Evolution of 3D Scene Representations

From Features to Neural Fields to Gaussians

Jugal Alan

M.Tech AI, IIT Ropar

2024AIM1004

Outline: The Evolution Journey

1. **Distinctive Image Features from Scale-Invariant Keypoints (2004):** SIFT
2. **Structure-from-Motion Revisited (2016):** Modern 3D reconstruction
3. **NeRF (2020):** Neural radiance fields for photorealistic rendering
4. **Instant-NGP (2022):** Real-time neural graphics with hash encoding
5. **3D Gaussian Splatting (2023):** Explicit primitives for real-time rendering

Key Question: How did we move from geometric pipelines to learned, real-time 3D representations?

Paper 1: SIFT - Scale-Invariant Feature Transform

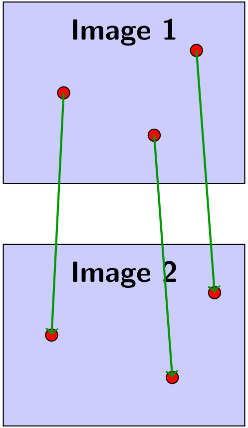
David Lowe, IJCV 2004

Core Idea:

- Detect and describe local features
- Invariant to scale, rotation, illumination
- Enable wide-baseline matching

Impact:

- Backbone for SfM/MVS pipelines
- Enabled 3D reconstruction from images
- Foundation for modern feature matching



● Keypoints, → Matches

SIFT: Technical Pipeline

1. Scale-Space Detection (DoG) → 2. Keypoint Localization → 3. Orientation Assignment → 4. Descriptor (128-D)

Key Components:

- **Difference of Gaussians (DoG):** Efficient scale-space extrema detection
- **Orientation histogram:** Achieves rotation invariance
- **Gradient-based descriptor:** 4×4 grid of 8-bin histograms $=$ 128 dimensions
- **Matching:** Nearest-neighbor with ratio test to reject ambiguous matches

Why it matters: Reliable correspondences across viewpoints $\rightarrow$ enables 3D reconstruction

Paper 2: Structure-from-Motion Revisited

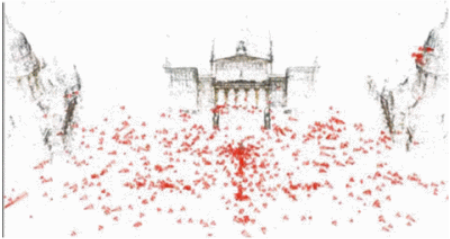
Schönberger & Frahm, CVPR 2016

Core Idea:

- Modern, robust incremental SfM
- Scales to thousands of images
- Produces camera poses + sparse 3D

Output:

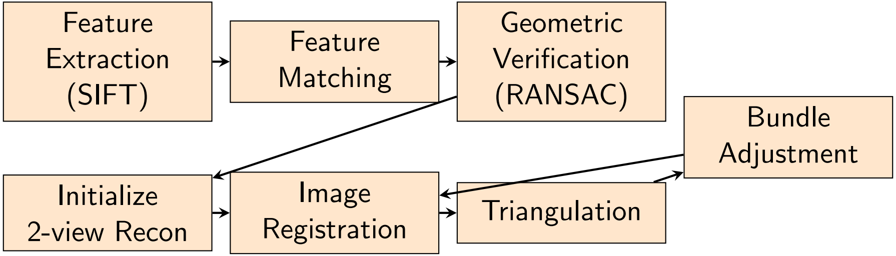
- Camera extrinsics & intrinsics
- Sparse 3D point cloud
- Foundation for COLMAP system



Impact:

- De facto standard for pose estimation
- Provides input for NeRF training

SfM: Incremental Reconstruction Pipeline



Bundle Adjustment: Joint optimization minimizing reprojection error

$$\min_{\{\mathbf{R}_i,\mathbf{t}_i\},\{\mathbf{X}_j\}} \sum_{i,j} \rho\left(\|\pi(\mathbf{R}_i\mathbf{X}_j + \mathbf{t}_i) - \mathbf{x}_{ij}\|^2\right)$$

Robustness: RANSAC outlier rejection, next-best-view selection, redundant view mining

Transition: From Geometry to Neural Rendering

Traditional: SfM + MVS

✓Accurate geometry
✓Explicit structure

✖ Sparse reconstruction
✖ Limited photorealism
✖ Difficult novel views

$\Rightarrow$

Neural: NeRF

✓Photorealistic rendering
✓Continuous representation
✓Novel view synthesis

✖ Slow training/rendering
✖ Requires posed images

Key insight: SfM provides the camera poses that NeRF needs. Now we can move from geometric reconstruction to learned radiance fields!

Paper 3: NeRF - Neural Radiance Fields

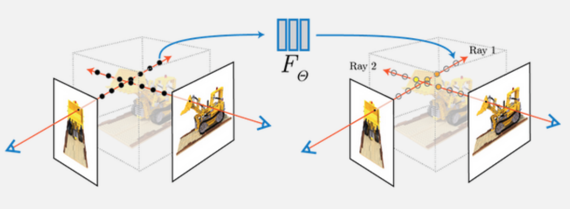
Mildenhall et al., ECCV 2020

Core Idea:

- Represent scene as continuous 5D function
- Map $(\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$

Revolutionary aspects:

- Photorealistic novel views
- No explicit geometry needed
- Continuous representation



Limitations:

- Slow training (hours-days)
- Slow rendering (seconds/frame)
- Per-scene optimization

NeRF: Technical Details

Input: Posed images (from SfM/COLMAP)

Representation: MLP $F_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$

- $\mathbf{x} = (x, y, z)$: 3D position
- $\mathbf{d} = (\theta, \phi)$: viewing direction
- σ: volume density (geometry)
- $\mathbf{c} = (r, g, b)$: emitted color (appearance)

Volume Rendering Equation:

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot \mathbf{c}(\mathbf{r}(t), \mathbf{d}) \, dt$$

where transmittance $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s)) ds\right)$

- Maps low-dimensional inputs to high-frequency features
- Enables MLP to represent fine details

NeRF: Training and Results

Training:

- Per-scene optimization (no generalization)
- Photometric loss: $\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{C}(\mathbf{r}) - \mathbf{C}_{gt}(\mathbf{r})\|_2^2$
- Hierarchical sampling: coarse + fine networks
- Training time: hours to days per scene

Quality:

- State-of-the-art photorealistic novel view synthesis
- Captures complex view-dependent effects (reflections, specularities)
- Continuous representation allows arbitrary resolution

Bottleneck: Rendering requires querying MLP hundreds of times per ray $\rightarrow$ seconds per frame

Question: Can we keep the quality but make it fast?

Müller et al., SIGGRAPH 2022

Core Idea:

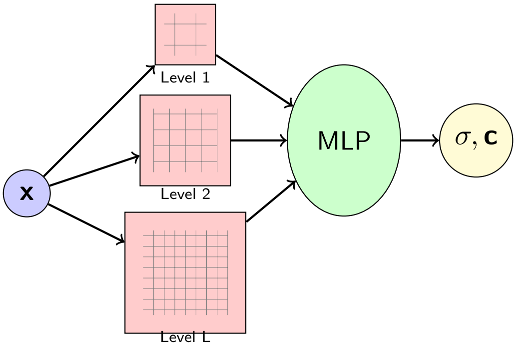
- Replace large MLPs with hash encoding
- Store features in compact hash tables
- Small MLP for final prediction

Speed gains:

- Train in seconds to minutes
- Real-time rendering (interactive fps)
- $1000\times$ faster than original NeRF

Impact:

- Enabled practical applications
- Foundation for many follow-ups



Multiresolution Hash Encoding

Instant-NGP: Hash Encoding Explained

Key Innovation: Multiresolution hash grids replace positional encoding + small MLP

How it works:

1. Map 3D position $\mathbf{x}$ to multiple resolution levels $L_1, L_2, \ldots, L_L$
2. At each level: hash $\mathbf{x}$ to compact table, retrieve feature vector
3. Concatenate all features + $\mathbf{x} \rightarrow$ feed to tiny MLP (2 hidden layers, 64 neurons)
4. MLP outputs density σ and color $\mathbf{c}$

Hash function: $h(\mathbf{x}) = \left(\bigoplus_{i=1}^{d} x_i \pi_i\right) \bmod T$

- T: hash table size (e.g., 2^{19})
- π_i: large prime numbers

Why it's fast:

- Hash lookups are $O(1)$ and cache-friendly
- Small MLP $\rightarrow$ fewer parameters, faster inference
- No expensive positional encoding computation

Instant-NGP: Performance Comparison

Method	**Training Time**	**Rendering Speed**	**Quality (PSNR)**
Original NeRF	1-2 days	30 sec/frame	31.0 dB
Instant-NGP	5-10 min	60+ fps	33.2 dB

Impact:

- Made NeRF practical for interactive applications
- Training time: days $\rightarrow$ minutes ($\sim$**1000$\times$ speedup**)
- Rendering: seconds $\rightarrow$ real-time ($\sim$**100$\times$ speedup**)
- Quality maintained or improved

$\boxed{\textit{From research curiosity to practical tool}}$

Limitation: Still implicit representation $\rightarrow$ volume rendering $\rightarrow$ can we go faster with explicit primitives?

Paper 5: 3D Gaussian Splatting

Kerbl et al., SIGGRAPH 2023

Core Idea:

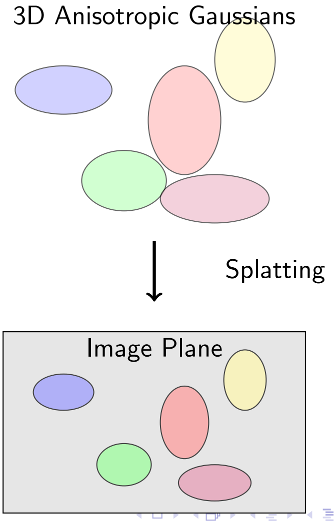
- Explicit 3D anisotropic Gaussians
- Differentiable splatting rasterization
- Adaptive density control

Breakthrough:

- Real-time rendering (¿30 fps at 1080p)
- Competitive quality with best NeRFs
- Editable, streamable representation

Advantages:

- Explicit primitives
- Memory efficient
- Enables editing/streaming



3D Gaussian Splatting: Representation

Each Gaussian is parameterized by:

- **Position:** $\boldsymbol{\mu} \in \mathbb{R}^3$
- **Covariance:** $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$ (anisotropic, full 3D ellipsoid)
- **Opacity:** $\alpha \in [0, 1]$
- **Color:** Spherical harmonics coefficients for view-dependent appearance

3D Gaussian function:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Covariance representation: $\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$

- **R**: rotation (quaternion)
- **S**: scaling (3D vector)
- Ensures positive semi-definite covariance

Initialization: From SfM point cloud (COLMAP output)

3D Gaussian Splatting: Rendering Pipeline

3D Gaussians in World Space → Project to 2D (Perspective) → Sort by Depth (Visibility) → Alpha Blending (Splatting)

Final Image

Key steps:

1. **Projection:** Transform 3D covariance to 2D via Jacobian of perspective projection

$$\mathbf{\Sigma}' = \mathbf{J}\mathbf{W}\mathbf{\Sigma}\mathbf{W}^T\mathbf{J}^T$$

2. **Tile-based rasterization:** Sort Gaussians by depth per screen tile (GPU-friendly)
3. **Alpha compositing:** Front-to-back blending

$$\mathbf{C} = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

3D Gaussian Splatting: Optimization

Loss function: $\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$

Adaptive density control: Key to quality

- **Clone:** Large Gaussians in under-reconstructed areas
- **Split:** Large Gaussians with high gradients (need more detail)
- **Prune:** Gaussians with low opacity ($< \epsilon_\alpha$)
- Applied every N iterations during training

Training time: 30 minutes on RTX 3090 (vs days for NeRF)

Rendering speed: Real-time at 1080p (¿30 fps)

Method	Quality (PSNR)	Rendering Speed
Mip-NeRF 360	27.7 dB	0.3 fps
Instant-NGP	25.5 dB	6 fps
3D Gaussian Splatting	**27.2 dB**	**138 fps**

3D Gaussian Splatting: Trade-offs

Advantages:

- Real-time, high-quality rendering
- Explicit representation $\rightarrow$ editable, streamable
- Fast training (30 min vs days)
- Memory efficient for bounded scenes
- Enables applications: VR, games, robotics

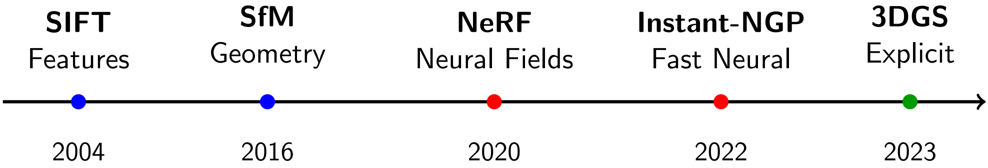
Limitations:

- Gaussian count can grow (millions for complex scenes)
- Less naturally handles unbounded scenes than some NeRF variants
- Explicit primitives may have artifacts at boundaries
- Still requires SfM preprocessing (camera poses)

Extensions:

- 4D Gaussians for dynamic scenes (CVPR 2024)
- Gaussian Splatting SLAM for real-time mapping (CVPR 2024)

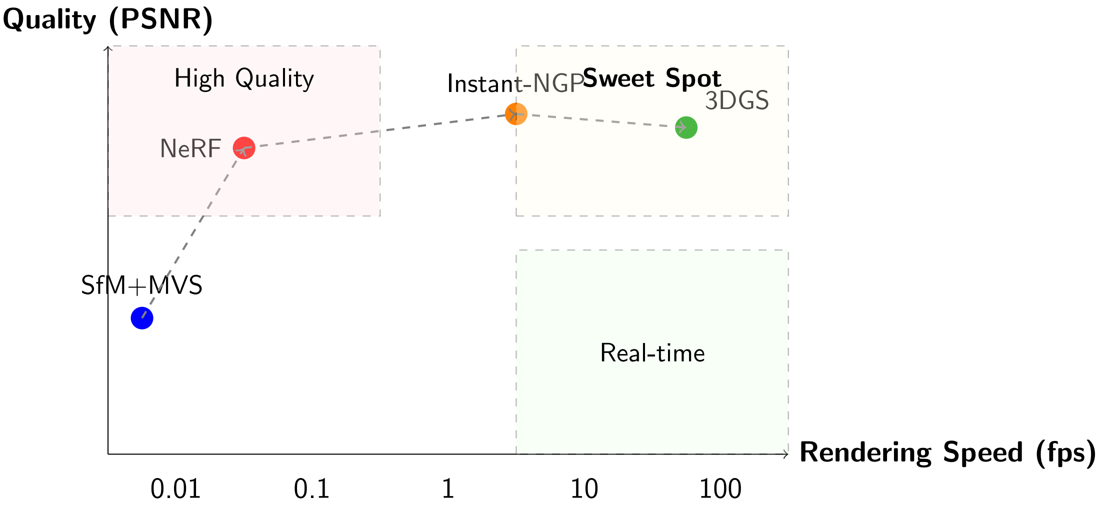
Evolution Timeline: The Full Picture



Key transitions:

- Geometric era: Features + optimization $\rightarrow$ accurate but sparse
- Neural implicit era: Continuous fields $\rightarrow$ photorealistic but slow
- Explicit learned era: Optimized primitives $\rightarrow$ real-time + quality

Quality vs Speed: The Trade-off Space



When to Use Each Method?

Method	Use When...	Avoid When...
SIFT + SfM	Need accurate geometry, camera calibration, sparse 3D	Need photorealistic rendering or dense reconstruction
NeRF	Prioritize quality over speed, offline rendering, research	Need real-time or interactive rates
Instant-NGP	Need balance of quality and speed, interactive preview	Need absolute best quality or real-time deployment
3DGS	Real-time rendering, VR/AR, games, robotics/SLAM	Extremely large unbounded scenes, limited memory

Practical recommendation:

Future Directions & Applications

Active research areas:

- **Generalization:** Move from per-scene to single model (e.g., pixelNeRF, generalizable NeRFs)
- **Dynamic scenes:** 4D Gaussians, D-NeRF, Neural Scene Flow
- **Large-scale:** City-scale reconstruction, satellite imagery
- **Sparse inputs:** Few-shot reconstruction, single image
- **Semantic understanding:** Combine with segmentation, object-level editing

Real-world applications:

- **Robotics/SLAM:** Gaussian Splatting SLAM, real-time mapping
- **VR/AR:** Immersive environments, photorealistic avatars
- **Film/VFX:** Virtual production, view synthesis
- **E-commerce:** Product visualization, virtual try-on
- **Cultural heritage:** Digital preservation, virtual museums

Summary: Key Takeaways

1. **SIFT:** Local features enable wide-baseline matching
2. **SfM:** Geometric reconstruction produces camera poses + sparse 3D
3. **NeRF:** Neural radiance fields achieve photorealistic novel views
4. **Instant-NGP:** Hash encoding brings NeRF to interactive speeds
5. **3DGS:** Explicit Gaussians enable real-time with quality

The evolution in one sentence:

From geometry to neural implicit to explicit learned primitives

Questions?

References

1. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*.
2. Schönberger, J. L., & Frahm, J. M. (2016). Structure-from-motion revisited. *CVPR*.
3. Mildenhall, B., et al. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. *ECCV*.
4. Müller, T., et al. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH*.
5. Kerbl, B., et al. (2023). 3D Gaussian splatting for real-time radiance field rendering. *SIGGRAPH*.