GIMME: 3D Gaussian Inverse Rendering for Mobile Mesh Extraction

Neil Nie
neilnie@stanford.edu

Hannah Norman
hnorman@stanford.edu

Abstract

We present a method to enhance the practicality of radiance fields for real-time mesh-extraction applications on mobile platforms. While Neural Radiance Fields (NeRF) offer photorealistic scene rendering, their computational intensity hampers real-time execution in environments lacking compute resources, such as mobile devices. By leveraging 3D Gaussian Splatting (3DGS), we address the computational challenges associated with NeRF, enabling photorealistic scene rendering on resource-constrained platforms. Our approach involves an iOS application for data capture, a pipeline for training 3DGS models, and a mesh extraction system for obtaining textured meshes ready to render. We demonstrate real-time rendering speeds and high visual quality on established datasets and newly collected data.

1. Introduction

In recent years, the intersection of neural rendering and real-time graphics has advanced digital asset creation and usage across various domains, notably in gaming and immersive augmented reality (AR) and virtual reality (VR) environments [14]. Neural Radiance Fields (NeRF) [7] have emerged as a technique for photorealistic scene rendering. However, the computational intensity of NeRF models makes them impractical for real-time applications, particularly on mobile platforms. This paper introduces a novel approach that leverages 3D Gaussian Splatting (3DGS) [6]—a technique for efficiently approximating the volumetric rendering equation—to enhance the viability of NeRF-derived assets in real-time, resource-constrained environments.

Historically, the creation of 3D models relied heavily on methods such as implicit representations with signed distance functions (SDF) and voxel-based models [3]. These techniques, while effective, often required sophisticated and costly multi-view capture systems that limited their accessibility and practicality, especially for dynamic and complex scenes. Moreover, the computational overhead and the lack of scalability in these methods presented substantial barriers to real-time application in gaming and interactive media.

Neural rendering introduces a paradigm shift by utilizing deep neural networks to synthesize photorealistic images from novel viewpoints [14]. Among the most notable techniques, NeRF offers a compelling approach by modeling the volumetric scene as a continuous function that maps 3D coordinates to color and density [7]. This technique has shown remarkable success in producing high-fidelity results from sparse inputs but can be computationally intensive.

3D Gaussian Splatting (3DGS) addresses these computational challenges by providing an efficient way to approximate the volumetric rendering equation [6]. By reducing the computational load, 3DGS makes it feasible to use NeRF-derived assets in real-time applications, especially on mobile platforms. This technique is integral to our approach, which aims to combine the strengths of neural rendering with the efficiency of 3DGS.

The core of our methodology has three components:

- A light-weight iOS application to capture data for neural rendering using ARKit
- A pipeline to train 3DGS models for real-time render
- A mesh extraction system to retrieve textured meshes from the 3DGS models for downstream applications, such as AR/VR gaming

This work is motivated by the growing demand for high-quality digital assets that can be dynamically generated and deployed in interactive media. Neural rendering stands at the forefront of this demand, offering unprecedented levels of realism and immersion. By introducing a method that integrates scene retrieval with mesh generation, our approach extends the benefits of neural rendering to mobile and AR/VR platforms, making it both a technically interesting and viable solution. In sum, our contributions are:

- **Fast and accurate asset capture**: Creating photorealistic 3D models for use in AR/VR environments can be hindered by high computational demands and a need for sophisticated capture systems. We address this problem by building a smartphone capture system.
- **Efficient mesh extraction**: Traditional 3D modeling techniques are cumbersome for real-time applications. We address this problem by integrating the SuGaR mesh reconstruction system [4] into our pipeline.

Our project proposes a solution that reduces computational demands while maintaining the high quality of NeRF-derived assets, making it viable for mobile and real-time applications. We call our approach GIMME. Our GIMME system significantly enhances photorealistic rendering of novel scenes using 3D Gaussian Splatting and SuGaR mesh extraction, delivering detailed, textured meshes that integrate seamlessly into AR and VR environments. For details on our qualitative results, see Figure 2.

2. Related Work

In describing our pipeline, we first discuss foundational work on Neural Radiance Fields (NeRF) [7]. Then, we focus on state-of-the-art (SOTA) real-time radiance field rendering methods—most significantly, 3D Gaussian Splatting (3DGS) [6]. Finally, we explore approaches for inverse rendering and mesh extraction for scene factorization, which have direct applications to digital asset creation.

Neural Radiance Fields for View Synthesis Mildenhall et al. present NeRF [7], a seminal work in the field of novel view synthesis. Given a series of images with known camera positions, the NeRF architecture leverages multi-layered perceptron (MLP) networks to render photorealistic continuous scenes at high resolution. This method preceded and motivated an explosion in the field of neural rendering [14], and it serves as a baseline for our implementation.

Soon after publication, the creators of NeRF addressed shortcomings of the original method—specifically aliasing artifacts and loss of fine-grained detail—with mip-NeRF [1]. This updated approach produces higher quality results, executes at a 7% faster rate, and is half of NeRF's original size [1]. Mip-NeRF became the new SOTA for neural radiance fields, and it is frequently utilized as a baseline comparison with proposed methods in the field.

One shortfall of NeRF is its portability to other platforms, particularly those with weaker computing resources. MobileNeRF offers a solution [2]. By adapting NeRF's framework to a more traditional rendering pipeline with polygon rasterization, MobileNeRF presents a parallelization scheme for NeRF that allows the method to run at competent frame rates on mobile devices [2]. Although our proposed approach does not render scenes on a mobile platform—focusing only on data capture with our iOS app—MobileNeRF suggests potential future work where we could integrate the data capture app with a full scene rendering pipeline optimized for mobile devices.

Real-Time Radiance Field Rendering NeRF catapulted neural rendering to the forefront of computer vision tasks, and though it offers high visual fidelity of generated scenes, it lacks real-time efficiency. Instant-NGP provides a solution to this train- and render-time problem [8]. Leveraging hash and occupancy grids as well as a smaller MLP network, Instant-NGP achieves significant speedup in the training of neural radiance fields, producing comparable results in up to five minutes when compared to the hours necessary for the SOTA Mip-NeRF method [8]. Even so, this speedup comes at the expense of photorealism, creating a tradeoff between quality and efficiency.

3D Gaussian Splatting (3DGS) [6] closes this gap. The 3D volumetric representation of 3DGS utilizes 3D Gaussians, and the method relies on anisotropic splatting to render scenes [6]. With this approach, 3DGS achieves real-time rendering alongside higher visual fidelity to the ground truth than NeRF and related methods. 3DGS represents the current SOTA, and we build our pipeline atop it.

Inverse Rendering and Mesh Extraction The ability to synthesize novel views of continuous 3D scenes given only a series of 2D images suggests a myriad of applications. We turn specifically to the sub-field of inverse rendering and mesh extraction, with a target use-case of generating 3D models and assets for use in game development and animation. A number of papers have explored extensions to NeRF and 3DGS to reach this end.

Following NeRF's release, its creators presented NeRV: Neural Reflectance and Visibility Fields [11], which allows complex illumination unseen during training. This is critical for 3D asset generation since models must offer high visual fidelity in new environments with unique lighting. Similarly, NeRFactor [16] addresses the challenge of factorizing scenes into shape and reflectance, enabling realistic rendering under novel lighting conditions by decomposing the visual input into these intrinsic components.

Yariv et al. offer another solution to the meshing problem with BakedSDF [15]. This method leverages spherical Gaussians and targets the polygon rasterization pipeline [15], similar to MobileNeRF [2]. It achieves real-time rendering speeds and produces meshes of high quality by baking neural scene representations into triangle meshes [2].

SuGaR, proposed by Guédon et al. [4], and 2DGS, proposed by Huang et al. [5], take the mesh extraction pipeline one step further by leveraging Gaussians. The former utilizes 3DGS [4], while the latter introduces a novel approach called 2D Gaussian Splatting (2DGS) [5]. Both methods generate high-quality meshes in real time and represent the SOTA of mesh extraction for inverse rendering.

3. Methodology

Here, we present our GIMME system. See Figure 1.

- First, we describe our data capture approach, encapsulated within a lightweight app targeting iOS devices.
- Then, we describe our method for training high-fidelity 3D Gaussian scene representations in real-time.
- Finally, we describe our mesh extraction strategy that leverages the SuGaR method [4] to retrieve precise, detailed meshes from the 3DGS scenes.

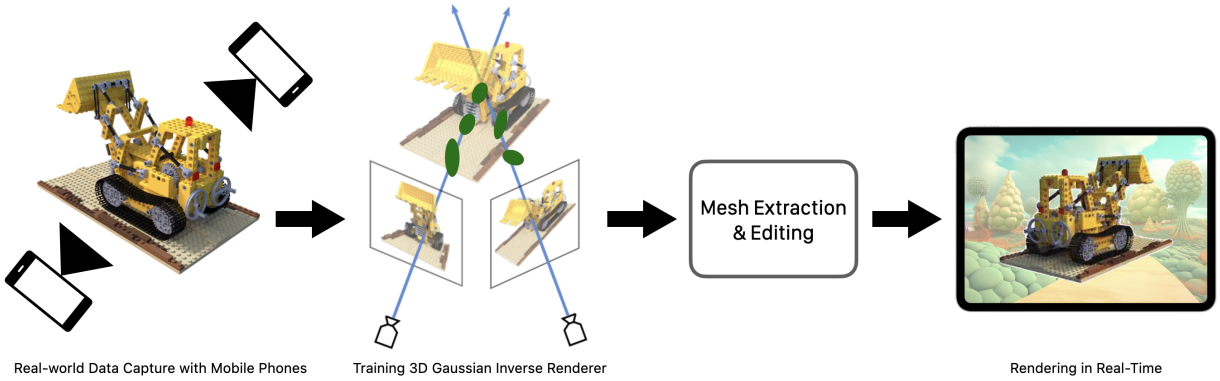

Figure 1. **GIMME system pipeline.** First, the GIMME Data iOS app is used to capture data for training neural radiance fields. Then, a 3D Gaussian inverse renderer trains on this data and produces a 3D scene. Mesh extraction and editing takes place next, after which a refined mesh can be rendered in real-time on any mobile or AR/VR application.

3.1. Data Capture

The GIMME system begins with a data capture app. This iOS app, called GIMME Data, is integrated with ARKit to capture data for training neural radiance fields (NeRF) and 3D Gaussian Splatting (3DGS) [1]. Leveraging ARKit, the system captures depth information and estimates camera extrinsics essential for NeRF, while simultaneously collecting images suitable for 3DGS processing.

After data collection, the system transitions to a robust processing pipeline designed to convert the data into the standard COLMAP format [10, 9]. COLMAP is widely recognized in the field for its utility in 3D reconstruction and photogrammetry. By converting to this format, our system facilitates a seamless integration into existing workflows, allowing for efficient model training, testing, and evaluation. This comprehensive approach not only maximizes the utility of the captured data but also streamlines the development process for applications involving advanced 3D rendering and visualization technologies.

3.2. 3D Gaussian Splatting

3D Gaussian Splatting [6] renders scattered data points into a continuous volume. This method is particularly effective for visualizing spatial data in three dimensions by interpolating the values onto a structured grid. It represents each data point as a Gaussian function, smoothly distributing the point's influence over a localized region in space [6].

Mathematically, for a given data point located at $\mathbf{x}_i$ with an associated scalar value s_i, the contribution of this point to the grid volume is modeled as a Gaussian function centered at $\mathbf{x}_i$ [6]. The value contribution $V(\mathbf{x})$ at any point $\mathbf{x}$ in the space is given by the equation:

$$V(\mathbf{x}) = \sum_i s_i \cdot \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$$

where σ is the standard deviation of the Gaussian kernel, which controls the spread of the data point's influence [6]. This parameter determines how much each data point contributes to its surrounding region in the grid, affecting both smoothness and resolution of the resulting visualization.

The integration of contributions from multiple data points is computationally managed by summing their Gaussian functions over the grid [6]. Each point effectively "splats" its value onto the grid according to the Gaussian distribution, blending with contributions from other points to produce a smooth, continuous field. An important aspect of Gaussian splatting is the choice of σ, which must be optimized based on the density of data points and the desired smoothness of the visualization [6]. This optimization is essential to balance detail and smooth interpolation without excessive blurring or aliasing effects, maintaining a faithful representation of the underlying data:

$$G(\mathbf{x}) = \sum_i V_i(\mathbf{x})$$

where $G(\mathbf{x})$ represents the final interpolated value at any given point $\mathbf{x}$ in the output volume, accumulating the influences of all data points appropriately weighted by their Gaussian splats [6].

After retrieving visual scene data with our GIMME Data app, we train a 3D Gaussian inverse renderer on it, inspired by the original 3DGS framework [6]. With the relevant images and the camera intrinsics values, we convert the data into the COLMAP format, and use the 3DGS open-source repo to train one model for each scene.

[1]The data capture system is enabled by resources from the open-source iOS developers community and Apple's ARKit documentation and example projects. For details, see `https://developer.apple.com/documentation/arkit/`.

3.3. Mesh Extraction

The use of Gaussian splatting to render scenes in real-time presents opportunities and challenges, as discussed extensively in the literature [3]. While the method excels in smooth data interpolation and visualization, integrating it directly into real-time rendering engines like Blender or ARKit introduces complexity. A significant limitation is the difficulty of implementation within these frameworks. These systems are primarily designed for polygon meshes and traditional rendering pipelines rather than volume-based data representation. As a result, adapting them to support the different data handling and rendering processes required by Gaussian splatting requires extensive modifications to the core rendering algorithms.

Moreover, another challenge with Gaussian splatting in real-time scenarios is the lack of straightforward mechanisms to compose separate Gaussian-splatted entities with other scene elements [3]. Gaussian splats inherently blend into each other, making it challenging to maintain distinct boundaries between different objects or layers within a scene. This blending can obscure details and complicate the task of scene composition where clear demarcation and interaction between objects are necessary.

An alternative approach to address these issues is to convert the volume data generated by Gaussian splatting into polygonal meshes, though this largely unexplored [13]. This conversion would enable the use of Gaussian-splatted data within traditional rendering pipelines and simplify composition with other scene elements. However, extracting accurate, detailed meshes from 3D Gaussian volume data is challenging. The process typically requires additional computational steps to define surface boundaries, extract meshes, and assign material properties. These steps are essential for ensuring that the meshes can be effectively used in downstream tasks such as animation, collision detection, and more. The complexity of this conversion process thus represents a substantial barrier to the practical use of Gaussian splatting in real-time rendering applications.

In this section, we describe our mesh extraction approach from 3DGS, which is based on "SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering," by Guédon et al. [4]. The SuGaR method creates accurate, editable meshes from 3D Gaussian splatting representations by optimizing the properties of tiny 3D Gaussians based on training images [4]. This technique is noted for its rapid performance compared to Neural Radiance Fields (NeRF).

To address the disorganized nature of Gaussians post-optimization, the authors introduce a regularization term that encourages the Gaussians to align more closely with the surface of the scene [4]. This regularization is essential for improving the distribution and alignment of the Gaussians, thereby facilitating the subsequent mesh extraction process.

For mesh extraction from Gaussians, the paper describes an efficient method to sample points on the visible part of a level set of the density function defined by the Gaussians [4]. These sampled points are then used in the Poisson reconstruction algorithm, which results in a detailed triangle mesh within minutes on a single GPU [4]. Following the initial mesh extraction, an optional refinement strategy is employed where the mesh and a subset of 3D Gaussians are optimized jointly through 3DGS rendering [4]. This optimization not only enhances the quality of the mesh rendering but also enables the effective use of traditional mesh editing tools, such as Blender or MeshLab.

The approach ingeniously combines the speed of Gaussian splatting for quick scene capture with the advantages of mesh-based representations for editable, high-quality rendering, marking it as a significant contribution to the field of computer graphics. Mathematically, the key equations involved include the definition of the density function $d(p)$ for a point p in space:

$$d(p) = \sum_g \alpha_g \exp\left(-\frac{1}{2}(p-\mu_g)^T \Sigma_g^{-1}(p-\mu_g)\right)$$

where μ_g, Σ_g, and α_g represent the mean, covariance, and blending coefficient of each Gaussian, respectively [4]. The regularization term R is formulated as:

$$R = \frac{1}{|P|}\sum_{p\in P} |f(p) - \hat{f}(p)|$$

where $f(p)$ and $\hat{f}(p)$ are the ideal and estimated signed distance functions (SDFs) derived from the density function d [4]. This mathematical framework ensures that the 3D Gaussians align accurately with the scene's surface, facilitating effective mesh extraction and refinement.

After training a 3D Gaussian inverse renderer on our captured data, we feed the resulting scene representation into the SuGaR framework to extract an editable mesh. This integration leverages the strengths of our capture system and the SuGaR methodology to produce high-quality, editable 3D models suitable for various downstream applications.

3.4. Implementation Details

Experimental Datasets To evaluate the practical applications of our system, we utilize a combination of standard datasets from the field of neural rendering and our own generated data. Specifically, the original datasets for NeRF are employed to benchmark our rendering techniques against established norms within the community. As for the data itself, we curate a dataset comprising two scenes from the NeRF Synthetic 360° dataset [7] ("Hotdog" and "Chair"), two scenes from the Mip-NeRF Unbounded 360° dataset [1] ("Bonsai" and "Stump"), and two scenes obtained with our GIMME Data app ("Sink" and "Orchid").

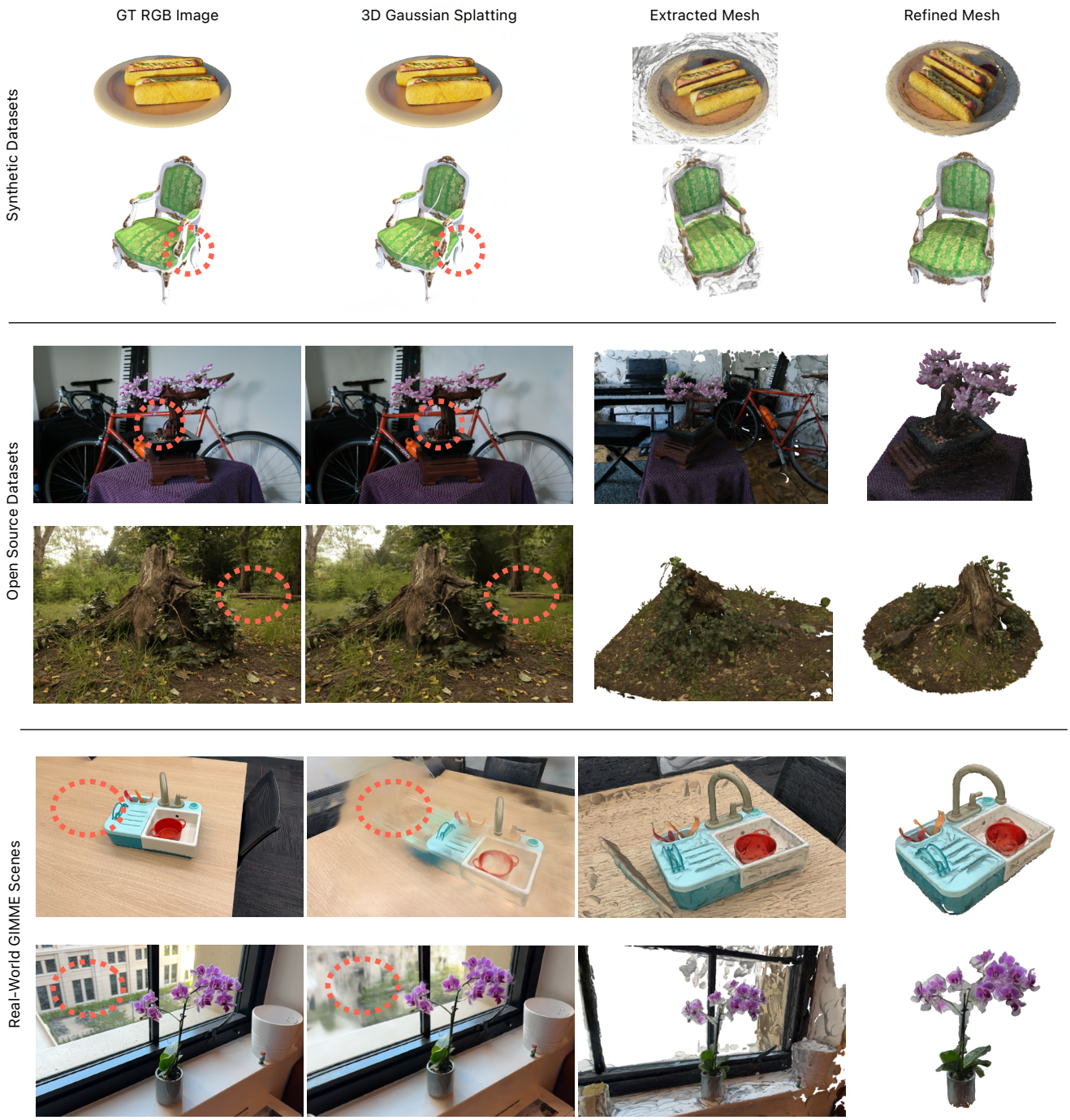

Figure 2. **GIMME pipeline.** The stages of the GIMME system's processing pipeline on synthetic, real-world, and self-generated datasets. Note the visual fidelity our system achieves in the final meshes across various datasets. Undesired artifacts, such as discrepancies in texture and geometry, are highlighted with red dashed circles, indicating areas for potential improvement in our pipeline.

Figure 2 visualizes scenes from our curated dataset and their renders as they progress through the GIMME pipeline, from original data to scene renders to meshes. Each scene is generated from 50 to 200 images captured from various angles. Resolutions range from 500×500 pixels to 2K, with a default resolution of 1920×1440. When using the GIMME Data app, we aimed to mimic real-world conditions and challenges typical in mobile and AR/VR environments, including reflective surfaces that can cause aliasing.[2]

[2]See Figure 3 in the Appendix for an image of GIMME Data's UI.

Metric	Synthetic		Real World		Our Data	
	Hotdog	Chair	Bonsai	Stump	Sink	Orchid
PSNR ↑	32.03	28.34	28.50	24.70	19.21	21.10
LPIPS ↑	0.0561	0.0669	0.1770	0.3320	0.4264	0.3805
SSIM ↓	0.9698	0.9451	0.9237	0.7231	0.6079	0.7627

Table 1. **Evaluation metrics for different scenes.**

Evaluation Metrics In evaluating our rendering pipeline, we use the Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS), and Structural Similarity Index Measure (SSIM) metrics [17]. These standard metrics quantify the quality of reconstructed images compared to their original high-resolution counterparts, providing clear measures of accuracy and visual fidelity. For evidence of high-fidelity rendering, we seek high PSNR and LPIPS scores and low SSIM scores.

4. Results

We consider the high-fidelity results produced by GIMME Data's generated scenes as evidence of the practicality of our lightweight iOS application in capturing high-quality data across various real-world settings, emphasizing the accessibility and usability of our approach for developers and content creators in the AR/VR space. These insights not only validate our technical contributions but also affirm the potential of GIMME to revolutionize asset creation in immersive media applications.

Furthermore, we offer analyses of our results across all scenes using the GIMME system below.

4.1. Quantitative Analysis

The numerical results shown in Table 1 show excellent performance across all data types, though different metrics lead to various conclusions. The synthetic data, specifically the "Hotdog" scene, scored highest on PSNR, followed closely by the real-world data, and finally by our GIMME data. Although the GIMME data has the lowest PSNR scores, it still achieves high numbers overall, illustrating the high fidelity of the resultant meshes. The SSIM scores follow a similar trend. Interestingly, the LPIPS scores are lower than expected when compared against scores for similar scenes in the literature [7, 6, 4]. Furthermore, we can see that the ranking is reversed: the GIMME data achieves the highest LPIPS scores, followed by the real world data, and the synthetic data—specifically the "Hotdog" scene which had scored the best for both PSNR and SSIM—achieves the lowest LPIPS rating. Taken together, the quantitative results suggest resultant scenes with strong visual fidelity overall across each scene in the curated dataset. We take these results to affirm the practical utility of the GIMME pipeline under varied conditions.

4.2. Qualitative Analysis

The qualitative results of the GIMME system more so underscore the efficacy of the 3D Gaussian Splatting (3DGS) [6] technique and SuGaR [4] system in enhancing real-time, photorealistic rendering and mesh extraction from novel scenes. Figure 2 illustrates this, where the high visual fidelity and smoothness of the rendered scenes are on full display. Notably, the integration of the SuGaR mesh extraction method proves efficient in generating detailed, textured meshes that can seamlessly integrate into any AR and VR environment.

We see that the extracted meshes are highly similar to the training images. The pipeline can capture fine details such as leaves on the bonsai tree, arms of the chair, and utensils in the toy sink. With some manual editing and mesh refinement using a standard application like Blender or MeshLab, the highly accurate refined meshes prove largely suitable for downstream applications.

That being said, there remain undesired artifacts in the rendered scenes. In particular, the 3DGS model struggles to capture the reflective plastic surfaces of the LEGO pieces in the "Bonsai" scene as well as the window in the "Orchid" scene. The model also struggles with the background of these large and complex scenes. The extracted meshes sometimes have holes near surfaces that are not observed during training. Even so, we note the high visual fidelity of the meshes overall.

5. Conclusion

The project successfully achieves its goal of constructing a pipeline for 3D Gaussian-enabled mesh extraction using data captured on a mobile device. We developed an iOS app, GIMME Data, which streamlines the data collection process and enhances user accessibility. Utilizing the 3D Gaussian Splatting (3DGS) framework, we effectively render 3D scenes from the captured data and integrate this output with the SuGaR mesh extraction system. The resulting meshes exhibit high visual fidelity, demonstrating the potential of our approach for practical applications. This work lays a solid foundation for future advancements in inverse rendering, paving the way for more sophisticated and efficient 3D model generation techniques in various domains such as game development, animation, and augmented/virtual reality.

5.1. Future Work

Future work includes accelerating the pipeline for full deployment on mobile devices, enabling the GIMME Data app to handle both data capture and transformation into renderable 3D meshes. We plan to extend our pipeline to include an automated mesh refinement stage, enhancing the SuGaR-generated meshes to a more user-friendly state. Currently, we manually refine the generated meshes using applications like Blender and MeshLab. Automating this stage presents opportunities to streamline the process and enable the extraction of specific objects from each scene mesh. Ongoing efforts focus on these enhancements.

6. Acknowledgements

We wish to thank Tiange Xiang, our CS 231N project mentor, for his feedback and support during this project. We would also like to thank the authors of these open source research projects: NeRFStudio [12], InstantNGP [8], 3D Gaussian Splatting [6], and SuGaR [4].

References

[1] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *CoRR*, 2021.

[2] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *CVPR*, 2023.

[3] A. Dalal, D. Hagen, K. G. Robbersmyr, and K. M. Knausgård. Gaussian splatting: 3d reconstruction and novel view synthesis, a review. *IEEE*, 11, 2024.

[4] A. Guédon and V. Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv*, 2023.

[5] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2d gaussian splatting for geometrically accurate radiance fields. *SIGGRAPH*, 2024.

[6] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph*, 42(4), 2023.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020.

[8] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph*, 41(4), 2022.

[9] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[11] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. *CVPR*, 2021.

[12] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[13] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation, 2024.

[14] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, Y. Wang, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Niessner, J. T. Barron, G. Wetzstein, M. Zollhoefer, and V. Golyanik. Advances in neural rendering. *CoRR*, 2021.

[15] L. Yariv, P. Hedman, C. Reiser, D. Verbin, P. P. Srinivasan, R. Szeliski, J. T. Barron, and B. Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *CoRR*, 2023.

[16] R. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph*, 2021.

[17] X. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *IEEE*, 2018.

Appendix

7. GIMME Data Application UI

Figure 3. **User interface for GIMME Data app.**