Revolutionizing 3D Scene Representation: From Gaussian Splatting to Advanced Ray Tracing and Unscented Transforms

An In-depth Look at 3DGS, 3DGRT, and 3DGUT

Kintan Saha

May 19, 2025

Outline

The Challenge: Realistic and Efficient 3D Scene Synthesis

- Creating realistic images of 3D scenes from novel viewpoints is a cornerstone of computer graphics and vision.
- Applications span virtual and augmented reality, robotics, entertainment, and digital preservation.
- Traditional methods often involved complex 3D modeling, texturing, and lighting pipelines.
- Image-based rendering and Structure-from-Motion (SfM) offered paths to reconstruct scenes from photographs.

The Rise of Neural Radiance Fields (NeRF)

- NeRF emerged as a groundbreaking technique, representing a scene as a continuous volumetric function (typically a neural network).
- This function maps 3D coordinates and viewing directions to volume density and emitted radiance (color).
- Achieved unprecedented photorealism for novel view synthesis.
- **However, NeRFs faced challenges:**
    - Slow training times (often hours to days).
    - Slow rendering times due to the need to query the neural network many times along each camera ray.
    - Difficulty in achieving real-time performance, especially for high-resolution images of complex scenes.

The Need for Speed and Quality

- The limitations of NeRF spurred research into methods that could:
  - Maintain or improve visual quality.
  - Significantly reduce training times.
  - Enable real-time rendering ($>$30 frames per second).
- This context set the stage for explicit scene representations that are more amenable to fast rendering, leading to innovations like 3D Gaussian Splatting.

3DGS: The Core Concept

- 3D Gaussian Splatting (3DGS) proposes representing a 3D scene as a collection of explicit 3D Gaussian primitives.
- Each Gaussian is essentially a "fuzzy" ellipsoid characterized by:
  - **Position (Mean μ):** The center of the Gaussian in 3D space.
  - **Covariance Matrix (Σ):** Defines the shape, size, and orientation of the Gaussian. This allows for anisotropic (non-uniform scaling) Gaussians that can adapt to scene geometry.
  - **Opacity (α):** How transparent or opaque the Gaussian is.
  - **Color (Appearance):** Typically represented by Spherical Harmonics (SH) to capture view-dependent lighting effects.
- This explicit, point-based representation is well-suited for fast rasterization.

3DGS: Representing Anisotropic Covariance

A key aspect of 3DGS is the flexible representation of geometry using anisotropic 3D Gaussians.

- The 3D covariance matrix Σ describes the ellipsoid's configuration.
- For optimization stability and intuitive control, Σ is decomposed:

$$\Sigma = RSS^T R^T$$

  - S: A 3D scaling matrix (represented by a 3D vector s).
  - R: A 3D rotation matrix (represented by a quaternion q).
- This decomposition allows independent optimization of scale and rotation.
- Anisotropic Gaussians can stretch and align with surfaces, efficiently representing both fine details and large, smooth areas.

3DGS: Initialization from SfM

Effective initialization is crucial for the optimization process.

- 3DGS typically starts with a sparse point cloud generated as a byproduct of Structure-from-Motion (SfM).
  - SfM algorithms calibrate camera poses and generate a rough 3D structure of the scene from input images.
- Each point from the SfM cloud is used to initialize a 3D Gaussian.
  - Initial position is the SfM point's location.
  - Initial covariance can be estimated based on distances to neighboring points (e.g., isotropic Gaussian with axes equal to the mean distance to the three nearest points).
  - Initial opacity and color can be set to default values.
- This provides a reasonable starting point, significantly better than random initialization for complex real-world scenes.

3DGS: Optimization - Learning the Scene

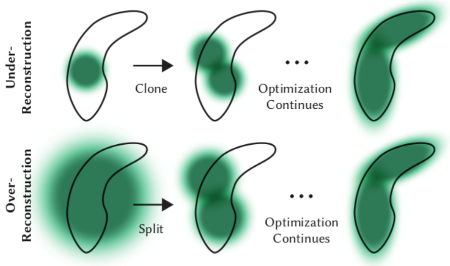
The core of 3DGS involves optimizing the parameters of these Gaussians.

- **Goal:** Make the rendered images from the Gaussians match the input training images.
- **Parameters Optimized:** Position (μ), opacity (α), covariance (via s and q), and Spherical Harmonic coefficients for color.
- **Process:**
  1. Render the current set of Gaussians from a training viewpoint.
  2. Compare the rendered image to the ground truth training image using a loss function.
  3. Compute gradients of the loss with respect to Gaussian parameters.
  4. Update parameters using an optimizer (e.g., Adam).
- **Loss Function:** Often a combination of an L1 difference and a structural similarity term (D-SSIM):

$$L = (1 - \lambda) L_1 + \lambda L_{D-SSIM}$$

3DGS: Adaptive Density Control - Refining the Representation

- **Need for Adaptation:** Initial SfM points are sparse and may not cover all scene details or might be poorly distributed. Hence the number and distribution of Gaussians must adapt.
- 3DGS interleaves optimization with adaptive density control:
  - **Densification:** Adding new Gaussians in regions that are poorly reconstructed.
  - **Pruning:** Removing redundant or unnecessary Gaussians (e.g., those that become too transparent).
- This is typically done periodically (e.g., every 100 iterations after an initial warm-up).



3DGS: Densification Strategies - Cloning and Splitting

Densification focuses on regions with large view-space positional gradients, indicating areas needing refinement.

- **Under-reconstruction (Missing Geometry):**
    - Identified by small Gaussians with high gradients.
    - **Action: Clone.** A copy of the Gaussian is made and moved slightly in the direction of the positional gradient. This helps cover newly forming details.
- **Over-reconstruction (Large Gaussians Covering Detail):**
    - Identified by large Gaussians with high gradients.
    - **Action: Split.** The large Gaussian is replaced by two (or more) smaller Gaussians. Their scale is reduced, and they are sampled from the original Gaussian's distribution. This helps represent finer details within a previously coarse area.
- Pruning removes Gaussians with opacity below a threshold ϵ_α.
- Setting opacity close to zero periodically also helps remove "floater" Gaussians.

3DGS: Rasterization - Projecting 3D Gaussians to 2D

The efficiency of 3DGS hinges on its fast differentiable rasterizer.

- To render an image, 3D Gaussians are projected onto the 2D image plane.
- A 3D Gaussian projects to a 2D Gaussian (an ellipse).
- The projection of the 3D covariance matrix Σ to a 2D covariance matrix Σ' is given by:

$$\Sigma' = JW\Sigma W^T J^T$$

  where:
  - W: Viewing transformation matrix (world to camera space).
  - J: Jacobian of the affine approximation of the projective transformation. This linearizes the potentially non-linear perspective projection. As such this may lead to errors as we shall observe later
- This technique is known as Elliptical Weighted Average (EWA) splatting.

3DGS:Fast Differentiable Rasterizer for Gaussians - Part 1

Initial Pipeline Steps:

1. The screen is **split into 16x16 pixel tiles**
2. All the 3D Gaussians are projected onto the screen using the EWA splatting technique.
3. **3D Gaussians are culled** against the view frustum (based on their 2D EWA splats), keeping those with a 99% confidence interval intersection.
4. A **guard band** is used to trivially reject Gaussians with means at extreme positions (e.g., near the near plane and far outside the frustum), as their projected 2D covariance would be unstable.
5. Each Gaussian is **instantiated (duplicated)** for each 16x16 tile it overlaps. This increases the number of Gaussians to process but simplifies control flow and parallelism.

Sorting and Tile Lists:

- Each Gaussian gets a **key** combining tile ID (high bits) and view-space depth (low 32 bits).
- All keys are **globally sorted** via fast GPU radix sort for depth ordering.
- Alpha blending is **approximate** but sufficient for small splats and boosts performance.

3DGS:Fast Differentiable Rasterizer for Gaussians - Forward Pass

Rasterization (Forward Pass):

- For each pixel in the tile, threads traverse the sorted list of Gaussians front-to-back.
- At each step i in the traversal:

$$\alpha_i = 1 - e^{-\sigma_i \delta_i}, \quad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$$

$$C = \sum_{i=1}^{N} T_i \alpha_i c_i, \quad \alpha_{\text{acc}} = 1 - \prod_{i=1}^{N} T_i$$

  where:
  - σ_i: opacity or density of the i^{th} Gaussian
  - δ_i: effective thickness (often set to 1)
  - c_i: color of the i^{th} Gaussian
- A thread **stops processing a pixel** when accumulated alpha saturates (e.g., $\alpha \approx 1$).
- Tile processing halts once all pixels are saturated.

3DGS: Fast Differentiable Rasterizer: Backward Pass

Differentiability (Backward Pass):

- To compute gradients, the full sequence of blended points per pixel must be recovered.
- This is achieved by **re-traversing the per-tile lists**, this time **back-to-front** . This avoids expensive dynamic memory management.
- Intermediate opacity values for gradient computation are **recovered** by storing the final accumulated opacity from the forward pass and dividing by each Gaussian's alpha during the back-to-front traversal.
- Numerical stability is maintained by skipping blending updates with very low alpha and clamping accumulated alpha.

3DGS: Advantages and Innovations Summarized

- **Real-Time Performance:** Achieves $\geq$ 30 fps at 1080p for novel view synthesis.
- **High Visual Quality:** Matches or surpasses state-of-the-art NeRF methods.
- **Fast Training:** Significantly faster training times compared to traditional NeRFs.
- **Explicit Representation:** Avoids costly neural network queries during rendering.
- **Adaptive Geometry:** Anisotropic Gaussians and density control allow detailed scene capture.
- **Efficient Differentiation:** Novel rasterizer design enables fast and memory-efficient backpropagation.

3DGS: Limitations - The Constraints of Rasterization

Despite its breakthroughs, the rasterization-based approach of 3DGS has inherent limitations:

- **Ideal Pinhole Camera Assumption:**
  - The EWA splatting formulation relies on linearizing the projection (Jacobian J), which is accurate for pinhole cameras but breaks down for highly distorted lenses (e.g., fisheye).
  - Deriving correct Jacobians for diverse and complex camera models is cumbersome and error-prone.
- **Time-Dependent Effects:**
  - Effects like rolling shutter, where different parts of the image are captured at slightly different times, are very hard to model within a single projective transform used by EWA splatting.
- **Secondary Lighting Effects:**
  - Rasterization is primarily a feed-forward process based on direct visibility.
  - It cannot easily simulate secondary rays needed for reflections, refractions, global illumination, or accurate shadows.
- **Antialiasing and Ordering:** While generally effective, tile-based sorting is an approximation of true per-pixel depth ordering, which can sometimes lead to minor artifacts. Sub-pixel Gaussians can also pose aliasing challenges.

Ray Tracing: The Fundamental Principle

- **Simulating Light Backwards:** Instead of simulating countless photons from light sources (forward ray tracing, or path tracing), most renderers trace rays *from the camera* into the scene (backward ray tracing).
- **For Each Pixel:**
  1. A **primary ray** (or camera ray, eye ray) is cast from the virtual camera's position, through the center of the pixel on the image plane, and into the 3D scene.
  2. The algorithm determines the **first object** this ray intersects.
  3. The properties of this object at the intersection point (material, texture, orientation) and its interaction with light sources determine the pixel's color.

Ray-Object Intersection Testing

A core component of ray tracing is finding intersections between rays and scene geometry.

- A ray can be parameterized as $P(t) = O + tD$, where O is the origin, D is the direction, and t is the distance.
- Objects are defined by mathematical equations (e.g., sphere, plane, triangle).
- **Intersection Test:** Substitute the ray equation into the object's equation and solve for t.
    - If $t > 0$, an intersection occurs.
    - The smallest positive t corresponds to the closest intersection point.
- Common primitives are triangles (for meshes) and spheres.

Acceleration Structures: Making Ray Tracing Practical

Testing every ray against every object in a complex scene is computationally prohibitive.

- **Acceleration Structures** are spatial data structures that organize scene geometry to quickly prune objects that a ray cannot possibly hit.
- **Bounding Volume Hierarchy (BVH):**
  - Objects are grouped into bounding volumes (e.g., boxes or spheres).
  - These volumes are organized hierarchically in a tree structure.
  - If a ray doesn't hit a parent bounding volume, it cannot hit any objects contained within it, allowing large parts of the scene to be skipped.
  - BVHs are widely used and highly effective.
- Other structures include k-d trees and uniform grids.

Shading and Secondary Rays: Achieving Realism

Once the closest intersection is found, shading determines its color. This often involves casting more rays:

- **Local Illumination:** Considers direct light from light sources.
    - **Shadow Rays:** Cast from the intersection point towards each light source. If a shadow ray hits another object before reaching the light, the point is in shadow from that light.
- **Global Illumination:** Accounts for light that has bounced off other surfaces.
    - **Reflection Rays:** For reflective surfaces, a new ray is cast in the mirror reflection direction. The color returned by this ray contributes to the surface's appearance.
    - **Refraction/Transmission Rays:** For transparent/translucent surfaces, a ray is cast through the material (potentially bending due to Snell's Law).
- This recursive process can continue for multiple bounces to simulate complex light interactions.

Types of Rays in a Ray Tracer

- **Primary Rays (Camera/Eye Rays):** Originate from the camera, one per pixel. Determine initial visibility.
- **Shadow Rays:** Test for direct line-of-sight to light sources. Typically boolean (hit/miss).
- **Reflection Rays:** Simulate mirror-like bounces from surfaces.
- **Refraction Rays (Transmission Rays):** Simulate light passing through transparent/translucent materials.
- **Ambient Occlusion Rays:** Sample the local hemisphere to determine how occluded a point is, affecting indirect ambient light.
- **Path Tracing Rays:** In path tracing (a more advanced form of ray tracing), rays are stochastically bounced around the scene to estimate the light transport equation, capturing complex global illumination effects.

Advantages of Ray Tracing

- **Photorealism:** Capable of simulating light physics accurately, leading to highly realistic images.
- **Natural Handling of Optical Effects:** Reflections, refractions, complex shadows, caustics, and global illumination are inherently supported.
- **Pixel-Perfect Accuracy:** No Z-fighting or other depth buffer artifacts common in rasterization.
- **Flexible Camera Models:** Easily supports non-pinhole cameras (fisheye, panoramic), depth of field, and motion blur by modifying ray generation.
- **Object Instancing Efficiency:** Multiple instances of an object can share geometry data, with only transformations differing, reducing memory.

Disadvantages and Challenges of Ray Tracing

- **Computational Cost:** Despite hardware acceleration, still more demanding than rasterization for simple scenes or primary visibility. The number of rays can grow exponentially with bounces.
- **Performance with Transparency:** Handling many layers of semi-transparent surfaces can be costly as rays need to continue through them, and order matters.
- **Noise in Stochastic Methods:** Path tracing and other Monte Carlo ray tracing techniques require many samples per pixel to converge to a noise-free image, or rely on denoising algorithms.
- **Acceleration Structure Overhead:** Building and maintaining BVHs (especially for dynamic scenes) has a cost.

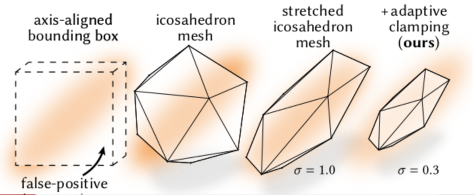
3DGRT: Bridging the Gap

Recognizing the limitations of 3DGS's rasterization, 3DGRT explores ray tracing 3D Gaussian scenes.

- **Motivation:**
    - Enable support for highly distorted cameras (e.g., fisheye in robotics).
    - Model time-dependent effects like rolling shutter.
    - Simulate secondary lighting effects (reflections, refractions, shadows).
    - Allow for stochastic ray sampling during training.
- **Core Idea:** Develop a specialized, efficient, and differentiable ray tracer for scenes represented by a large number of semi-transparent 3D Gaussian particles.

3DGRT: Efficiently Intersecting Rays with Gaussians

- **Problem:** Standard ray tracing is optimized for opaque, well-defined surfaces (like triangles). Gaussians are volumetric and semi-transparent.
- **3DGRT's Solution - Proxy Geometry:**
  - Each 3D Gaussian particle is encapsulated by a tight **bounding mesh primitive**:a stretched icosahedron).
  - This mesh acts as a proxy in the ray tracing acceleration structure (BVH).
  - Hardware-accelerated ray-triangle (forward facing triangle) intersection tests are used with these proxy meshes.
  - The proxy mesh is scaled anisotropically based on the Gaussian's covariance Σ and its opacity α, ensuring a tight fit. This is called "adaptive clamping."


axis-aligned bounding box
icosahedron mesh
stretched icosahedron mesh
+adaptive clamping (ours)
false-positive
σ = 1.0
σ = 0.3

3DGRT: The Stretched Icosahedron and Adaptive Clamping

Construction:

- Starts with a canonical icosahedron with a unit inner-sphere.
- Each vertex $\mathbf{v}$ is transformed by an **anisotropic rescaling** and the particle's pose:

$$\mathbf{v} \leftarrow \sqrt{2\log(\sigma/\alpha_{\text{min}})}\mathbf{S}\mathbf{R}^T\mathbf{v} + \boldsymbol{\mu}$$

- Here, $\boldsymbol{\mu}$ is position, $\mathbf{R}$ is rotation, and $\mathbf{S}$ is scaling from the particle's covariance $\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$.

Minimum Response Threshold (α_{min}):

- A specified minimum response value that the bounding primitive must capture: **0.01**.
- The scaling analytically computes the size needed to cover the region where the particle's volumetric density (ρ) is at least α_{min} of the maximum volumetric density.

Incorporating Opacity (σ) and Adaptive Clamping:

- Large, nearly-transparent particles (σ is small relative to α_{min}) have smaller bounding primitives than opaque particles of similar spatial size.
- This further reduces **false-positive intersections** for low-contribution particles,

3DGRT: Accelerated Ray Tracing Algorithm

Proposed Algorithm Overview:

- A **ray-gen program** initiates tracing a ray against the BVH containing bounding primitives.
- An **any-hit program** collects and maintains a sorted buffer of the indices of hit bounding primitives (up to a size k). Particle response is *not* evaluated here for efficiency.
- For each hit in the k-buffer, it retrieves the corresponding particle and computes its response and radiance.
- Radiance is accumulated, and transmittance is updated based on the particle's contribution.
- The ray is then effectively re-traced starting from the distance of the last processed hit.
- This repeats until all particles intersecting the ray have been processed or accumulated transmittance reaches a threshold.

Benefit: Processes intersections in a consistent depth order without missing particles or approximating transmittance, crucial for differentiability and accuracy.

3DGRT: Evaluating Particle Response

After a ray hits a particle's bounding primitive, we need to determine the particle's contribution. This is done by evaluating the particle's properties at a specific point along the ray.

Where to Sample?

- Simple methods (like projecting the particle center) work poorly for stretched, anisotropic particles.
- The proposed method analytically computes a single sample location, $\boldsymbol{\tau}_{\text{max}}$, along the ray $\mathbf{o} + \tau \mathbf{d}$ i.e. $\boldsymbol{\tau}_{\text{max}} = \max_\tau \rho(\mathbf{o} + \tau \mathbf{d})$

Maximum Response Point ($\boldsymbol{\tau}_{\text{max}}$):

- This is the point along the ray where the particle's volumetric density (ρ) is maximal.
- For a **Gaussian particle** with position $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the point of maximum response along the ray $\mathbf{o} + \tau \mathbf{d}$ is given by the parameter τ_{max}:

$$\tau_{\text{max}} = \frac{(\boldsymbol{\mu} - \mathbf{o})^T \boldsymbol{\Sigma}^{-1} \mathbf{d}}{\mathbf{d}^T \boldsymbol{\Sigma}^{-1} \mathbf{d}}$$

3DGRT: In action

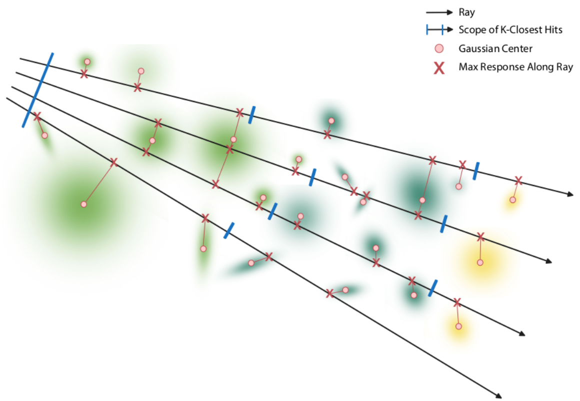


Fig. 5. **Rendering:** on each round of tracing, the next k closest hit particles are collected and sorted into depth order along the ray, the radiance is computed in-order, and the ray is cast again to collect the next hits.

3DGRT: Differentiable Ray Tracing and Optimization

For training, the ray tracer needs to be differentiable.

- **Backward Pass:**
  - Similar to the forward pass, rays are re-cast to sample the same set of particles in the same order.
  - Gradients of the rendering loss are computed with respect to particle parameters (position, covariance, opacity, SH coefficients).
- **Optimization Scheme Adaptation:**
  - The core 3DGS optimization logic (densification, pruning) is adapted.
  - A key change: uses 3D world-space gradients for densification criteria, as screen-space gradients are less meaningful or unavailable for general, incoherent rays.
  - The BVH must be rebuilt periodically as particle parameters change during optimization.

3DGRT: Enabling Complex Camera Models

Ray tracing's flexibility shines with non-standard cameras.

- **Distorted Lenses (e.g., Fisheye):**
  - Each ray is generated according to the specific camera's projection model, no matter how non-linear.
  - Allows training directly on images from distorted cameras, preserving all image information (unlike undistortion which crops pixels).
- **Rolling Shutter Effects:**
  - Common in fast-moving cameras where sensor rows are exposed sequentially.
  - 3DGRT can model this by generating rays where each ray (or group of rays corresponding to a scanline) has a slightly different camera pose (origin/orientation) corresponding to the capture time of that part of the sensor.
- This is critical for applications like robotics and autonomous vehicles.

3DGRT: Simulating Secondary Ray Effects

3DGRT unlocks advanced optical effects not feasible with 3DGS rasterization.

- **Reflections and Refractions:**
  - When a ray hits a reflective/refractive particle (or an inserted mesh with such properties), new reflection/refraction rays are cast into the Gaussian scene.
  - These secondary rays are traced through the same BVH of Gaussians.
- **Shadows:**
  - Shadow rays can be cast from intersection points towards light sources (if defined) or used to modulate contributions for artistic shadows.
- **Depth of Field:** Achieved by stochastic sampling of multiple rays per pixel, distributed over a lens aperture.
- **Object Instancing:** Multiple transformed copies of a group of Gaussians can be rendered efficiently by re-using their BVH subtree.

3DGRT: Generalized Gaussian Kernels

3DGRT explores particle kernels beyond the standard Gaussian.

- The ray tracing framework is not strictly tied to the Gaussian shape. As such other particle representations can be considered.
- **Generalized Gaussians:**

$$\hat{\rho}_n(x) = \sigma e^{-((x-\mu)^T \Sigma^{-1} (x-\mu))^n}$$

  - $n = 1$ is the standard Gaussian (after taking square root of exponent's argument). The paper uses n directly in the exponent, so their degree-2 is a sharper variant.
  - Higher degrees (e.g., $n = 2$ in the paper's formulation) create particles with sharper fall-offs.
- **Benefit:** Sharper particles can mean fewer particles contribute significantly along a ray (fewer "hits"), potentially improving rendering speed by reducing the number of evaluations.
- Other kernels like "kernelized surfaces" (flat Gaussians) are also explored.

3DGRT: Performance and Quality Insights

- **Visual Quality:** Generally achieves visual quality comparable to or slightly better than 3DGS rasterization on standard benchmarks.
- **Rendering Speed:**
  - For standard pinhole camera rendering of primary rays, 3DGRT is typically slower than 3DGS's highly optimized rasterizer (e.g., 2-5x slower per iteration during training, 3x slower for inference).
  - The overhead comes from BVH construction/traversal and more complex shading logic per ray.
  - However, it still achieves real-time or interactive rates for many scenes.
- **Key Advantage:** Enables capabilities simply not possible with rasterization, making the speed trade-off acceptable for those applications.
- Training with incoherent (randomly sampled) rays is possible but can be slower due to reduced coherence in memory access.

3DGUT: Motivation - Speed of Rasterization, Power of Generality

While 3DGRT offers generality via ray tracing, can we achieve similar benefits *within* a rasterization framework to retain maximum speed for primary views?

- **The Challenge with 3DGS's EWA Splatting:**
  - Relies on linearizing the projection function via Jacobians.
  - This approximation is poor for highly distorted cameras.
  - Difficult to derive Jacobians for every new camera model.
  - Hard to represent time-dependent effects (rolling shutter) with a single Jacobian.
- **Goal of 3DGUT:**
  1. Extend 3DGS to support arbitrary (distorted, time-dependent) camera models efficiently using rasterization.
  2. Align the rasterization formulation with ray-tracing to enable secondary ray effects (reflections, refractions) in a hybrid manner.

3DGUT: The Unscented Transform (UT) for Gaussian Projection

3DGUT replaces the EWA splatting's linearization of the *projection function* with an approximation of the *3D Gaussian particle* itself.

- **The Unscented Transform Idea:**
  - A technique from estimation theory (e.g., Unscented Kalman Filter).
  - Approximates a probability distribution (our 3D Gaussian) using a small, fixed set of deterministically chosen sample points called **sigma points**.
  - For an N-dimensional Gaussian, $2N + 1$ sigma points are typically used (e.g., 7 points for a 3D Gaussian).
  - These sigma points are chosen to capture the mean and covariance (and often higher moments) of the original Gaussian.

3DGUT: Sigma Point Generation

Step 1: Generate Sigma Points for Each 3D Gaussian

Given a Gaussian with mean $\mu \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times3}$, construct the sigma points:

$$x_0 = \mu$$

$$x_i = \mu + \sqrt{(3+\lambda)\Sigma_{[i]}} \quad \text{for } i = 1, 2, 3$$

$$x_i = \mu - \sqrt{(3+\lambda)\Sigma_{[i-3]}} \quad \text{for } i = 4, 5, 6$$

Here, $\Sigma_{[i]}$ denotes the i-th column of the matrix square root of Σ.

Corresponding weights:

$$w_0^{\mu} = \frac{3\lambda}{2(3+\lambda)}, \quad w_i^{\mu} = \frac{1}{2(3+\lambda)} \text{ for } i = 1, \ldots, 6$$

$$w_0^{\Sigma} = \frac{3\lambda}{2(3+\lambda)} + (1 - \alpha^2 + \beta), \quad w_i^{\Sigma} = \frac{1}{2(3+\lambda)} \text{ for } i = 1, \ldots, 6$$

UT parameters: $\lambda = \alpha^2(3+\kappa) - 3$, where typical values are $\alpha = 1e{-3}$, $\beta = 2$, $\kappa = 0$.

3DGUT: Nonlinear Projection and 2D Gaussian

Step 2: Project Sigma Points via Exact Camera Model
For each 3D sigma point x_i, apply nonlinear projection:

$$v_{x_i} = g(x_i)$$

where $g(\cdot)$ is the exact camera projection function (supports distortion, rolling shutter, etc.).
Step 3: Re-estimate 2D Gaussian in Image Plane
Use the projected points $\{v_{x_i}\}$ and weights $\{w_i\}$ to compute:

$$v_\mu = \sum_{i=0}^{6} w_i^\mu v_{x_i}$$

$$\Sigma' = \sum_{i=0}^{6} w_i^\Sigma (v_{x_i} - v_\mu)(v_{x_i} - v_\mu)^T$$

This 2D conic (v_μ, Σ') is then rasterized using the standard 3DGS pipeline.

3DGUT: Aligning Rasterization with Ray Tracing

Key Insight: The 2D conic computed via UT is used only for **tiling and culling**—to quickly identify which particles may affect which pixels. The actual particle response is computed independently in 3D, avoiding backpropagation through the non-linear projection.

1. **Evaluating Particle Response in 3D (Same as 3DGRT):**
   - Instead of evaluating 2D Gaussians at pixel centers, 3DGUT evaluates the original 3D Gaussian along the ray.
   - The evaluation point is the 3D location with maximum response along the ray, τ_{max}.
   - This avoids gradients through the 2D projection and improves stability.
2. **Depth Sorting for Accurate Alpha Blending:**
   - Alpha blending requires particles to be sorted along rays.
   - 3DGUT proposes using MLAB or k-buffer methods to track the top k particles per ray—improving consistency over tile-based sorting used in 3DGS.

3DGUT: Advantages of UT-based Projection

This UT-based projection offers significant benefits over EWA splatting:

- **No Jacobians Needed (Derivative-Free):** Completely avoids the complex and error-prone derivation of Jacobians for different camera models.
- **General Camera Model Support:**
  - Any camera projection function, no matter how non-linear or distorted, can be applied directly to the sigma points.
  - Seamlessly supports fisheye, panoramic, or other custom lenses.
- **Handles Time-Dependent Effects (Rolling Shutter):**
  - For rolling shutter, each sigma point x_i can be projected using a slightly different camera extrinsic matrix (pose) corresponding to the precise time its corresponding part of the scene would be scanned by the sensor.
- **Potentially Better Accuracy:** By transforming multiple points, UT can capture the non-linear transformation of the Gaussian's shape more accurately than a first-order linearization, especially under strong distortion.
- **Retains Rasterization Core:** The underlying rendering paradigm is still rasterization, aiming to preserve its speed advantages.

3DGUT: Key Contributions and Advantages

- **Generalized Rasterization for Complex Cameras:** Extends 3DGS to arbitrary camera models (fisheye, rolling shutter) using the Unscented Transform, without needing per-model Jacobians.
- **Unified Rendering Formulation:** Aligns particle evaluation and sorting with ray-tracing principles.
- **Hybrid Rendering for Secondary Effects:** Enables simulation of reflections and refractions by effectively using ray tracing thus enabling secondary rays within the same scene representation.
- **Efficiency:** Aims to maintain rendering rates comparable to 3DGS for primary views due to its rasterization core, while being more flexible.
- Outperforms dedicated methods on datasets with distorted cameras.

3DGUT: Implementation and Training Details

- Built upon PyTorch with custom CUDA kernels.
- Employs advanced culling strategies.
- Most optimization parameters (learning rates, loss) are kept consistent with 3DGS for fair comparison.
- UT parameters (α, β, κ) are set (e.g., $\alpha = 1.0, \beta = 2.0, \kappa = 0.0$).
- Densification and pruning logic adapted: Since 2D screen space gradients aren't the primary driver for projection quality, 3D positional gradients (similar to 3DGRT) are used, divided by distance to camera.
- Loss: Weighted sum of L2 and SSIM ($L = L_2 + 0.2 L_{SSIM}$).

The Evolution: From 3DGS to 3DGRT and 3DGUT

- **3DGS (Foundation - Fast Rasterization for Pinhole):**
  - *Strength:* Real-time, high-quality rendering for standard views.
  - *Innovation:* Anisotropic Gaussians, efficient tile-based EWA splatting rasterizer, adaptive density control.
  - *Limitation:* Restricted to pinhole cameras; no easy support for secondary rays or complex time-dependent effects.
- **3DGRT (Generality via Ray Tracing):**
  - *Strength:* Natively handles complex cameras, secondary rays, and arbitrary ray distributions.
  - *Innovation:* Efficient BVH construction for Gaussians (proxy meshes), specialized k-buffer ray tracing algorithm, differentiable tracing pipeline.
  - *Trade-off:* Generally slower than pure rasterization for primary views.
- **3DGUT (Advanced Rasterization + Hybrid):**
  - *Strength:* Brings complex camera support to rasterization; enables secondary rays via hybrid approach, aiming for rasterization speed for primary views.
  - *Innovation:* Unscented Transform for robust projection, aligning 3D particle evaluation with tracing.
  - *Goal:* "Best of both worlds" - speed and extended capabilities.

Addressing Specific Limitations of 3DGS

- **Limitation 1: Constrained to Ideal Pinhole Cameras / Difficulty with Distortions**
  - **3DGRT's Approach:** Abandons rasterization's projective assumptions. Each ray is generated independently according to the true camera model, however complex.
  - **3DGUT's Approach:** Replaces EWA splatting's Jacobian linearization with the Unscented Transform. UT can handle arbitrary non-linear projection functions by transforming sigma points exactly.
- **Limitation 2: Difficulty with Time-Dependent Effects (e.g., Rolling Shutter)**
  - **3DGRT's Approach:** Each ray can have its own unique origin and direction based on the precise capture time for that sensor line/pixel.
  - **3DGUT's Approach:** Each sigma point can be projected using a different camera extrinsic corresponding to its effective capture time.
- **Limitation 3: Lack of Support for Secondary Lighting Effects (Reflections, Refractions)**
  - **3DGRT's Approach:** Naturally supports these by casting additional reflection/refraction rays into the Gaussian scene and tracing them.
  - **3DGUT's Approach:** Achieves this via hybrid rendering. Primary rays are rasterized. At points of intersection requiring reflection/refraction, secondary rays are generated and traced.

Comparing Methodologies: Rasterization vs. Ray Tracing vs. Hybrid

- **3DGS (Pure Rasterization):**
  - *Pros:* Extremely fast for primary visibility with pinhole cameras. Highly optimized pipeline.
  - *Cons:* Inflexible for camera models, no inherent secondary effects.
- **3DGRT (Pure Ray Tracing):**
  - *Pros:* Maximum flexibility for camera models, true secondary effects, arbitrary ray patterns.
  - *Cons:* More computationally intensive than rasterization for primary views, BVH overhead.
- **3DGUT (UT-Rasterization + Hybrid Tracing):**
  - *Pros:* Fast primary views like rasterization, but with support for complex cameras (via UT). Secondary effects possible via tracing secondary rays.
  - *Cons:* UT projection adds some overhead compared to simple EWA. Hybrid rendering still requires a tracing component. Approximation quality of UT for highly distorted Gaussians can be a factor.

Particle Evaluation: 2D vs. 3D

A key shift in approach from 3DGS to 3DGRT/3DGUT:

- **3DGS (EWA Splatting):**
  - Projects 3D Gaussian to a 2D conic on the image plane.
  - Evaluates the color/opacity contribution based on this 2D projected shape relative to pixel centers.
  - Gradients flow back through the (linearized) projection.
- **3DGRT 3DGUT (for aligned rendering):**
  - Identify candidate particles that might affect a pixel/ray.
  - Evaluate the original **3D Gaussian particle's response** directly in 3D space at a specific point along the camera ray (typically τ_{max}, the point of maximum response).
  - This avoids propagating gradients through the potentially complex/approximated projection step when calculating particle contribution, which can be more stable and accurate.
  - For 3DGUT, the UT-projected 2D conic is still used for *culling and tiling* (determining which particles affect which pixels efficiently), but not for the final response evaluation if aligning with tracing.

Synergies and Shared Components

Despite their differences, these methods share and build upon common foundations:

- **Core Representation:** All three use 3D Gaussian particles (position, covariance, opacity, SH color) as the fundamental scene primitive.
- **Optimization Framework:** All rely on differentiable rendering and stochastic gradient descent to optimize Gaussian parameters from input images.
- **Adaptive Density Control:** The concepts of cloning, splitting, and pruning Gaussians are central to achieving high-quality representations in all three methods, though the specific triggers (e.g., screen-space vs. world-space gradients) might differ.
- **Hardware Acceleration:** All are designed to leverage GPU parallelism, with custom CUDA kernels for performance-critical parts.

Concluding Thoughts

- **3D Gaussian Splatting** revolutionized real-time radiance field rendering with its explicit representation and fast rasterizer.
- **3D Gaussian Ray Tracing** significantly expanded the capabilities by introducing efficient ray tracing for Gaussian scenes, enabling complex camera models and secondary lighting effects.
- **3D Gaussian Unscented Transform** cleverly brings many of these advanced camera capabilities to a rasterization framework and enables hybrid rendering, striking a balance between speed and feature richness.
- These works collectively represent major strides towards achieving the goal of truly photorealistic, real-time, and flexible 3D scene synthesis.

Thank You

Thank You Questions?