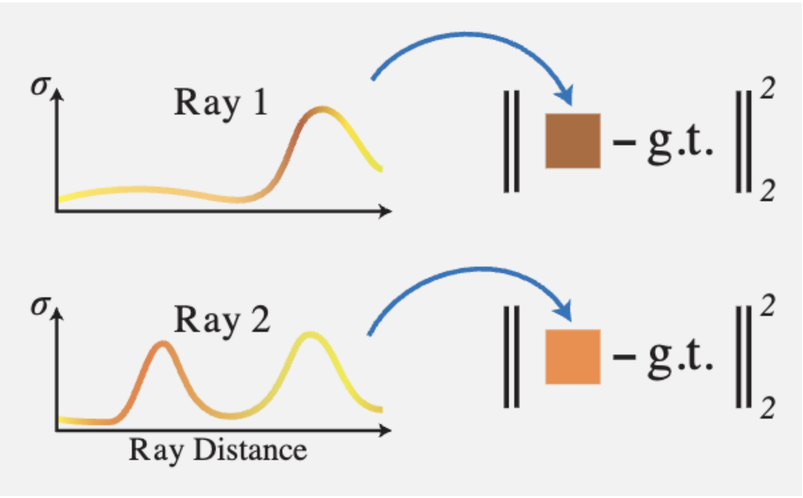
NeRF

Query 5D coordinates along camera rays – ray tracing

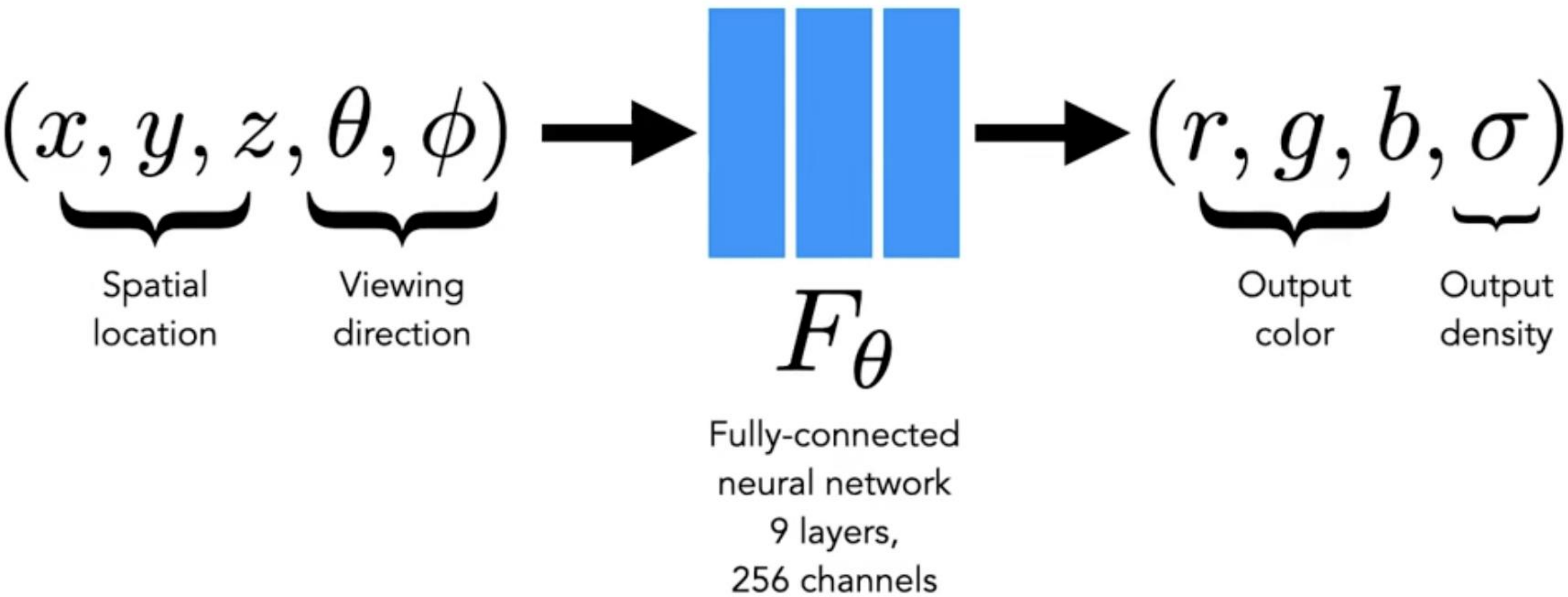


NeRF

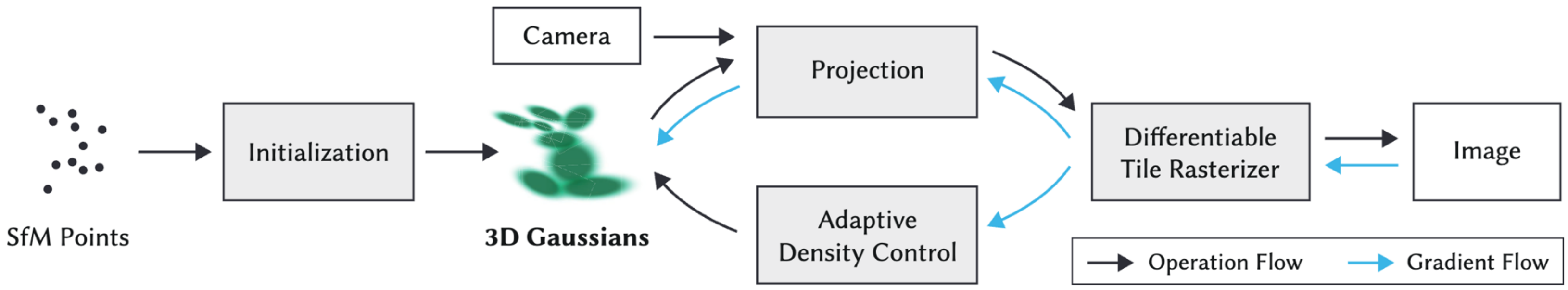
- Memorize the scene
- Weights are the scene



Volume Rendering Rendering Loss

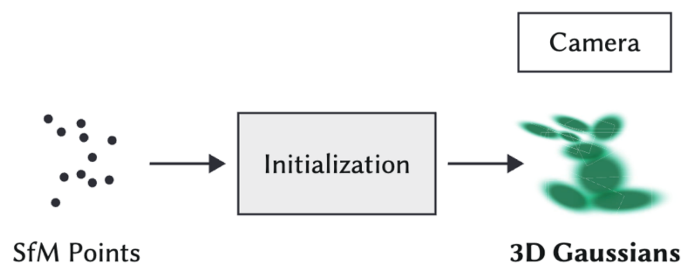


Gaussian Splatting



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 α



Step1

Gaussian Splatting

Algorithm 2 GPU software rasterization of 3D Gaussians

w, h: width and height of the image to rasterize

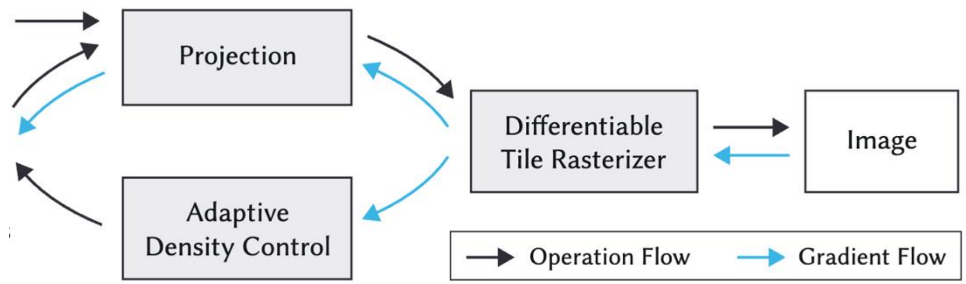
M, S: Gaussian means and covariances in world space

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′ ← ScreenspaceGaussians(M, S, V)          ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M′, T)                 ▷ Indices and Keys
    SortByKeys(K, L)                                ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                           ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
    return I
end function
```

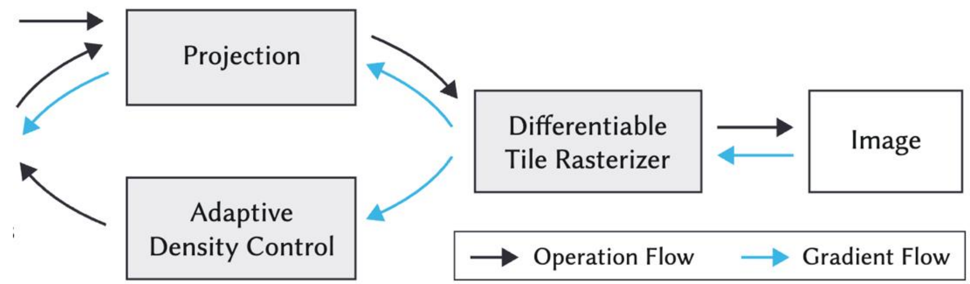
Step 2



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

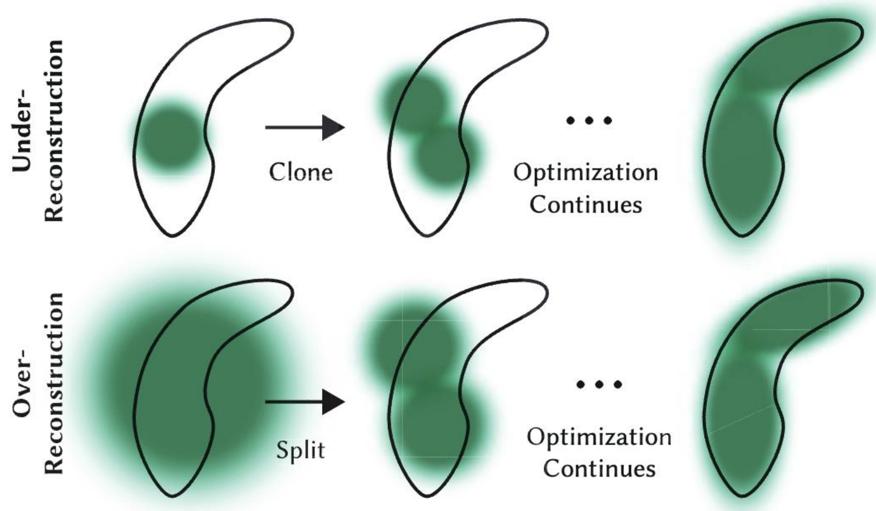
Gaussian Splatting

Step 3



Adaptive density control

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



Gaussian Splatting

Lecture Outline for Today

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

Summary of the Course

- Fundamental problems of networked applications
- XR fundamentals, headsets, glasses, wearables
- 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, depth videos, point clouds, mesh sequences
- Multiview compression algorithms
- Streaming fundamentals
- Stored, live, and interactive streaming protocols
- Streaming XR content (videos, point clouds, meshes, holograms, spaces)
- Local streaming via WiFi, mmWave and optical wireless links
- Remote and hybrid rendering
- Visual and wireless sensing for person tracking
- Networked XR platforms such as ARKit/Core, Unity, Open3D
- Building XR systems such as 3D telepresence (VR), Spatial Web (AR)
- Tracking fundamentals: Eyes, Hands, Face, Head, Body, etc; Outside-in, Inside-out.