Converting 2D Images to Geospatial 3D Models Using Generative AI

Tim Woodard, Brent Bartlett, Zoë LaLena, Everett Spackman, Chris Holland
NVIDIA
Santa Clara, CA
twoodard@nvidia.com, bbartlet@nvidia.com, zlalena@nvidia.com, espackman@nvidia.com, cholland@nvidia.com

ABSTRACT

The volume of geospatial imagery has increased significantly in recent years through the access of easy-to-use collection devices such as cellphones and automated airborne platforms. Commercial drone technology allows users to collect high-resolution geo-tagged images of outdoor objects from multiple perspectives quickly and at a relatively low cost. A commonly desired goal within many industries is to rapidly convert these 2D images into 3D models that accurately represent a large area of interest and can be used in existing visualization tools. Although semi-automated processes and photogrammetry methods have helped to reduce the time required to produce useful results compared to manual 3D modeling, they still often require significant processing time and do not scale well to large areas.

Generative AI, a modern branch of deep learning, produces synthetic output that conforms to the pattern of a training dataset. Within the domain of generative AI, neural radiance fields (NeRFs) enable the synthesis of novel views based on a collection of overlapping 2D images. In addition, the trained AI model can be utilized to produce extremely detailed 3D meshes through the process of neural surface reconstruction. The resulting mesh can be converted into traditional open standards, enabling use of this data in a wide range of applications.

This paper describes a process that utilizes GPU-acceleration of traditionally CPU-bound algorithms combined with generative AI-based methods, enabling the rapid conversion of 2D images into geospatial 3D models which can then be represented in the 3D Tiles format. We will also demonstrate visualization in a real-time ray tracing environment capable of interactively rendering extremely complex scenes. Finally, we will discuss modifications to the workflow that resulted in order-of-magnitude speedups, the challenges and limitations of processing geospatial imagery using these methods and cover current developments likely to impact the future performance, fidelity, and scalability of this process.

ABOUT THE AUTHORS

Tim Woodard is a Senior Solutions Architect with NVIDIA's Professional Visualization group, focusing on advanced rendering technology, virtualization, XR, AI, and Omniverse for MS&T. Tim has over 25 years of experience designing and developing architectures for real-time simulation visual systems for medical, driving, and flight training. He has received patents for run-time geospatial database generation technology.

Brent Barlett, Ph.D. is a Senior Solutions Architect at NVIDIA focusing on bringing cutting edge Generative AI and Accelerated Geospatial technology to bear across the complex problem sets faced by the Public Sector. He holds a Doctorate in Imaging Science from the Rochester Institute of Technology and enjoys exploring the intersection of AI and Remote Sensing and opening new use cases with accelerated computing.

Zoë LaLena is an intern at NVIDIA, pursuing a Master of Science in Computer Vision from Carnegie Mellon University. She holds a Bachelor of Science in Imaging Science from the Rochester Institute of Technology. Zoë has a background in historical document imaging, remote sensing, and machine learning.

Everett T. Spackman is a Solutions Architect at NVIDIA. He has a background in spatial analysis, remote sensing, and autonomous vehicle mapping systems. Everett holds a Bachelor of Science in Anthropology and Geography from the California Polytechnic State Institute, San Luis Obispo, and a Master of Science in Spatial Data Science from the Pennsylvania State University.

Chris Holland is a Senior Solutions Architect at NVIDIA for Public Sector, focusing on Generative AI and Computer Vision. He holds a Master of Science in Geospatial Data Science from University of Missouri and a Master in Anthropology from University of Texas.

Converting 2D Images to Geospatial 3D Models Using Generative AI

Tim Woodard, Brent Bartlett, Zoë LaLena, Everett Spackman, Chris Holland
NVIDIA
Santa Clara, CA
twoodard@nvidia.com, bbartlet@nvidia.com, zlalena@nvidia.com, espackman@nvidia.com, cholland@nvidia.com

INTRODUCTION

Many missions across the modeling, simulation, and training (MS&T) community revolve around obtaining an up-to-date 3D model of an area of interest. This process typically involves transforming source data into an accurate visual representation of the area to derive actionable insights. From the complex landscape of available geospatial tools, we consider two primary techniques in this paper: novel view synthesis and surface mesh extraction. With novel view synthesis, the goal is to interactively render a scene in full 3D fidelity from any viewpoint. This provides an operator with the ability to quickly understand the layout of different objects in a scene from viewpoints that may not have been explicitly collected. Surface mesh extraction is a useful technique for creating a polygonal digital asset of a scene that can then be stored in industry-standard formats like 3D Tiles, which is widely supported by commonly used industry software applications that enable further analysis, dissemination, and visualization. This work considers the task of creating workflows suitable for both novel view synthesis and surface mesh extraction from sets of overlapping photographs taken from the air, as shown in Figure 1.

Figure 1. Example surface mesh extracted from a series of overlapping drone images of Safety Park, a first responder training facility in Atlanta, GA. This is viewed within the popular open-source tool Blender and is of suitable quality for creating a digital twin of the facility.

The community has created a proven set of techniques for accomplishing detailed surface mesh extraction through traditional photogrammetric means. The heart of this approach relies on finding the same points across several overlapping images and using these image correspondences to solve for these locations in world coordinates. Several processing steps typically follow to create a final surface mesh. While this approach is a proven workhorse for the community, we explore utilizing relatively recent techniques as viable replacements: Neural Radiance Field (NeRF), 3D Gaussian Splatting (3DGS), and accelerated mesh processing. Our primary motivation is to explore what results can be achieved and to implement potential speed-ups of the overall processing pipeline.

Traditional Pipeline Overview

Photogrammetry is a technique that uses overlapping images, often collected from an airborne platform, to record the positions of matching features and calculate these points in three-dimensional space. For a typical flight sortie, the number of matching points across all images can be quite large and allows one to solve for camera pose, camera model, and a set of sparse points in 3D space. This information is useful for initializing both NeRF and 3DGS techniques as well. There is a robust ecosystem of tools to accomplish solving for camera location, pose, and model that include commercial offerings and open-source projects. From these offerings that produce accurate camera and sparse point cloud information, we leverage COLMAP (Schonberger and Frahm, 2016) as our structure from motion solution. There are also many methods available to perform a dense reconstruction from the initial sparse set of points. In general, these approaches interpolate between the initial sparse points to better approximate the surface geometry present in a scene. This can then be transformed into a surface mesh and textured using source imagery to produce a visually appealing 3D product suitable for visualization or additional processing.

Accelerated Pipeline Overview

In recent years, work on radiance field rendering has expanded, producing many new techniques to derive highly accurate and robust visualizations of real-world scenes. These techniques hold the promise of replacing portions of the traditional photogrammetric pipeline with methods that can produce high quality results with fewer input images. One seminal approach that sparked new interest in neural-based rendering is called the neural radiance field or NeRF (Mildenhall et al., 2021).

NeRF utilizes a multilayer perceptron (MLP) as a fully connected neural network capable of reconstructing a 3D scene from a set of 2D images and camera poses (location and viewing direction). The MLP is optimized such that, when given a 3D point and viewing direction, it produces the view-dependent radiance (color) and volume density at that point, trained to match the input images when rendered. These MLPs are scene-specific, such that for a new set of images and camera poses, a new NeRF must be created by training a new MLP. To render a NeRF scene, the MLP must be prompted repeatedly with sample 3D points along the viewing direction. The resulting colors and densities are then used to create the volume rendering. A visualization of the full NeRF pipeline is shown in Figure 2.

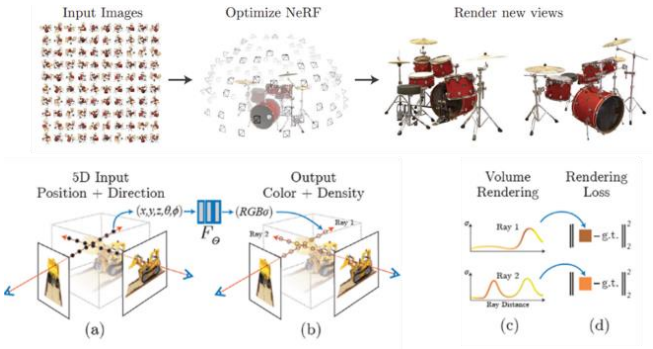

Figure 2. Process outlined in the seminal NeRF approach, turning input images into a rendered scene. The process of optimizing the MLP is illustrated in the bottom row of plots.

While the original NeRF implementation (Mildenhall et al., 2021) took at least 12 hours to train a scene, Muller et al. decreased the space and time required to train a NeRF in their method, InstantNGP (Müller et al., 2022). InstantNGP trains on the order of seconds to minutes due to multiresolution hash encoding and its implementation based on CUDA (Compute Unified Device Architecture). CUDA (NVIDIA CUDA, 2024) is a C-like programming language that enables software to leverage the massively parallel architecture of a modern GPU (graphics processing unit), which can have over fifteen thousand processing cores per chip. This enables orders-of-magnitude speedups for algorithms like NeRF. The memory required to train a NeRF with InstantNGP is log linearly related to the resolution, allowing for larger scenes to be trained. In addition, several innovative approaches have been proposed that allow for larger-scale NeRFs to be trained using multiple GPUs. One such method is NeRF-XL, which allows for arbitrarily large scenes. The more GPUs available, the larger the scene that can be trained (Li et al., 2024).

The tradeoff between speed and quality in NeRF-based approaches led to a new class of methods: Gaussian Splatting. Gaussian Splats use an explicit representation, not requiring the time-consuming ray tracing required for implicit representations like NeRFs. Gaussian Splatting methods use a large quantity of 3D Gaussian functions, or splats, to represent a 3D scene (Kerbl et al., 2023). These Gaussians have optimizable positions, sizes (anisotropic covariance), opacities, and spherical harmonics (viewing direction dependencies). During training, Gaussians are added and removed from the scene as necessary, allowing the density and number of Gaussians to be optimized as well. Gaussian Splatting, like NeRF methods, requires 2D images and camera positions. Additionally, Gaussian Splatting requires an initial point cloud from a structure from motion (SfM) method, such as COLMAP. This sparse point cloud is used to initialize the centers of the splats. The Gaussian Splatting optimization flow is shown in Figure 3.

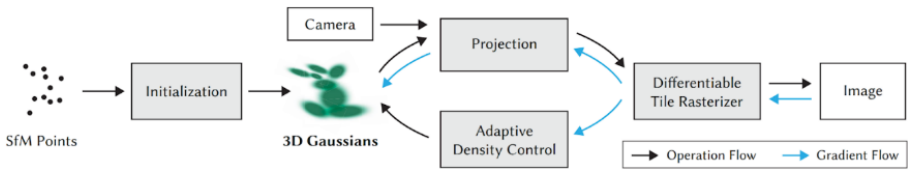

Figure 3. The process for initializing and optimizing 3D Gaussians. (Kerbl et al., 2023)

Generating novel views with NeRFs or Gaussian Splats is very useful for many tasks, but it is also often desirable to extract a surface mesh from a trained model. Once a NeRF or Gaussian Splat is converted into a mesh it can be displayed and manipulated with pre-existing tools including traditional Image Generators (IGs) and applications based

on popular game-engines. Several methods have been created to take the output from Gaussian Splatting and convert it to a mesh. One such method is 2D Gaussian Splatting (2DGS) (Huang et al., 2024). 2DGS utilizes Gaussian planar discs rather than 3D Gaussians. These flat Gaussians better model the thin nature of surfaces, allowing for estimation and optimization of the surface normals to get accurate depths. The depths are used to reconstruct the surface and produce a mesh.

Neuralangelo is an adaption of Instant NGP (neural graphics primitives) that additionally implements numerical gradients, coarse to fine optimization, and uses a signed distance function (SDF) representation of 3D surfaces (Li et al., 2023). Rather than optimizing the density and color of points, the MLP in Neuralangelo optimizes an SDF in one MLP and color in another. Neuralangelo recovers high quality, and accurate 3D representation of both indoor and outdoor scenes at extremely high fidelity using long training iterations. Since the scene is represented as an SDF, the surface of the scene has an implicit representation from which an isosurface mesh can be extracted via an accelerated marching cubes implementation, as shown in Figure 4.

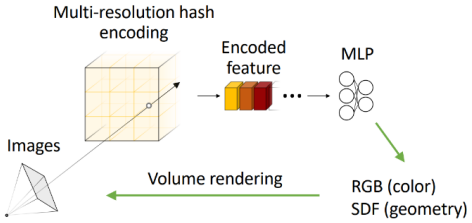

Figure 4. Neuralangelo architecture for training neural radiance field. (Li et al., 2023).

Extracting an isosurface boundary mesh from Neuralangelo via marching cubes that retains fine grain surface details results in a mesh that is extremely dense and utilizes vertex-based color. For such meshes to be practical for use in standard MS&T 3D workflows, we have defined a process to convert the output from Neuralangelo into 3D Tiles. This requires first removing minor mesh artifacts, parameterizing the 3D coordinates into 2D image-space, transferring vertex colors to a texture image corresponding to the parameterization, and finally performing mesh decimation. The result can then be converted to 3D Tiles and streamed into a wide range of MS&T applications.

ACCELERATED PIPELINE DETAILS

For this work, we identify several key steps for transforming 2D source images into 3D content that is ready for use. Figure 5 outlines the high-level pipeline steps which depend sequentially on each other. For practitioners that only require viewing of the 3D scene, the pipeline can end at the QuickView stage. If further analysis is required, all stages can be completed to produce results in the final 3D Tiles format.

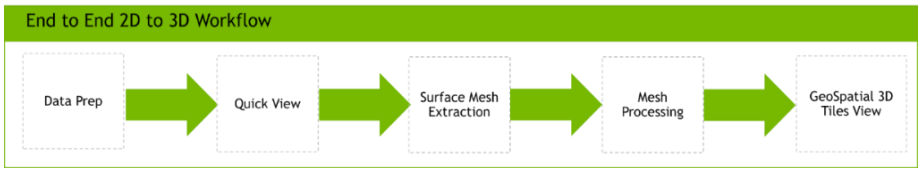

Figure 5. End-to-End pipeline illustrating each stage required to transform 2D images to 3D geospatial tiles.

We will illustrate each step of our pipeline in more detail in the following sections with results derived from a dataset collected with an inexpensive consumer drone, focusing on NeRF-based implementations for QuickView and surface mesh extraction.

To collect the sample data that we used for this process, we flew a commercial-off-the-shelf drone (a DJI Mini Air 2) to collect 4K video in a ~two-minute orbit that was 240m in diameter. The collection area was “Safety Park” where first responders train - an ideal test since it contains many different objects within a compact area. All results shown in the following sections are derived from this dataset, shown in Figure 6.

Figure 6. Example images extracted from done orbit video for Safety Park scene.

Data Preparation Stage

For this work, we assume that the process starts with source imagery that has minimal motion blur, limited atmospheric effects (haze, etc.) and that has sufficient overlap. In addition, angular coverage spanning 360 degrees of a scene is required to derive full 3D reconstruction. The guidelines for creating NeRF datasets, available in the Instant NGP repository's documentation, offer valuable starting points for beginners. The NeRF dataset tips highlighted at https://github.com/NVlabs/instant-ngp/blob/master/docs/nerf_dataset_tips.md are a good place to start. Our exploration has shown that a single orbit of a scene from an aircraft or drone with ~50-200 images is sufficient for good visual reconstruction. We also have early indications that traditional photogrammetric mapping flights also produce high quality results (e.g., five camera bundles capturing in front, back, left, right, down configurations).

Figure 7 breaks our data preparation pipeline into stages that align well with the COLMAP workflow. We extract SIFT (scale-invariant feature transform) features targeting ~16,000 features per 4K drone image with an example feature match shown in Figure 8. Bundle adjustment solves for camera intrinsic, pose, and sparse reconstruction points, which is visualized in Figure 9. Finally, we generate the necessary inputs for either NeRF or 3DGS. This process completes exceptionally quickly for scenes of the scale of Safety Park since both feature extraction and matching make use of GPU acceleration within the COLMAP tool. Additionally, bundle adjustment is rapid for smaller datasets of this type with the entire data preparation process taking ~10 minutes on a single-GPU desktop workstation.

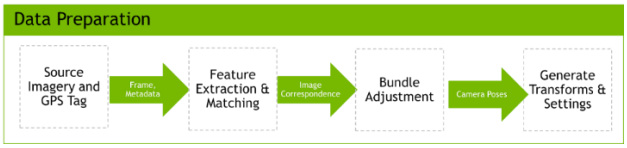

Figure 7. Data preparation pipeline detail stages used to generate required information for subsequent tasks.

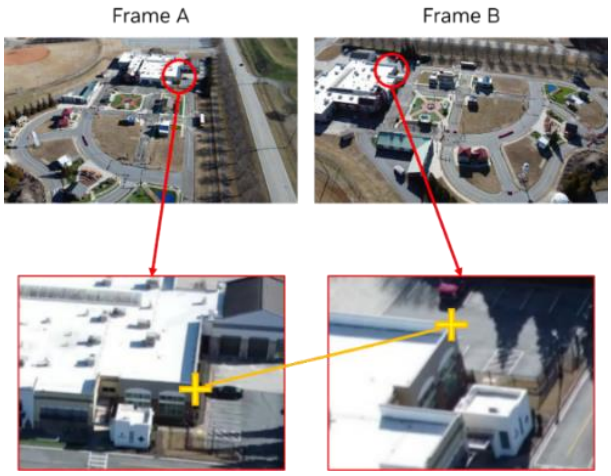

Figure 8. Two frames used in our reconstruction with matching locations (yellow +) of the same building corner taken from different view angles.

QuickView Stage

Once the input imagery has been appropriately processed, a "QuickView" product can be generated. Instant NGP is an ideal tool for smaller datasets like our Safety Park scene. Figure 10 shows the Instant NGP GUI training and rendering at the same time to create a 3D visualization of the scene. The training process creates an acceptable quality model within only a few seconds and provides a representation of the scene that could be used for many missions within the MS&T community.

While the main objective is to illustrate the ability to generate 3D content, one can also render static 2D views of the 3D scene. We use this insight to create a "top down" view of the QuickView scene using an orthographic camera type. This type of virtual camera renders the scene with parallel rays pointing straight down. Geospatial practitioners call this an "orthomosaic." This top-down image can then be tagged with the geo-coordinates derived from the source GPS locations and placed on a traditional 2D map. Figure 11 shows this orthomosaic overlaid on top of an existing mapping product. Note that an advantage of this approach is that no artifacts such as the tipping of tall structures due to image parallax or seam lines between images exist producing a very high quality 2D mapping product.

Figure 9. Visualization of bundle adjustment process running in the COLMAP GUI. Each camera location (red triangles) is being refined and placed showing our circular flight path around the scene with corresponding sparse reconstruction (colored dots).

Figure 10. Generating a QuickView product using InstantNGP to train and display a NeRF of the Safety Park scene. Initial training (top left), converging model (top right), and two new views of the scene (bottom left / right) all occurring in a few seconds.

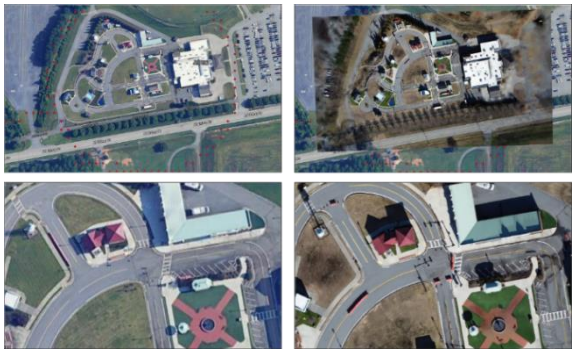

Figure 11. Safety Park scene with pre-existing image basemap and drone locations overlaid with red dots (top left), orthomosaic rendered from QuickView NeRF (top right), zoom in of town square (bottom left) and updated zoom in with NeRF based render (bottom right). Notice the absence of tilted buildings / structure in bottom right NeRF based orthomosaic.

Surface Mesh Extraction Stage

Once the data preparation stage is completed and a satisfactory quick view is generated, we then proceed to extracting a surface mesh. In this example, we utilize the project Neuralangelo to train a model on the Safety Park dataset and extract an isosurface via the marching cubes algorithm provided by the tool. Figure 12 shows extracted surface meshes at various checkpoints during the training process. We see that the model begins to converge rather rapidly with most large-scale structures emerging within the first 25,000 iterations. Fine-grain details and structures emerge last with a good visual representation by 300,000 iterations.

We were able to achieve eleven iterations per second running on a single NVIDIA H100 300W PCIe GPU, with substantial opportunity to further increase runtime performance via CUDA kernel optimizations of the existing PyTorch implementation. Additional performance tuning is possible via hyperparameter optimization.

After the Neuralangelo model has been trained, an isosurface can be extracted via a GPU-accelerated marching cubes implementation. Marching cubes utilizes a set of 256 pre-defined triangle configurations representing the possible ways in which the boundary of an isosurface might intersect a cube (Lorensen et. al., 1987). A virtual 3D grid of vertices is mapped over the extent of the model, and the vertices are evaluated against the isosurface to determine if they fall inside or outside of the SDF. With the eight vertices of each cube evaluated, a configuration is selected, resulting in a triangle mesh that approximates the boundary of the isosurface, shown in Figure 13.

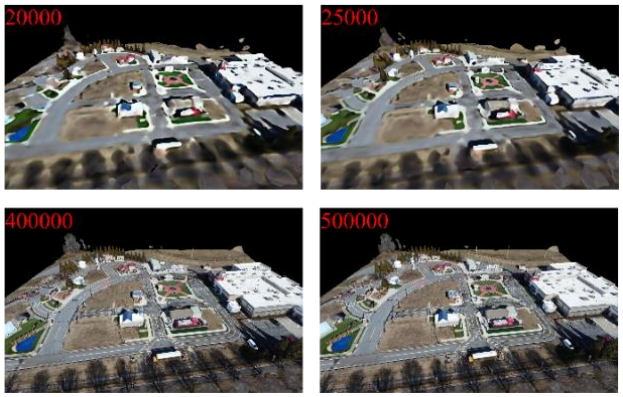

Figure 12. Visualization of surface mesh extracted during training at various checkpoints (iteration noted in top left corner of each image).

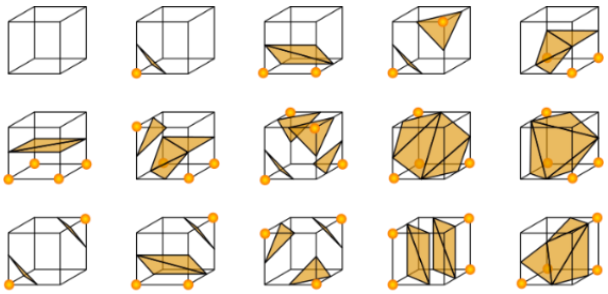

Figure 13. Some pre-defined polygon configurations used by the marching cubes algorithm. Depending on which combination of a cube's vertices fall in or out of an SDF, an appropriate configuration of triangles is selected for inclusion in the mesh.

An important benefit of marching cubes is that it can be efficiently parallelized, making it very suitable for GPU acceleration. This is valuable because the quality of the mesh produced by marching cubes is a function of the density of the 3D grid relative to the complexity of the isosurface boundary. With a sufficiently dense grid, small details can be represented more accurately. Furthermore, the SDF generated from Neuralangelo produces a high-quality surface which captures details that may be missed by traditional photogrammetric surface mesh extraction methods.

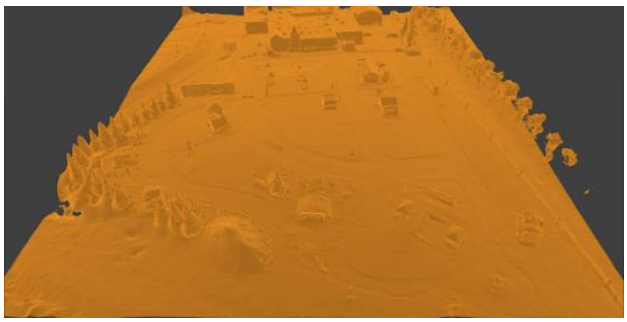

Figure 14. Visualization of marching cubes surface mesh extracted from trained Neuralangelo model.

Neuralangelo employs marching cubes by breaking the scene into blocks and extracting a mesh for each block at the desired density. Figure 14 shows the mesh visualized within the tool Blender. This isosurface exhibits good fine-grain detail considering the relatively sparse single orbit collection geometry. It is also possible to color the mesh by using the estimated surface normals and NeRF model, as shown in Figure 15. While this is a useful result, a very high vertex count is generated to achieve good representation of fine grain surface detail, ~30 million vertices in this example.

Figure 15. Visualization of the final mesh with per-vertex coloring.

Mesh Processing Stage

Apart from the high-quality output of Neuralangelo, there are several issues that arise when using marching cubes which must be addressed with mesh processing to allow 3D Tiles to be created. This process is outlined in Figure 16. First, the resulting mesh can contain a significant number of triangles, with a potentially large number being effectively unnecessary due to being coplanar or nearly coplanar (e.g., on the side of a building or areas of flat terrain). Having a large polygon count limits the usefulness of the mesh, as most existing 3D applications used for visualization struggle to perform well as polygon counts increase. Additionally, the amount of memory required to store such dense meshes increases the time required for loading from storage, the bandwidth for transmitting over networks, and space for representing in GPU memory. Combined, these factors reduce the overall scale that can be practically represented.

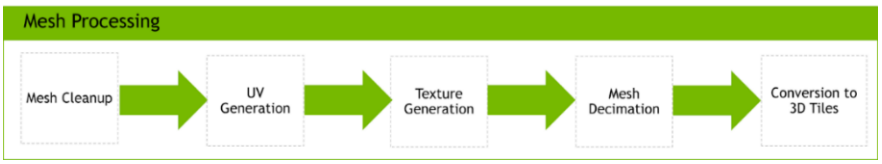

Figure 16. Mesh processing pipeline for converting marching cubes mesh to 3D Tiles.

Second, Neuralangelo embeds the color information with each mesh vertex, as shown in Figure 17. Typical 3D mesh representations used in training and simulation applications utilize texture mapping, such that a 2D texture image that holds the color information for a model is mapped onto the 3D surface. This is also the method utilized by traditional photogrammetry. An important advantage to texture mapping is that it enables the resolution of the geometry to vary independently from the surface color, which is necessary when representing a mesh at varying levels of detail (LODs). For example, the side of a building can be represented using only two triangles if an image of the facade is mapped onto the surface.

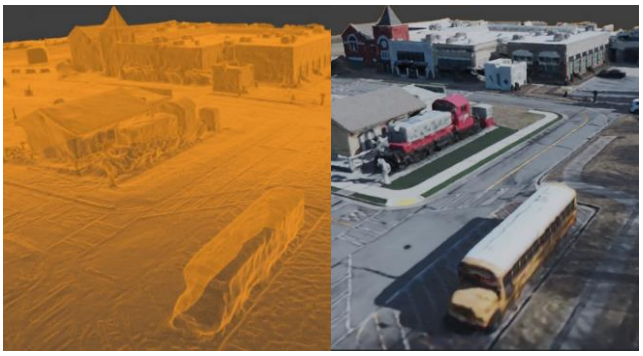

Figure 17. Original mesh produced by Neuralangelo with (left) and color information (right).

Third, because the marching cubes processing is performed in chunks, the output of Neuralangelo contains small discontinuities along major axis boundaries due to floating point precision errors resulting in a mesh that is not "watertight". As such, any attempt to simplify the mesh results in the exaggeration of these discontinuities, as shown in Figure 18. Visualizing meshes with these discontinuities in graphics applications results in anomalies such as scintillation and aliasing, generally considered unacceptable artifacts as they can be extremely distracting. As shown in Figure 19, these discontinuities can be addressed by employing some relatively simple pre-processing to merge vertices that are very close and remove T-vertices (that is, vertices that fall along the edge of an adjacent triangle). Furthermore, one can also eliminate small, disconnected objects, which would include things like vegetation clumps or other "orphaned" objects that would appear as floating geometry. This results in a well-structured mesh which can then be simplified. Note that important geometric features are preserved, and in some cases the elimination of high-frequency undulations improves the visual quality of nearly planar surfaces such as buildings and roads. Although correcting discontinuities prior to simplification results in a higher-quality mesh, the issue remains that simplifying a mesh that uses vertex-based color will result in a significant loss in visual fidelity, as shown in Figure 20.

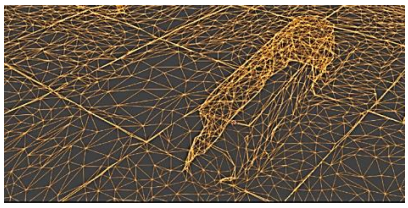

Figure 18. Boundary discontinuities.

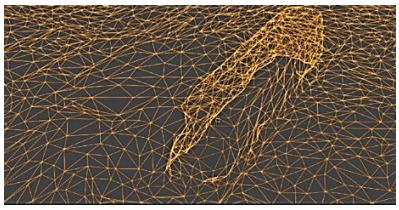

Figure 19. Mesh simplification after discontinuity removal.

Figure 20. Neuralangelo mesh after clean-up and simplification.

Before simplification, we must first transfer the per-vertex mesh colors into a 2D texture image that we can then map onto the surface. This requires that we first parameterize the 3D mesh's (X,Y,Z) vertices into 2D coordinates, typically designated as normalized (U,V) values. There are numerous approaches for mesh parameterization. For the purposes of this workflow, an important desirable property is that the parameterization produces large contiguous patches. This is because although each vertex color could simply be mapped into an isolated 2D coordinate, with this approach there is no guarantee that neighboring vertices would map to nearby 2D coordinates, greatly limiting the possibility for mesh simplification.

Another parameterization approach would be to use projection. For example, assuming the Z axis represents up/down, one could simply take the (X,Y) values of each vertex and normalize them by subtracting the minimum coordinate and dividing by the extent of the dataset. This would effectively be a top-down orthographic projection that produces a single contiguous mapping that can be greatly simplified. Unfortunately, this approach does not produce good results for vertical surfaces because vertices that are directly above and below one another share the same (U,V) values, so color information is lost in these regions, as shown in Figure 21

Figure 21. Top-down mesh parameterization causes loss of color detail in vertical geometry.

Other projections (e.g., cylindrical, spherical, cubic, etc.) all result in similar distortion artifacts for different 3D mesh compositions. For this reason, there has been a significant amount of research to develop mesh parameterization algorithms that behave well with complex 3D geometry by strategically placing seams and flattening out arbitrary curved patches into the 2D plane, as shown in Figure 22. In this example parameterizing a 3D elephant onto a 2D plane is non-trivial, but necessary for producing a texture without significant distortion. Note that even non-co-planar regions are packed into contiguous areas of the texture.

Figure 22. A complex 3D mesh (left) and a corresponding 2D parameterization (right).

These algorithms must balance a tradeoff: some seams are necessary to split the geometry into patches and construct a UV map without too much stretching and distortion, but placing too many seams would split the map into many incoherent islands, limiting the ability to simplify the mesh.

We make use of a custom AutoUV solution to automatically construct high-quality UV maps, which is based on Boundary First Flattening (Sawhney and Crane, 2017) and includes additional heuristics to decompose very large meshes into reasonably sized patches, and place texture seams in unobtrusive regions of the reconstructed geometry. Each separate island is flattened independently in parallel for efficiency, and the resulting UV patches are combined into a non-overlapping atlas with a brute-force polygon packer, as shown in Figure 23.

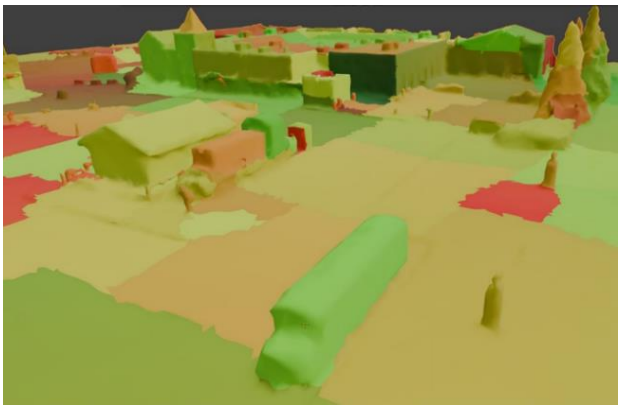

Figure 23. AutoUV Mesh parameterization of the Safety Park dataset. The quilt-like patchwork indicates large contiguous areas of the mesh. Note that non-planar surfaces can be grouped, as indicated by the school bus.

The 2D parameterization produced by the AutoUV approach can be utilized to transfer vertex colors to the areas of the image that correspond to the (U,V) coordinates. As seen in Figure 24, numerous objects are recognizable in the resulting texture image, including the school bus.

With the mesh parameterized into 2D space, we can then perform simplification using an algorithm that preserves texture mapping values, such as quadric edge collapsing. This process recursively removes triangles based on removing vertices that are least important to preserving the shape. This results in a mesh with a target number of triangles, and so can be tuned for the desired level of complexity for the highest level of detail required. With the Safety Park dataset, we found that reasonable quality results were still achievable even when reducing from approximately 20,000,000 triangles down to 200,000.

Figure 24. On the left, the texture generated by transferring vertex colors to a texture image based on the mesh parameterization produced by AutoUV. On the right, a subset of the texture focusing on the school bus.

Given a textured and simplified 3D mesh, it is then possible to convert the mesh into 3D Tiles. Developed by Cesium, 3D Tiles is an Open Geospatial Consortium (OGC) standard that allows geospatial geometry, textures, and metadata representing the terrain and other objects to be represented in a level-of-detail tile hierarchy based on glTF, a compact 3D format designed for efficient streaming and GPU rendering (Open Geospatial Consortium, 2024). Many industry-standard visualization frameworks and applications provide support for 3D Tiles, including Epic's Unreal Engine, Unity, and NVIDIA Omniverse. Cesium's Ion platform was used for this effort, as it enables the creation of 3D Tiles and provides a mechanism for hosting and streaming 3D Tiles both from the cloud or on-prem. The Ion web interface allows existing meshes to be uploaded to the platform and for the geospatial coordinates of the mesh to be specified. Applications can then utilize the appropriate login credentials to enable remote access to the assets associated with a specific account.

Visualizing the surface meshes in Omniverse via the Cesium 3D Tiles plugin shows that the data can be aligned to other mapping products like Google Maps Photorealistic 3D Tiles, producing a much higher resolution update. Omniverse renders the scene using real-time ray tracing, a computer graphics technique that enables interactive visualization of extremely complex environments, on the order of billions of polygons (Woodard, 2019). Ray tracing supports physically based rendering (PBR), which utilizes light and surface material properties to compute accurate pixel colors. A future improvement to this process is to utilize AI to generate material mappings for each texel in the mesh texture, enabling dynamic relighting as well as the simulation of non-visible spectrum sensor modalities.

Temporal Use Case Results

A good use case for the workflow highlighted in the prior section is to perform multiple scans of a site that is undergoing change, as shown in Figure 25. This allows one to rapidly create an updated digital twin of the scene and compare changes that have occurred over time. To illustrate this concept, we utilized the same drone and collection orbit strategy that was used in the Safety Park scene to record data of a construction site. After waiting for two weeks, we re-scanned the same site and performed 3D reconstruction**.** The two models were then rendered in the same environment, which provided the ability to toggle between them and observe the progress. It is important to keep in mind that the total cost / time of the collection was very minimal, with each scan only taking a total time of ~20 minutes.

Figure 25. Temporal results, comparing Photorealistic 3D Tiles (top left) with mesh derived from our process (top right). Rescan differences such roof and siding (lower left vs lower right).

3DGS QuickView Results

In this section we explore the QuickView stage implemented by training a 3DGS model via gsplat training and rendering. The construction scene from the prior section presents an interesting challenge due to the highly complex nature of the scene. The tower crane and 2x4 structures are quite difficult to reconstruct due to many fine grain details, while the gray concrete gives little contrast. In Figure 26 we show a QuickView result that highlights the exciting potential of 3DGS for high fidelity rendering.

We also explore QuickView results for a larger scale scene that presents a significant challenge in both scene complexity and data volume. The data for this scene was collected by the US Civil Air Patrol aircrews working in the Florida Panhandle in the aftermath of Hurricane Michael. They conducted imagery flights with specially equipped aircraft in support of the Federal Emergency Management Agency. The system used two Canon EOS 5DS R cameras with a 50mm lens, collecting ~1700 nadir and oblique 50Mpix images.

Figure 26. 3DGS rendering highlighting fine-grain detailed rendering (top), 2x4 structure (lower left), and crane supports (lower right).

The source imagery was then passed through our data preparation stage to acquire the camera pose and sparse point cloud used to initialize the gsplat model. Each image location pose can be optimized during gsplat training to account for any residual pose errors after running the COLMAP SfM pipeline. Figure 27 shows an overview image rendered from both Google Earth and our gsplat model. The extent of building and facility damage is clearly visible.

Figure 27. Render of Tyndall AFB using gsplat (left) and current day from Google Earth (right).

Finally, we compare the visual quality between the 3D model provided by the civil air patrol, which was created via traditional processing means with SkyLine Photomesh, to the model produced via gsplat training. Figure 28 shows how traditional photogrammetry can produce artifacts with missing information in areas of uniform color. Holes appear in regions on the jets and roof of the white truck. The 3DGS model does an excellent job of reconstructing these regions. It should be noted that the image quality is visually superior in the traditional case with some remaining softness in the 3DGS result, indicating pros and cons of each approach.

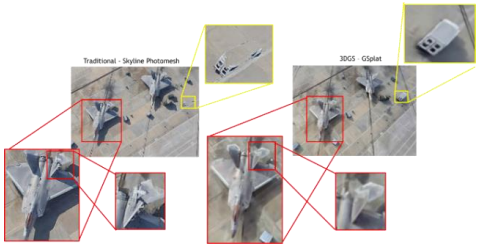

Figure 28. Geometry artifacts present in traditional pipelines (left) vs more complete capture using 3DGS (right).

2DGS Mesh Extraction Results

While high quality mesh extraction can be achieved via NeRF-based approaches, as highlighted in the surface mesh extraction section with Neuralangelo, Gaussian Splatting approaches for surface mesh extraction are rapidly progressing in quality and speed. Figure 29 shows the results from running 2D Gaussian Splatting for Geometrically Accurate Radiance Fields on the Safety Park dataset using the default hyperparameters. It should be noted that some options like pose optimization and depth loss are not currently available but will be available soon in the GSplat implementation.

Figure 29. Surface mesh visualization of Safety Park generated via 2DGS.

Metric Comparisons

It is also useful to compare several metrics between the techniques that we have utilized in this work to get a sense for some of the practical tradeoffs when running on real world datasets. These results, shown in Table 1, are highly dependent on the total number of pixels accessed during training, scene complexity, hyperparameter settings, and computational hardware. Given current trends, these training runtimes will decrease as the approaches mature and hyperparameter tuning for this type of data becomes more standard.

Table 1. Safety Park results from 124 4K images with one NVIDIA 6000 Ada GPU.

Method	Training Runtime	Checkpoint Size	Product
InstantNGP	< 1 min.	1.2GB	Native Render
gsplat	25 min.	543 MB	Web Render
Neuralangelo	12.5 hrs.	1.5GB	Mesh
2DGS	1.5 hrs.	310 MB	Mesh

CONCLUSIONS AND FUTURE WORK

Our main contribution is a novel method for generating textured, geospatial 3D geometry from 2D aerial images without 3D supervision or prior shape information. Our method demonstrates the strength of GPU-accelerated Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) for producing high-quality meshes that preserve fine details and colors of the input images. We outlined mesh processing steps that enable the creation of 3D Tiles, suitable for geospatial workflows. We demonstrated the effectiveness of our method on various real-world datasets and showed that it can outperform existing methods in terms of visual quality, geometric accuracy, and mesh complexity.

Future work will focus on continued optimizations of each stage of the processing pipeline, as well as automating stages that currently require manual input. There are many opportunities for applying GPU acceleration, including camera pose estimation, mesh extraction, mesh parameterization, and mesh simplification. Applying multi-GPU and multi-node acceleration will make the use of these methods on large datasets practical when combined with the overall framework described in (Li, R. et al, 2024).

We also plan to investigate the use of non-visible-spectrum imagery, which has shown promise with initial experiments. We will also explore the application of automated segmentation of the output, enabling material property assignment, model substitution, and the application of different mesh processing algorithms appropriate for each type of scene geometry. There is significant ongoing research in each of these areas, which are independent of the approaches used for mesh creation, such that they could easily be applied as additional stages of the pipeline we have outlined. Finally, object segmentation applied directly to a trained 3DGS model is an exciting and promising approach, which in some cases can bypass the need for large-scale mesh extraction.

REFERENCES

Huang, B., Yu, Z., Chen, A., Geiger, A., & Gao, S. (2024). 2d Gaussian Splatting for Geometrically Accurate Radiance Fields. *arXiv preprint arXiv:2403.17888*.

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, *42*(4), 139-1.

Kim, JJ. "What Is Photogrammetry?" NVIDIA Blog, NVIDIA, 23 June 2023, blogs.nvidia.com/blog/what-is-photogrammetry/.

Li, R., Fidler, S., Kanazawa, A., & Williams, F. (2024). NeRF-XL: Scaling NeRFs with Multiple GPUs. *arXiv preprint arXiv:2404.16221*.

Li, Z., Müller, T., Evans, A., Taylor, R. H., Unberath, M., Liu, M. Y., & Lin, C. H. (2023). Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8456-8465).

Lorensen, W. E., & Cline, H. E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87 (pp. 163--169). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/37401.37422

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Nerf, R. N. Representing scenes as neural radiance fields for view synthesis., 2021, 65. *DOI: https://doi.org/10.1145/3503250*, 99-106.

Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, *41*(4), 1-15.

NVIDIA CUDA Documentation (7/6/2024). Retrieved from https://docs.nvidia.com/cuda/

Open Geospatial Consortium 3D Tiles Overview (7/6/2024). Retrieved from: https://www.ogc.org/standard/3dtiles/

Sawhney, R., & Crane, K. (2017). Boundary first flattening. ACM Transactions on Graphics (ToG), 37(1), 1-14.

Schonberger, J. L., & Frahm, J. M. (2016). Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4104-4113).

Woodard, T., (2019), Ray Tracing for Training and Simulation, *IMAGE 2019 Conference Proceedings*, Dayton, OH

Yu, Z., Sattler, T., & Geiger, A. (2024). Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*.