References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural Point-Based Graphics. *ECCV*, 2020. 2

[2] Sai Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Fredo Durand, Aaron Lefohn, and Yong He. SLANG.D: Fast, Modular and Differentiable Shader Programming. *SIGGRAPH Asia*, 2023. 5

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *ICCV*, 2021. 2

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR*, 2022. 4, 5, 7

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. *ICCV*, 2023. 2, 3, 4

[6] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient Geometry-Aware 3D Generative Adversarial Networks. *CVPR*, 2022. 2

[7] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Courier Corporation, 1960. 3

[8] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *Visualization and Computer Graphics*, 2024. 2

[9] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. *CVPR*, 2023. 2

[10] Jaehoon Choi, Yonghan Lee, Hyungtae Lee, Heesung Kwon, and Dinesh Manocha. MeshGS: Adaptive Mesh-Aligned Gaussian Splatting for High-Quality Rendering. *ACCV*, 2024. 2

[11] Mike Cyrus and Jay Beck. Generalized Two- and Three-Dimensional Clipping. *Computers & Graphics*, 1978. 3, 2

[12] B Delaunay. Sur la Sphère Vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles*, 1934. 3

[13] Herbert Edelsbrunner. An acyclicity theorem for cell complexes in d dimensions. *Symposium on Computational geometry*, 1989. 2, 3

[14] Tim Elsner, Victor Czech, Julia Berger, Zain Selman, Isaak Lim, and Leif Kobbelt. Adaptive Voronoi NeRFs. *arXiv:2303.16001*, 2023. 2

[15] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *CVPR*, 2022. 2

[16] Lin Gao, Jie Yang, Bo-tao Zhang, Jia-mu Sun, Yu-jie Yuan, Hongbo Fu, and Yu-kun Lai. Real-time Large-scale Deformation of Gaussian Splatting. *ACM Transactions on Graphics (TOG)*, 2024. 2

[17] Xiangjun Gao, Xiaoyu Li, Yiyu Zhuang, Qi Zhang, Wenbo Hu, Chaopeng Zhang, Yao Yao, Ying Shan, and Long Quan. Mani-GS: Gaussian Splatting Manipulation with Triangular Mesh. *CVPR*, 2025. 2

[18] Joachim Georgii and Rudiger Westermann. A Generic and Scalable Pipeline for GPU Tetrahedral Grid Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2006. 3

[19] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant Foam: Real-Time Differentiable Ray Tracing. *ICCV*, 2025. 2, 5, 6, 7, 8

[20] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. MILo: Mesh-In-the-Loop Gaussian Splatting for Detailed and Efficient Surface Reconstruction. *SIGGRAPH Asia*, 2025. 2

[21] Minghao Guo, Bohan Wang, Kaiming He, and Wojciech Matusik. TetSphere Splatting: Representing High-Quality Geometry with Lagrangian Volumetric Meshes. *arXiv:2405.20283*, 2024. 3

[22] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized Exponential Splatting for Efficient Radiance Field Rendering. *CVPR*, 2024. 2

[23] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 2018. 5, 7

[24] Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. *arXiv:2505.19175*, 2025. 2, 7

[25] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. *CVPR*, 2025. 5

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. *SIGGRAPH 2024*, 2024.

[27] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable Radial Kernel Splatting. *CVPR*, 2025. 2

[28] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping Edges in Triangulations. *Symposium on Computational Geometry*, 1996. 1

[29] Michael S Karasick, Derek Lieber, Lee R. Nackman, and VT Rajan. Visualization of Three-Dimensional Delaunay Meshes. *Algorithmica*, 1997. 2, 3

[30] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *CVPR*, 2018. 2

[31] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (TOG)*, 2023. 1, 2, 5, 6, 7

[32] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-Scale

Scene Reconstruction. *ACM Transactions on Graphics (TOG)*, 2017. 5, 7

[33] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient Sphere-Based Neural Rendering. *CVPR*, 2021. 2

[34] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3D-HGS: 3D Half-Gaussian Splatting. *arXiv:2406.02720*, 2024. 2

[35] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 2018. 2

[36] Ancheng Lin, Yusheng Xiang, Paul Kennedy, and Jun Li. Direct Learning of Mesh and Appearance via 3D Gaussian Splatting. *arXiv:2405.06945*, 2024. 2

[37] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable Beta Splatting. *SIGGRAPH*, 2025. 2

[38] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG*, 2021. 2

[39] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. *International Conference on Motion in Games*, 2016. 2, 8

[40] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T Barron, and Yinda Zhang. EVER: Exact Volumetric Ellipsoid Rendering for Real-time View Synthesis. *ICCV*, 2025. 2, 5, 6, 7

[41] Nelson Max, Pat Hanrahan, and Roger Crawfis. Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions. *Workshop on Volume Visualization*, 1990. 2, 3

[42] J. L. Meijering. Interface area, edge length, and number of vertices in crystal aggregates with random nucleation. *Philips Research Reports*, 1953. 2

[43] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *ECCV*, 2020. 1, 2

[44] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *SIGGRAPH Asia*, 2024. 2, 7

[45] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 2007. 8

[46] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (TOG)*, 2022. 2, 4

[47] Ken Museth and Santiago Lombeyda. TetSplat Real-Time Rendering and Volume Clipping of Large Unstructured Tetrahedral Meshes. *IEEE Visualization*, 2004. 3

[48] Udo Pachner. Pl homeomorphic manifolds are equivalent by elementary shellings. *European Journal of Combinatorics*, 1991. 1

[49] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians. *CVPR*, 2024. 2

[50] Haoxuan Qu, Zhuoling Li, Hossein Rahmani, Yujun Cai, and Jun Liu. DisC-GS: Discontinuity-aware Gaussian Splatting. *NeurIPS*, 37, 2024. 2

[51] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering, 2024. 2

[52] Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Transactions on Graphics (TOG)*, 2024. 2

[53] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising Densification in Gaussian Splatting. *ECCV*, 2024. 4

[54] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *CVPR*, 2016. 3

[55] Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. *CVPR*, 2024. 2

[56] Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. Sparse Voxels Rasterization: Real-time High-fidelity Radiance Field Rendering. *CVPR*, 2025. 2, 5, 6, 7

[57] Thibault Tricard. Interval shading: using mesh shaders to generate shading intervals for volume rendering. *Computer Graphics and Interactive Techniques*, 2024. 3, 5

[58] Nicolas von Lützow and Matthias Nießner. LinPrim: Linear Primitives for Differentiable Volumetric Rendering. *arXiv:2501.16312*, 2025. 3

[59] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. GaMeS: Mesh-Based Adapting and Modification of Gaussian Splatting. *arXiv:2402.01459*, 2024. 2

[60] Jiepeng Wang, Yuan Liu, Peng Wang, Cheng Lin, Junhui Hou, Xin Li, Taku Komura, and Wenping Wang. GausSurf: Geometry-Guided 3D Gaussian Splatting for Surface Reconstruction. *arXiv:2411.19454*, 2024. 2

[61] P Wang, L Liu, Y Liu, C Theobalt, T Komura, and WP Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Neurips*, 2021. 2

[62] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end View Synthesis from a Single Image. *CVPR*, 2020. 2

[63] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based Neural Radiance Fields. *CVPR*, 2022. 2

[64] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *NeurIPS*, 2021. 2

[65] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron,

and Ben Mildenhall. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. *SIGGRAPH*, 2023. 2

[66] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Transactions on Graphics (ToG)*, 2024. 2

[67] Ziyu Zhang, Binbin Huang, Hanqing Jiang, Liyang Zhou, Xiaojun Xiang, and Shunhan Shen. Quadratic Gaussian Splatting for Efficient and Detailed Surface Reconstruction. *arXiv:2411.16392*, 2024. 2

[68] Chengwei Zheng, Lixin Xue, Juan Zarate, and Jie Song. GauSTAR: Gaussian Surface Tracking and Reconstruction. *CVPR*, 2025. 2

Radiance Meshes for Volumetric Reconstruction

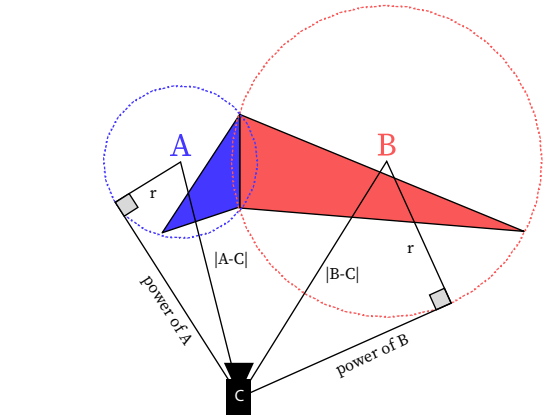
Supplementary Material

7. Smoothness Proof Sketch

A Delaunay triangulation can be formed by taking a triangulation and then flipping the edges of triangles that are not Delaunay (e.g. the edges that have a circumsphere containing a vertex) [28]. In 3D, these edge-flips are called Pachner moves [48]. As discussed in Sec. 4.2, during optimization we make take a valid Delaunay triangulation and then move the vertices by some small amount ϵ such that a vertex may enter the circumsphere of a neighboring tetrahedron, thereby triggering a topological flip. This implies that the vertex was already nearby the circumsphere boundary prior to the move. Consequently, these flips must involve sets of points that are nearly cospherical, as the geometric configurations before and after the flip share an almost identical circumsphere.

8. Sorting Proof Sketch

To demonstrate how a Delaunay triangulation can be visibility-sorted relative to a specific viewpoint using the power of circumcircles, consider two adjacent triangles. Geometrically, the edge shared by these two triangles lies along the radical axis (the line of intersection) of their respective circumcircles. This axis is the line of equal power; it divides the plane into two regions. On one side, the viewpoint has a lower power distance to the first circumcircle, but on the other side it has a lower power distance to the second. Since the relative depth order of the two triangles flips only when the viewpoint crosses this axis, the power distance serves as a consistent sorting criterion. This implies that sorting the primitives by the power of the viewpoint relative to their circumcircles yields the correct visibility order. This property naturally generalizes to 3D, where the shared face of two tetrahedra lies on the radical plane. See Figure 9 for a visualization.



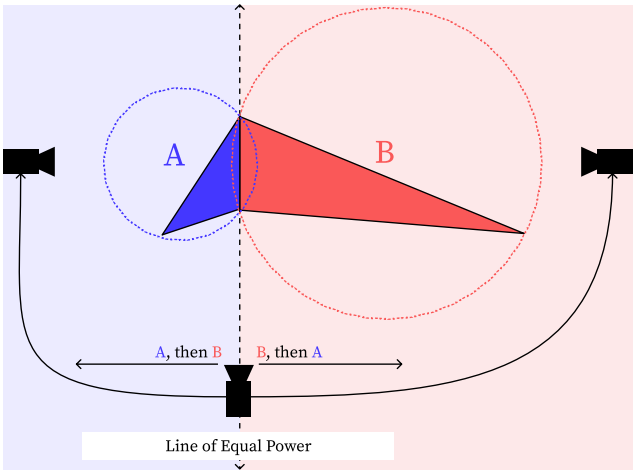


Figure 9. To sort two adjacent triangles relative to a viewpoint, we need only look at their circumcircles. The radical axis (the line along which the power of a spheres is equal) goes through the two intersection points between the circles, and divides the circumscribed triangles. Consider the centers of two circumcircles A and B and a viewpoint C located along the radical axis. Since $|A-C|$ increases as the camera moves right and $|B-C|$ decreases, the power of B is less than the power of A to the right, and the reverse is true to the left. This extends to non-intersecting circles, and can be used for non adjacent triangles. Combining this observation with the the Delaunay empty sphere property allows us to sort all triangles.

9. Details

Even though we are using GPU acceleration, recomputing a Delaunay triangular from scratch is still somewhat slow. As such, we only update vertex locations and their corresponding Delaunay topology every 10 iterations. On a scene with 1.2 million tetrahedra, our timings during training are approximately 19 milliseconds for rendering, 13 milliseconds to query the instant-NGP, 143 milliseconds for the backwards pass, and 92 milliseconds to compute the Delaunay triangulation.

Because densification changes the circumcenters of the primitives and therefore their colors, we temporarily increase the learning rate of some of the model to allow it to rapidly accommodate for this change. After densification at iteration I, we increase the learning rate of the instant NGP weights and the vertex coordinates by adding a "spike" function with duration L on top of the decay func-

tion $l(i)$:

$$\mathbb{1}\{i - I > 0\}\,(l(0) - l(I))\exp\left(-\frac{6(i-I)}{L}\right) \tag{16}$$

This increases the decaying learning rate back up to its maximum (initial) value, before decaying it back down in L iterations.

During optimization we follow Zip-NeRF and regularize our model using the normalized L_2 weight decay on Instant-NGP grid.

9.1. Algorithm Details

The naive version of our shader would, for each of the 4 vertices of each tetrahedra, load all of the tetrahedra information and attached it each of the vertices. Then, when the fragment shader for any given triangle is called, the information from the closest vertex would be obtained, and the full Cyrus-Beck algorithm [11] would be run on the rest of the tetrahedra to compute the back intersection, which then allows us to compute the integral along the intersection with the primitive.

In our optimized implementation, we leverage the capabilities of a mesh shader to allow the four vertices of each tetrahedra to share information between themselves. This reduces the number of loads by a factor of four. We also optimize the Cyrus-Beck algorithm: in the vertex shader we precompute the intersection distance with each of the vertices, and then perform interpolation between these distances to find the intersection distance with each plane. This interpolation must be perspective-correct barycentric interpolation to ensure that the interpolation results are correct. A similar technique can be applied to the color gradient to further simplify the fragment shader. This reduces runtime cost to only 12 FLOPs. See Algorithm 1.

10. Results

See Figure 10 for additional results.

Algorithm 1 Shader code to perform accelerated intersections using wave intrinsics within a mesh shader.

Mesh shader intersection pre-calculation

```
uint bli = (WaveGetLaneIndex() / 4) * 4;
#define WRTV(x, id) WaveReadLaneAt(x, bli + id)

static const uint4 kTetTriangles[4] = {
    uint4(0, 2, 1, 3),
    uint4(1, 2, 3, 0),
    uint4(0, 3, 2, 1),
    uint4(3, 0, 1, 2),
};

tri = kTetTriangles[tetVertexId];

// outward facing normal
n = cross(
    WRTRV(vertex, tri[2]) -
    WRTRV(vertex, tri[0]),
    WRTRV(vertex, tri[1]) -
    WRTRV(vertex, tri[0]));

v = WRTRV(vertex, tri[0])
num = dot(n, v - rayOrigin);

o.planeN = float4(
    WRTRV(num, 0),
    WRTRV(num, 1),
    WRTRV(num, 2),
    WRTRV(num, 3) );
o.planeD = float4(
    dot(WRTRV(n, 0), o.rayDir),
    dot(WRTRV(n, 1), o.rayDir),
    dot(WRTRV(n, 2), o.rayDir),
    dot(WRTRV(n, 3), o.rayDir) );
return o;
```

Fragment shader intersection routine

```
d = length(rayDir);
o.planeD /= d;

all_t = o.planeN / o.planeD;

t_enter = max(o.planeD > 0.0f ? all_t : -FLT_MAX)
    ;
t_exit  = min(o.planeD < 0.0f ? all_t : FLT_MAX);
```

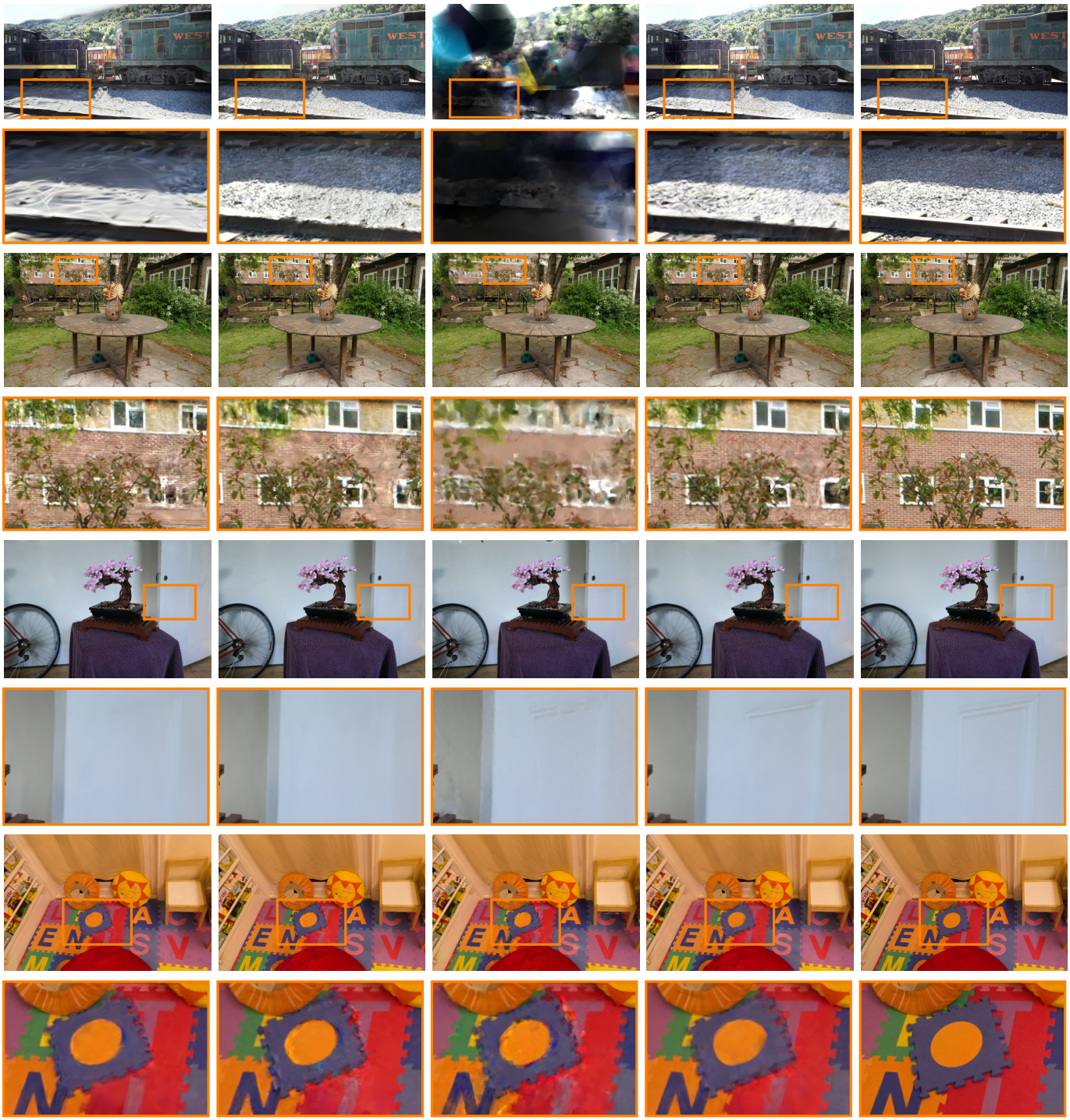


Figure 10. Visual comparison of our method alongside baseline algorithms on various scenes from the datasets we use.