How Gaussian Splatting works

Gaussian Splatting renders a 3D scene by representing each point as a 3D anisotropic Gaussian, projected and rasterized directly without meshing.
Each point is represented as a 3D Gaussian blob with

- **Mean (position in 3D)**
- **Covariance (orientation, scale, shape)**
- **Color**
- **Opacity (alpha)**

Rendering Pipeline in detail:

1. Scene Representation

A dense 3D scene is stored as a collection of many many Gaussians instead of triangles or point sprites.

2. Camera Projection

Each 3D Gaussian is projected into screen space (2D) using the view and projection matrices, just like any other 3D object in a graphics pipeline.

3. Depth Sorting

The Gaussians are sorted back-to-front (like in transparency rendering), so they blend correctly when rendered.

4. Rasterization / Splatting

Each Gaussian becomes a soft, blurry, elliptical splat on the screen. The 2D footprint of the 3D Gaussian is determined based on the camera angle, distance, and covariance.

Instead of rendering one pixel per point (like with classical point clouds), each splat covers a region of the screen, which is where the term *splatting* comes from.

5. Alpha Blending (Compositing)

Each splat contributes **color and opacity** to the screen buffer using **alpha blending**:

```
final_color = alpha * splat_color + (1 - alpha) * background
```

Training Workflow overview:

We get as input images from different viewpoints and camera variables, like focal length, position, orientation and more. We then need to get an initial estimate of 3D geometry and estimate the camera poses through ‘structure from motion’. We continue by assigning initial colors from image projection and default shape and size to the splats. Those splats then get rendered in a differentiable way and projected onto a 2D image plane. We then compare the rendered image to the real image using a specific loss function and then use backpropagation along with gradient descent to train the gaussian parameters. We repeat this learning across all views for multiple epochs to get good parameter values.

Classical Point Cloud Rendering – In Principle:

In this technique, a 3D scene is represented as a **set of discrete points**, each with:

- Position `(x, y, z)`
- Color `(r, g, b)`
- Optional: Normals, size, etc.

Each point is treated as a small, screen-space primitive (like a square pixel or round sprite), and rendered directly to the screen.

Rendering Pipeline:

For the full details I refer to my Task 1 report.

1. Vertex Data

You load a point cloud from a file (like `.pts`, `.ply`, or `.xyz`) which contains all the 3D point coordinates (and optionally color, normals). Each point is sent as a vertex to the GPU.

2. Projection

Each point is:

- Transformed by the Model-View-Projection (MVP) matrix
- Mapped to screen space just like a mesh vertex

3. Rasterization

Instead of rasterizing triangles, we rasterize GL_POINTS. Each point gets drawn as:

- A pixel (by default)
- A square or round "sprite" if `gl_PointSize` is set like we did in task 1.

4. Fragment Shader, if specified

Can be used to:

- Shade the point (e.g., with lighting based on normals)
- Discard fragments to make round points (via `gl_PointCoord`)
- Color points

5. Blending

Optional transparency or additive blending (often not used). No deep blending or adaptive density like in Gaussian Splatting.

Key Differences: Gaussian Splatting vs Classical Point Clouds

Feature	Classical Point Cloud	Gaussian Splatting
Primitive	Fixed-size point sprites	Elliptical 3D Gaussians
Appearance	Flat, discrete	Smooth, overlapping, blended
Surface Continuity	Sparse, holey look	Smooth appearance, feels more like a surface
Rendering Approach	GL_POINTS or raster primitives	Ray/fragment-based alpha blending
Training / Input	Raw sensor data	Learned from images or optimized
Depth Compositing	Basic depth test	Depth-aware alpha blending

Potential Advantages of Gaussian Splatting:

Much smoother and more photo-realistic rendering. Gaussian splats overlap and blend, which fills holes and creates a continuous and smooth surface appearance. Lighting and color look softer and more natural, especially with view-dependent effects. You can tweak individual Gaussians (e.g., for editing or refinement). Alpha blending with proper depth sorting provides soft but effective occlusions.

Potential Disadvantages of Gaussian Splatting:

Higher complexity and resource usage. Gaussian splats store more data per point (position, orientation, scale, color, opacity). Rendering requires custom shaders, sorting, and blending, which can be slower or more memory-intensive than simple `GL_POINTS`.

The Advantages of Gaussian Splatting Over Traditional Point Clouds

Feature	Gaussian Splatting	Classic Point Cloud Rendering

Smooth Appearance	Yes (blended ellipsoids)	Often noisy or sparse
Adaptive Detail	Yes (bigger Gaussians far away)	Fixed point size
Real-Time Rendering	Highly optimized	Slower at high resolution
Occlusion / Transparency	Better with blending	Harder to manage

Analyze the quality of the visualization

- **What types of artifacts can you detect?**

- Blurry and smeared regions.
- Huge visible blobs in areas where little information is given.
- Floating or misplaced (and duplicate) streaks and ghosting.
- Color bleeding, especially in transparent or over-exposed regions.
- Glowing “halo” effects around certain edges or objects (e.g., horns).
- Some areas are noisy and less dense (visible gaps).
- The whole background is black and often leaks through the point cloud.

What could be the reason for the artifacts?

1. **Sparse or uneven input image coverage:**
   - Some parts of the model (like the top of the skull or background) have fewer observations from different angles (or none), leading to poor reconstruction or insufficient Gaussian density.
2. **Incorrect camera calibration or pose estimation:**
   - If some input camera poses are slightly off, Gaussian centers and orientations can be misaligned, causing ghosting or floating artifacts.
3. **Illumination differences in input images:**
   - Strong lighting changes or reflections across photos can cause inconsistent color blending or light halos when encoded in the Gaussians.
4. **Low-contrast or overly uniform backgrounds (e.g. plain white walls, skies) can make it harder for the model to:**
   - Differentiate the foreground from the background

    - Localize object boundaries
    - Receive strong gradients during optimization, especially in regions with little texture or structure
5. **Objects moving in the scene (are in different positions depending on foto)**
6. **Blurry or out of focus images**

Compare the artifacts with the classical point rendering

Aspect	**Classical Point Cloud**	**Gaussian Splatting**
Artifacts	Sparse, jagged holes	Ghosting, streaking, blurring
Cause	Not enough points	Overlapping Gaussians or misfits
Appearance	More “pointy” and raw	Smoother but potentially distorted

Aspect	**Classical Point Cloud Rendering**	**Gaussian Splatting**
Typical Artifacts	- Sparse holes - Jagged edges - Disconnected surfaces - Flickering from missing points	- Ghosting (duplicate/misplaced detail) - Floating splats - Streaking or color bleeding - Overblending (loss of fine structure)
Root Cause	- Not enough points - No interpolation - No surface estimation - Gaps between points due to viewing angle	- Overlapping Gaussians misaligned or oversized - Depth sorting or alpha blending errors - Inaccurate position/opacity updates - View-dependent inconsistencies

Appearance	- Sharp but "raw" looking - Clear point edges - Abrupt transitions between sampled areas and holes	- Smooth and soft visuals - May appear blurry or “melty” - Can produce continuous surfaces that look *too* smooth or dreamy
Perceived Sharpness	- Crisp when close - Very pixelated when far - No blending or smoothing	- Soft at all distances - May glow around edges - Tends to blur fine details
Behavior with Occlusion	- Poor occlusion handling (flat or unrealistic depth perception) - Points simply stack in depth - No alpha blending or depth ordering - Points leak through nearby objects -No sorting — just dumps all points	- Proper occlusion requires sorted alpha blending - Wrong order causes ghosting and halos -Uses alpha blending and depth sorting to simulate visibility -Must render Gaussians back-to-front to simulate occlusion properly
Lighting Interaction	- Flat appearance - No view-dependent effects - No shading or highlights	- View-dependent color and shading possible - May create unintended reflections or "light leaks"
Temporal Stability (e.g., videos)	- Usually stable unless point positions change - Simple rendering pipeline	- May flicker, ghost, or trail if camera poses or splat parameters shift slightly

Response to Low Texture / Uniform Areas	- Points simply missing (white space) - No interpolation	- Gaussians may be misplaced or overblended - Boundaries get fuzzy, model may "bleed" into background

Use-case Implications

Scenario	Better with...
High precision tasks (e.g. geometry inspection)	Point cloud (crisp, accurate)
Realistic rendering or immersive view synthesis	Gaussian splatting (continuous, blended)
Low input quality or few views	Point cloud may fail completely; splatting degrades gracefully but with artifacts
Artistic or stylized applications	Gaussian splats can look painterly or soft

How could the model be improved by adjusting the photos and how they are taken?

Densify the Camera Coverage

- Covers more of the object to reduce ghosting, floating blobs, and gaps.
- Take more photos from a wider range of angles, especially:
    - From above and below
    - Around occluded parts (like horns, back of head, inner gaps)
- Ensure overlap between consecutive camera positions.
- Avoid big angle jumps (like jumping from front to back without sides)

Improve Lighting Consistency

- Reduces color mismatches, glowing halos, and blending artifacts.
- Use soft, even lighting (like a ring light or cloudy daylight).
- Avoid moving shadows, reflections, or changes in exposure between shots.
- Use consistent camera settings.

Higher Image Resolution & Better Focus

- Improves texture detail and splat sharpness.
- Use a high-resolution camera (DSLR or good phone camera).
- Make sure all images are in sharp focus, especially near edges or details.
- Avoid motion blur or shaky shots (use a tripod if needed).

Ensure Accurate Camera Calibration

- Reduces alignment errors that cause ghosting or floating artifacts.
- Use a strong Structure-from-Motion (SfM) pipeline.
- Make sure features are detected across many images.
- Run bundle adjustment to refine all camera poses.
- Avoid photos with featureless backgrounds or motion blur, which break pose estimation.