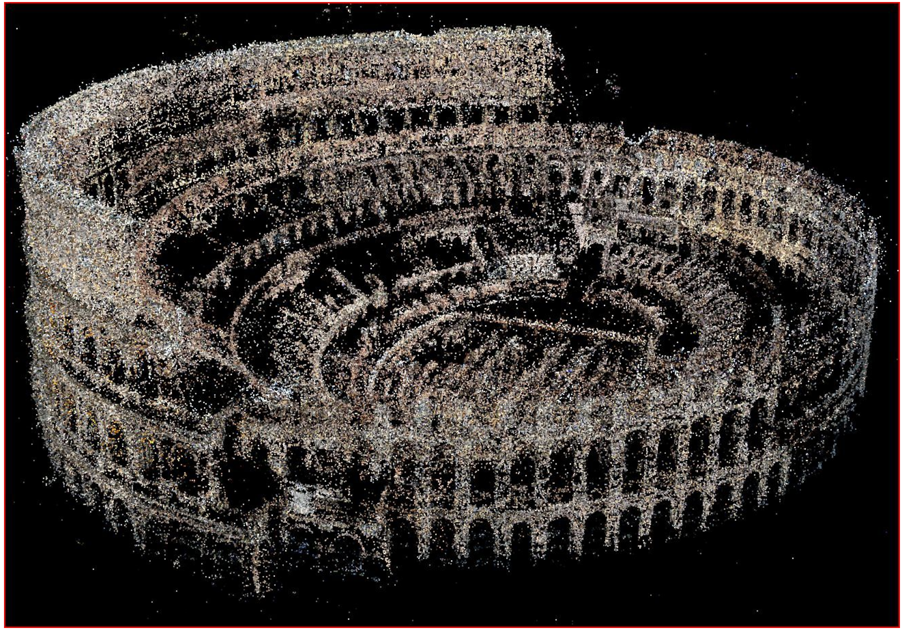
3D Gaussian Splatting

Evan O’Brien | Preksha Sarda

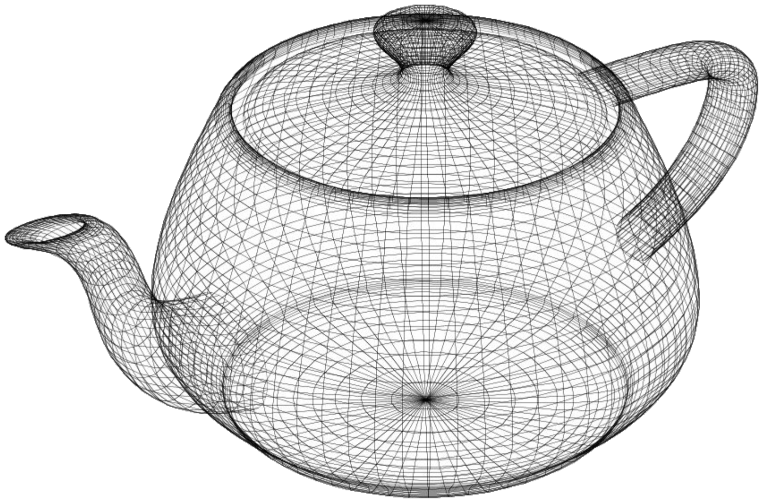
Point Clouds

- World representation using points in 3D space
- Sparse data
  - Efficiency struggles
- Inefficient in storage



Polygons

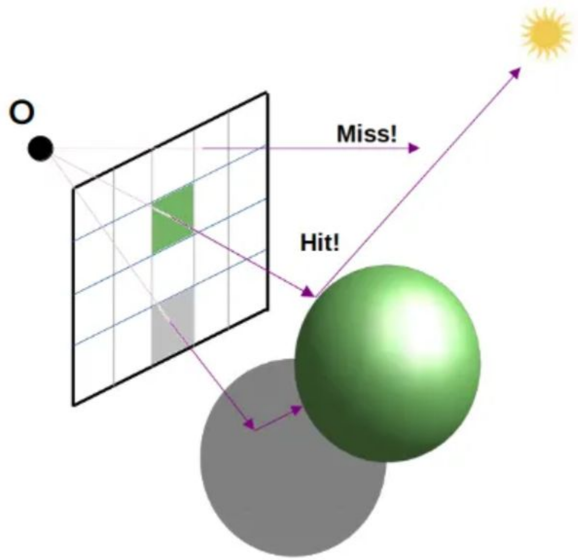
- Current Standard
- Combination of triangles to form an image
- Works well with graphics cards but, doesn't map well to the natural world



Polygon Mesh: 3D representation of an object composed of many polygons (edges, vertices, faces).

NeRF

- **Nerf:** Neural Radiance Field
- Uses raytracing
- Comparing to a Point Cloud: Adds the following attributes:
  - Covariance: how it's stretched/scaled (3x3 matrix)
  - Alpha: how transparent it is (α)
- Solves sparsity issue of the point cloud
- All these attributes are learnable using self-supervised training

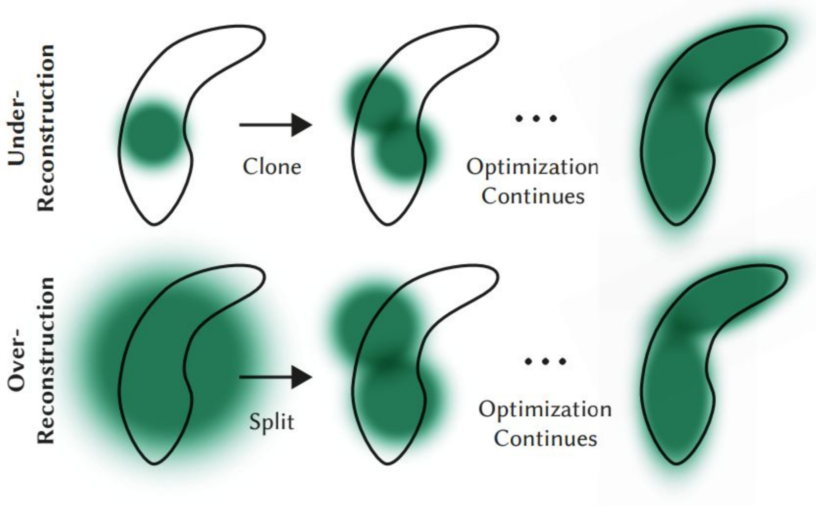


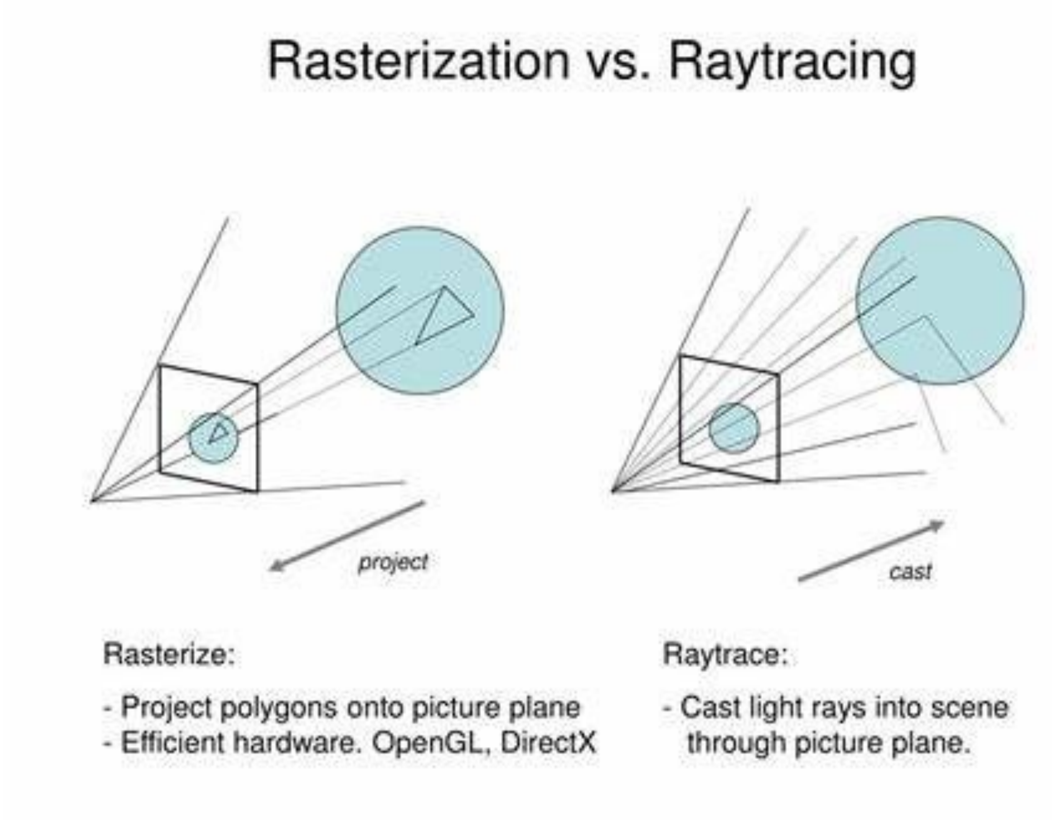
Ray Tracing

“If traditional 3D representations like polygonal meshes are akin to vector images, NeRFs are like bitmap images: they densely capture the way light radiates from an object or within a scene” -David Luebke, vice president for graphics research at NVIDIA.

3D Gaussian Splatting

- Uses **rasterization** technique
  - Basic
    - Project into 2D
    - Sort by depth
    - Iterate over every gaussian and find its contribution to the pixel
      - Front to back
    - Blend them all together (weighted by alpha channel)
  - Training
    - Seeding
      - Splits into 2 to better represent the space
    - Pruning
      - If opacity of gaussian goes to 0, remove it
- This process is repeated until a low loss is achieved





Source: https://th.bing.com/th/id/OIP.1RBVYOHDgdIGrlE1gw9ZxAAAAA?rs=1&pid=ImgDetMain


3D
GAUSSIAN
SPLATTING

Connection to ML and AI

- Similar to training a 1-layer neural network
  - Shallow learning
- Learn off the loss function and adjust the gaussian parameters
- Initializing with point cloud values produces better results

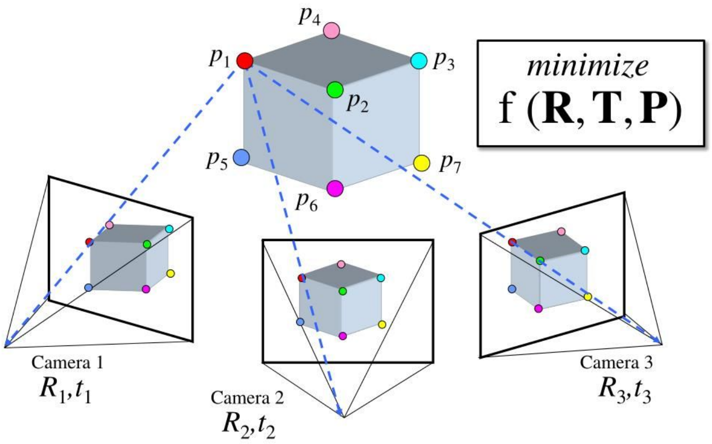
Math

- Learned parameters
  - [[x, y, z], [r, g, b, a], [x_scale, y_scale, z_scale], [q0, q1, q2, q3]]
  - Center, color, 3d scaling vector, quaternion orientation
- For rendering the 3d scaling vector and quaternion are converted to rotation and scaling matrices
- Example: Quaternion: [q0, q1, q2, q3] -> 3D rotation matrix: [[m0, m1, m2], [m3, m4, m5], [m6, m7, m8]]
- Loss function: torch.abs((network_output - ground_truth)).mean() for each pixel in the ground truth and the inference image

Structure From Motion

- A classic computer vision technique for generating point clouds and camera pose estimations from a series of still images
- Similar to Orb SLAM
- This is used to initialize the gaussian

  centers before training starts



https://image1.slideserve.com/2570747/structure-from-motion-l.jpg

Training

- 7K Iterations
  - About 5-8 minutes
  - Trained well
  - Authors stated this was best for balance between time and product
- 30K Iterations
  - About 35 minutes
  - Background artifacts significantly removed

Future Work

- Self supervised models lend themselves to training generative autoencoders. This can provide a framework for making generative models of 3D worlds.
- Style transfer on a 3D foundation model could significantly improve the pace of art design in video game production while possible being more efficient than current techniques.

References

- Papers
  - 3D Gaussian Splatting for Real-Time Radiance Field Rendering (inria.fr)
  - Orb Slam paper: https://arxiv.org/abs/1502.00956
  - Nerf Paper: (https://arxiv.org/abs/2003.08934)
  - NeRF from Scratch. Motivating the explanation from… | by Raja Parikshat | Medium
- Videos
  - https://www.youtube.com/watch?v=VkIJbpdTujE&t=938s&pp=ygUjM2QgZ2F1c3NpYW4gc3BsYXR0aW5nIHJhc3RlcmI6YXRpb24%3D
  - 3D Gaussian Splatting - Why Graphics Will Never Be The Same (youtube.com)