DIFFERENTIABLE RENDERING FOR SYNTHETIC APERTURE RADAR IMAGERY

Michael Wilmanski[#*] and Jonathan Tamir[*]
[#]KBR (Current), [*]The University of Texas at Austin

Abstract. There is rising interest in integrating signal and image processing pipelines into deep learning training to incorporate more domain knowledge. This can lead to deep neural networks that are trained more robustly and with limited data, as well as the capability to solve ill-posed inverse problems. In particular, there is rising interest in differentiable rendering, which allows explicitly modeling geometric priors and constraints in the optimization pipeline using first-order methods such as backpropagation. Existing efforts in differentiable rendering have focused on imagery from electro-optical sensors, particularly conventional RGB-imagery. In this work, we propose an approach for differentiable rendering of Synthetic Aperture Radar (SAR) imagery, which combines methods from 3D computer graphics with neural rendering. We demonstrate the approach on the inverse graphics problem of 3D Object Reconstruction from limited SAR imagery using high-fidelity simulated SAR data.

1. Introduction

This work presents a proof-of-concept for differentiable rendering in synthetic aperture radar (SAR) imagery. SAR is an imaging modality based on pulse-doppler radar. As a radar antenna moves, successive pulses of radio waves are transmitted to illuminate a scene. Echoed pulses from multiple antenna positions can be combined, which forms the synthetic aperture that enables higher resolution images than would otherwise be possible for a physical antenna [1]. In aerial and automotive vehicles, the vehicle's natural motion provides the movement needed to form the synthetic aperture.

Motivations for differentiable rendering for SAR are similar to those for electro-optical (EO) imagery. The survey in [2] describes many applications of differentiable rendering in EO imagery. While the most common application is for 3D object reconstruction, differentiable rendering is useful for any first-order optimization problem that may benefit from pixel-level supervisions flowing to 3D properties, including for training deep neural networks. While there have been successful demonstrations of methods for differentiable rendering of EO-domain imagery in recent years, these methods are not directly applicable to SAR-domain imagery, as SAR and EO images exhibit significantly different geometry and phenomenology.

Since SAR is a coherent imaging method and a form of active sensing, many priors about a scene are inherently known and controlled. Specifically, detailed knowledge about the scene's illumination as well as positions of the sensor and illuminator relative to the area on the ground being imaged are always known. In addition to control over illumination and prior positional knowledge that can be expected, SAR imagery is invariant to many uncontrollable environmental effects that mar EO-domain imagery. This includes weather and atmospheric effects such as cloud cover, as well as illumination changes due to day/night cycles. Collectively, these differences mean that SAR-domain images have few scene parameters that are uncontrollable or unknown to the user, and thus are especially well-suited for differentiable rendering.

Therefore, this work presents a proof-of-concept and starting point for differentiable rendering in SAR imagery. Our proposed approach is composed of a "Feature Rasterizer" followed by a "Neural Shader," depicted in the block diagram of Figure 1. The Feature Rasterizer produces image-like features that convey context about the structure and illumination of the scene. These features act as the input condition to the Neural Shader, which is implemented as a conditional generative adversarial network (CGAN). The Neural Shader enhances the scene with realistic SAR scattering effects to complete the rendering process. We focus on the application of 3D object reconstruction for demonstration since it is a common and highly useful downstream task enabled by differentiable rendering. Consequently, differentiation is with respect to an object's mesh vertices.

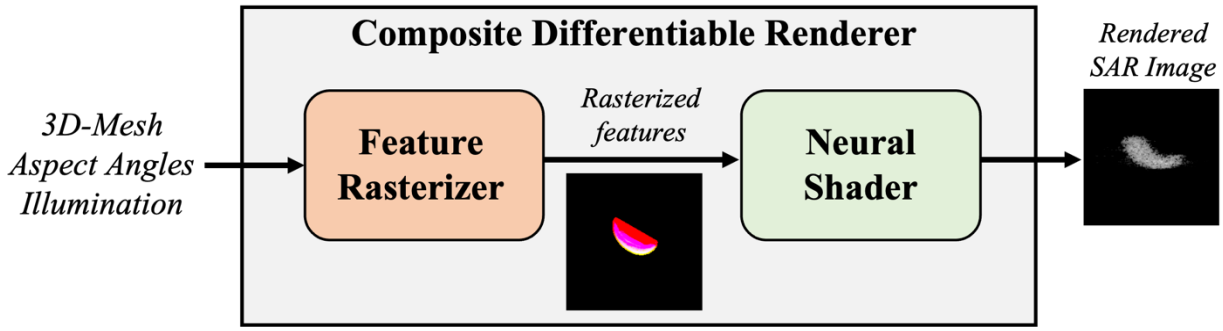

Figure 1: Block diagram of proposed composite differentiable rendering approach.

Our contributions are the following:

1. We develop a differentiable rendering pipeline for SAR imagery, composed of a Feature Rasterizer and Neural Shader. The former uses softened rasterization to project mesh-based scene representations into 2D feature maps tailored to SAR's imaging geometry, while the later uses a CGAN to predict SAR scattering effects from the feature maps.
2. We show how to train the CGAN for shading, using simulated SAR data.
3. We demonstrate proof-of-principle use for solving the inverse graphics problem of 3D object reconstruction.

2. Related Work

2.1 SAR Simulators

The survey of SAR simulators in [3] distinguishes two main categories of SAR simulation systems:

a) SAR *image* simulators, which directly produce focused SAR images.
b) SAR *raw signal* simulators, which simulate raw sensor measurements.

Category (a) simulators include examples such as RaySAR [4], CohRaS® [5], and SARviz [6]. A comparison of these and assessment of their limitations is provided in [7]. Of these, SARviz is the only simulator that uses rasterization, while the others involve a form of ray tracing. RaySAR is

the only project that is open source, but also has the most limited shading effects, relying on Phong shading [8].

Simulators of category (b) are usually based on physical optics approximations [9] and enhanced further with the shooting and bouncing rays (SBR) method pioneered in [10]. Instead of producing images directly, these physics-based simulators simulate raw sensor measurements, which the user must process themselves to coherently form images. These methods are the most realistic, but also computationally intensive. Examples in this category include Xpatch [11], CASpatch [12], FACETS [13], SigmaHat [14], FEKO [15], and POFACETS [16]. Most simulators in this category are closed-source commercial products and do not explicitly support differentiable optimization. Of these, POFACETS is the only project that is open source, but lacks some advanced features such as SBR.

Our proposed differentiable rendering approach would be categorized as a SAR *image* simulator, though the data used to evaluate it was produced by a SAR *raw signal* simulator. Importantly, the simulators above are not suitable for solving inverse problems as they are not easily differentiable, and are therefore limited to black-box optimization.

2.2 Neural and Differentiable Rendering

Neural rendering is an umbrella term referring to a broad field of methods that fuse machine learning models with techniques or domain-knowledge from computer-graphics. A beneficial property of techniques in this family is that they are inherently differentiable; since they are composed of neural network building blocks, the resulting models can be easily backpropagated through. While they have been successful, it is worth noting that most neural rendering methods aim to manipulate existing images, rather than create images from raw scene representations [17]. Such techniques are thus more akin to sophisticated pixel shaders, and not renderers in the literal sense. The survey in [18] describes these collectively as "2D Neural Rendering" methods, and distinguishes them from an emerging paradigm it calls "3D Neural Rendering."

Methods of this new paradigm, such as [19], learn to *represent* a scene in 3D, using a differentiable rendering algorithm to facilitate training a neural network to represent the scene. Methods of this branch of "neural rendering" are consequently best described as neural scene representations, and are yet another application of differentiable rendering. Cases such as [19] use volume rendering techniques, which are naturally differentiable and also memory inefficient.

Besides volumetric techniques, most differentiable rendering work in the literature has focused on rasterization-based methods from computer graphics [2]. Rasterization is computationally-efficient, but not naturally differentiable. However, smoothed variants have emerged [20-22], which have made rasterization-based methods viable. Physics-based differentiable rendering methods which model global light transport effects exist as well [23,24], but are substantially slower. These methods are the most accurate, but the hypothetical improvements in rendering accuracy may not be worth the increased computational cost.

Use of neural and differentiable rendering methods in the context of SAR has so far been limited. A number of papers have proposed utilizing CGANs for image-to-image translation from SAR to EO imagery [25-27]. However, perhaps the most related example is [28], which attempts to improve the realism of the RaySAR SAR image simulator [4] by training a deep neural network

to estimate realistic SAR scattering effects. This falls under the "2D Neural Rendering" paradigm and is similar in principal to our use of a CGAN for the Neural Shader.

3. Data

3.1. Object Shapes

Computer-aided design models for a total of seven object shapes were created, pictured in Figure 2. The objects were created in Blender [29] and exported as triangulated meshes. The ground plane gridded lines are spaced in units of one meter, making these objects similar in size to medium or large vehicles.

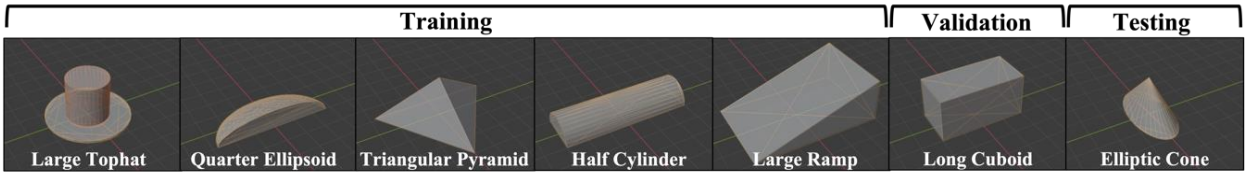

Figure 2: Meshes of the seven objects used to generate train, validation, and test SAR images.

The first five objects are used to generate training data for the Neural Shader. The Long Cuboid is used as a validation object to inform hyperparameter tuning for both the Feature Rasterizer and Neural Shader. The Elliptic Cone is reserved as a test-only object and used only after all aspects of the end-to-end rendering process were frozen.

3.2 Aspect Angles

The data generated for this study consist of six distinct elevation angles, evenly spaced between [10°, 60°], and 36 distinct azimuth angles, evenly spaced between [0°, 350°]. This gives a total of 216 distinct aspect-angle combinations per object. A geometric representation of the aspect angles is illustrated in Figure 3. Data are assumed to be monostatic, meaning transmitter and receiver are co-located, which is the most common scenario for SAR.

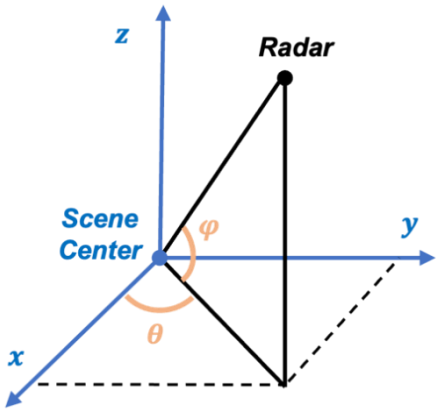

Figure 3: Collection geometry for aspect angles θ (azimuth) and φ (elevation).

3.3 Coherent Augmentation via Polarization

The transmitting and receiving antennas can each be configured at either horizontal (H) or vertical (V) polarizations. Simulations are done for all linear polarization combinations for transmitter and receiver (VV, VH, HV, HH). Although images are ordinarily only formed using co-pol measurements (S_{VV}, S_{HH}), the cross-pol measurements (S_{VH}, S_{HV}) can be used as physically meaningful data augmentation to coherently augment the co-pol measurements for additional data diversity as:

$$\tilde{S}_{VV} = S_{VV} + \gamma \cdot S_{HV} \quad (1)$$
$$\tilde{S}_{HH} = S_{HH} + \gamma \cdot S_{VH} \, . \quad (2)$$

Realizable scenarios would be transmitting on one or both pols, corresponding to $\gamma \in \{0, 1\}$, respectively. However, any $\gamma \in [0, 1]$ is reasonable for data augmentation purposes. As cross-pol measurements degrade coherency of the formed images, we chose $\gamma \in \{0, 0.4\}$ as a compromise to balance the benefits of increasing sample diversity with preserving image quality. This results in four example SAR images for every combination of aspect angles.

3.4 Simulation Details

Every object is assumed to be made of perfect electric conducting material. This simplifies the simulations and is a reasonable assumption for most metallic objects, though it is straightforward to simulate other materials at modest increase in data generation time. The texture of the surfaces of each object are assumed to be slightly rough (up to a few centimeters of surface variability). Each simulation uses nominally 256 pulses at X-band frequencies, spanning 11° of synthetic aperture and having 2GHz of total bandwidth. The resulting SAR collections have a very fine spatial resolution of 0.075m in both range and cross-range directions.

3.5 SAR Image Formation and Processing

The raw Fourier data (known as "phase history" in the SAR community) acquired from the simulations are first Taylor-weighted as described in [30]. Adding such a taper to the data is used primarily to decrease the prevalence of sidelobes in the formed images. Next, images are formed via Backprojection, based on [31]. This implementation also projects the SAR images into the ground-plane. The complex-valued images are remapped via the Piecewise Extended Density Format, as implemented in the SarPy library [32]. Figure 4 shows examples of resulting images after this remapping.

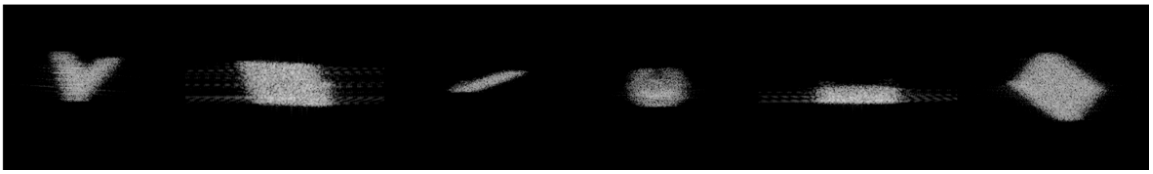

Figure 4: Example SAR images after full processing.

4. SAR Feature Rasterizer

In this section we describe the SAR Feature Rasterizer, which projects mesh-based scene representations into 2D feature maps that convey context about the structure and illumination of the scene. The Feature Rasterizer consists of three stages: Coordinate Transformations, Rasterization, and Shading. Standard computer graphics rendering follows analogous stages; however, each stage requires alterations either to accommodate backpropagation, account for

SAR's unique imaging geometry, or for the features we want to produce for the Neural Shader. The implementation of this Feature Rasterizer extends and is based on the PyTorch3D API [33].

4.1 SAR Imaging Geometry and Coordinate Transformations

SAR's unique imaging geometry makes adapting methods in differentiable rendering designed for EO imagery non-trivial. For rasterization-based methods, imaging geometry is controlled primarily through the coordinate transformation sequence. Consequently, understanding the geometry differences between these imaging modalities is necessary to design a coordinate transformation sequence.

In EO imagery, distance from the sensor affects an object's perceived size. This results in the "oblique aerial" imaging geometry, seen in Figure 6. In computer graphics, this effect is known as "perspective," and is typically accounted for via a "perspective projection" during the conversion to Normalize Device Coordinates (NDC) [34]. However in SAR imagery, distance from the sensor determines which range-bin reflected energy funnels into, and thus affects an object's perceived *position* in range.

This leads to a natural "slant-plane" imaging geometry for points on the ground. SAR images may be projected into the "ground-plane" as part of image formation, nominally mapping image pixels to equally spaced points along the ground. Distance to the sensor is also influenced by an object's height. Consequently, imaging objects of variable height will yield sampling distortions known as "foreshortening" and "layover" [30, 35].

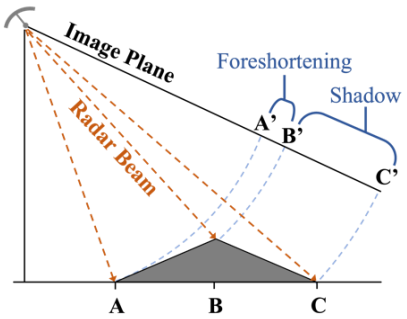

Figure 6: Graphic representation of foreshortening phenomenon.

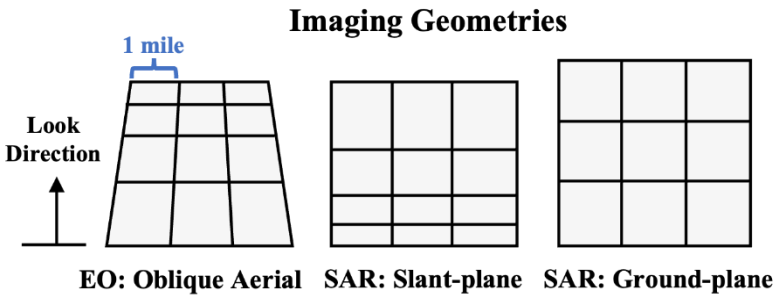

Figure 5: Comparison of imaging geometries in EO and SAR when imaging points on the ground.

The goal of the coordinate transformation stage is to transform mesh coordinates appropriately to get the desired SAR ground-plane imaging geometry, including foreshortening/layover effects. Coordinate Transformation sequences must start in world space coordinates and end in NDC. The primary sequence is described in Table 1 and illustrated in Figure 7. If the ground-plane projection were to be omitted, the result would be a SAR slant-plane imaging geometry instead. An equivalent EO "oblique aerial" imaging geometry would require omitting both the range transform and ground-plane projection, as well as changing the orthographic projection to a perspective projection.

Transform Name	Input Coordinates	Output Coordinates	Description
View Transform	World Space	View Space	Coordinates relative to camera's point of view
Range Transform	View Space	Range Space	z-coordinates (depth, or distance from camera) become new y-coordinates
Ground-plane Projection	Range Space	Ground Space	Stretches perceived y-dimension s.t. each pixel maps to consistent-sized cell on ground
Orthographic Projection	Ground Space	NDC Space	Without perspective projection (distance from camera does not affect perceived size)

Table 1: Coordinate transformation details for SAR ground-plane imaging geometry.

Figure 7: Coordinate transform sequence for SAR ground-plane imaging geometry.

4.2 Rasterizing and Shading Feature Maps

Soft Rasterization. The rasterization method is based on the "soft rasterizer" of [22]. The method in [22] blends facet contributions for pixels near edges to mitigate the discontinuities that would arise if trying to backpropagate through a standard rasterization operator. Soft rasterization will yield slightly distorted images but allows backpropagating more useful gradients. Examples of artifacts from soft rasterization are shown in Figure [8].

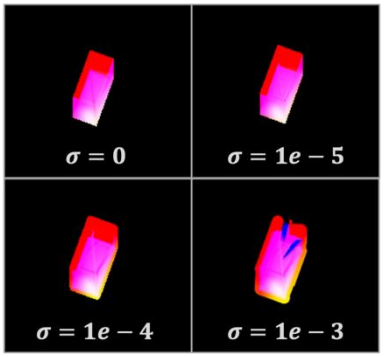

Figure 8: Visual artifacts from soft rasterization using varying levels of blending radius σ.

Silhouette and Surface Normal Features. Rasterizing with our custom coordinate transform sequence produces the fragments necessary to shade the first two feature maps. The first feature map is a basic silhouette of the object. This can be produced by making a mask out of fragments that have no corresponding facets. The second feature we want is pixel-wise surface-normal information, relative to the direction of the sensor and illumination. For monostatic imagery, this is given as:

$$d = -\hat{q} \cdot \hat{n}\,, \tag{3}$$

where $\hat{q}$ is the direction of the illumination, $\hat{n}$ is the per-pixel surface-normal of the object, and $\cdot$ denotes the inner-product.

Alpha Feature with SAR Self-Shadow. The last feature map to shade is an alpha channel. Deriving the alpha channel is complicated by SAR's self-shadowing effect. To address this, we create a shadow mask. The shadow mask can be predicted by rasterizing two reference views. In both views the camera is facing top-down. The first view involves no additional projections, while the second view includes a foreshortening projection. Both reference views are illustrated in Figure 9.

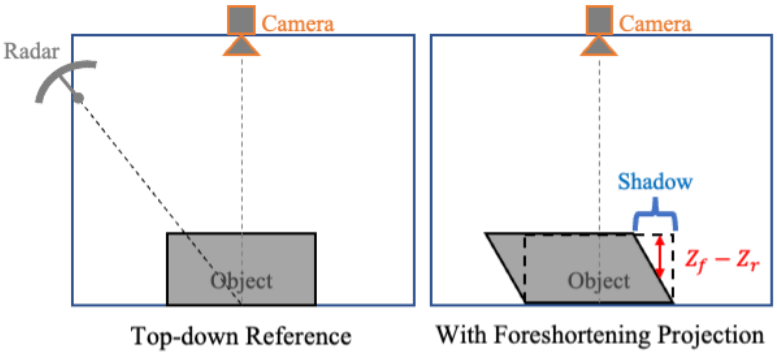

Figure 10: SAR shadow estimation reference views.

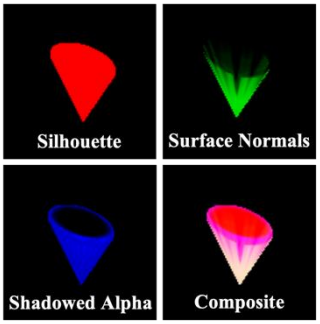

Figure 9: Shaded feature maps.

The resulting rasterization fragments can be used to derive a per-pixel height map of the object for both views. Pixels where height *decreased* following the foreshortening projection constitute the shadow mask. We make the shadow soft instead of hard so that it produces more useful gradients during backpropagation. The shadow mask is computed as:

$$M_{shadow} = tanh(\beta \cdot (Z_f - Z_r)) \,, \tag{4}$$

where Z_r and Z_f are depth buffers of the reference views with and without foreshortening, respectively. Scaling parameter β controls shadow sharpness and has a default value of 6.0. Since $Z_f \geq Z_r$, pixel values of M_{shadow} will be between [0,1). Finally, the shadow can be added to the standard alpha channel α by:

$$\alpha^* = \alpha \odot (1 - M_{shadow}) \,. \tag{5}$$

An example of all three shaded feature maps is shown in Figure 10.

5. Neural Shader

The Neural Shader is trained to predict realistic SAR shading/scattering effects conditioned on the features output by the Feature Rasterizer. Given this is effectively a paired image-to-image translation problem, we choose a model based on Pix2Pix [36]. While some newer approaches have outperformed Pix2Pix, we choose it for its simplicity in this proof of principle work. The most noteworthy deviation made is to use batch renormalization [37] instead of instance

normalization. Batch renormalization addresses the original issues that motivated the authors to use instance normalization, while improving consistency and accuracy.

5.1 Training

The feature map inputs used in training involve minor data augmentation via differing values for blending radius $\boldsymbol{\sigma}$ (0, 1e-5, 1e-4, 1e-3), like those seen in Figure 8. This adds diversity to the training data and should help the CGAN be more robust to distortions caused by soft rasterization. As described in subsection 3.3, the SAR images are augmented via differing polarization combinations.

The CGAN is trained for a total of 225 epochs, each of which is 1080 iterations, with a batch size of one. Both the generator and discriminator are trained with the Adam optimizer [38]. Learning rates start at 2e-4 and decay linearly every epoch to reach 2e-5 by the end of training.

5.2 Results

Figure 11 shows example images generated by the fully trained CGAN. Although the most important results are those for the test object, we include predictions from objects in the validation and training sets for additional comparison.

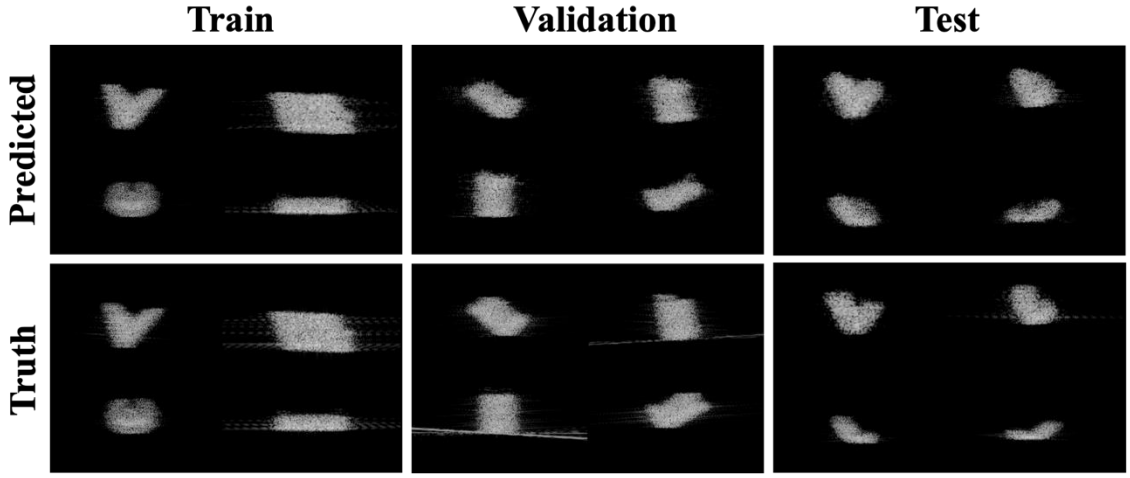

Figure 11: Example images generated by the fully trained CGAN.

The train, validation and test object sets are compared in terms of the loss produced when generating images of each with the CGAN, in Table 2. The L_{cGAN} and L_{L1} losses in Table 2 are the same as those found in [36], with the same recommended weighting of 1 and 100, respectively. L_{cGAN} is a measure of how well the generator fools the discriminator, while L_{L1} is the scaled L1 distance between predicted and truth images.

We also include an energy-normalized metric, L_{L2}^{*}, to account for bias due to varying object sizes. The sum-squared-error of the residual is normalized by the energy of the label image, and this ratio is converted to a dB scale. The metric is defined in equation 6, where norms are over pixels.

$$L_{L2}^{*} = 10 \log_{10} \left(\frac{\|y - \hat{y}\|^2}{\|y\|^2} \right) . \quad (6)$$

When using the energy-normalized metric L_{L2}^{*}, we can see that loss is lowest on the training set as expected, with an increase of +3.87 dB loss on the validation set, and an additional +0.64 dB loss on the test set. This is the pattern of losses we would expect, which suggests the CGAN suffers a modest amount of over-fitting to the training data and a minor amount of over-tuning to the validation data.

Data Set	L_{cGAN}	L_{L1}	L_{total}	L_{L2}^{*}
Train	0.64	1.74	2.38	-10.86
Validation	0.60	2.90	3.50	-6.99
Test	0.61	1.52	2.13	-6.35

Table 2: Average loss comparison by object set for trained CGAN.

6. 3D Reconstruction Experiment

The goal of this demonstration is to test if the proposed differentiable renderer can be used to reconstruct a mesh of a new object, given only SAR images of the object at various known aspect angles.

6.1 Losses

This subsection details losses used for the 3D reconstruction experiments. The total loss is aggregated from five sources, with weights that were tuned based on reconstruction performance using the validation object, and is given as:

$$L_{total} = L_p + 1.5L_l + 0.02L_n + 0.03L_e + 0.4L_f . \quad (7)$$

The first term uses pixel-wise mean-squared-error to compare the high-fidelity simulated SAR images (at known aspect angles) to output predictions of the differentiable SAR renderer (at the same aspect angles). Despite the notorious fickleness of SAR speckles, we found this loss still works reasonably well for demonstration purposes and is simple. The other loss terms are described next.

Laplacian. We use the "uniform" weighting of Laplacian mesh regularization based on [39] and [40], which discourages sharp vertex corners. If y_i represents the ith vertex of the mesh, S_i is the set of vertices it shares edges with, and $\mathbb{E}[S_i]$ is its centroid, then the Laplacian regularization loss can be computed as:

$$L_{laplacian} = \sum_i \|\mathbb{E}[S_i] - v_i\| . \quad (8)$$

Normal Consistency. We use normal consistency mesh regularization as implemented in [33], which encourages adjacent faces to have similar surface-normal directions. If $\hat{n}_{a_k}$ and $\hat{n}_{a_k}$ are

surface normal vectors of faces sharing the kth edge, then the normal consistency regularization loss is given as:

$$L_{normal} = \sum_{k} 1 - \hat{n}_{a_k} \cdot \hat{n}_{b_k} \,. \tag{9}$$

Edge Length. Edge length regularization is meant to encourage mesh edges to be of similar length. This has the effect of encouraging vertices to be uniformly distributed along the mesh's surface. If γ is the set of the mesh's edge lengths, the edge length regularization loss can be written as:

$$L_{edge} = \sum_{k} (\mathbb{E}[\gamma] - \gamma_k)^2 \,. \tag{10}$$

Floor-Plane. Since only overhead views are realizable, vertices on the underside of the mesh will not receive guiding gradient information from the Primary loss. However, it is known as a prior that such vertices are likely to reside on the scene's floor plane, as opposed to being suspended in air.

We created the 'floor-plane' regularizer to encourage this by first rasterizing the object from a bottom-up view. The depth buffer Z from the raster fragments can be used to derive a 2D height map H of the underside of the mesh. We penalize the heights as:

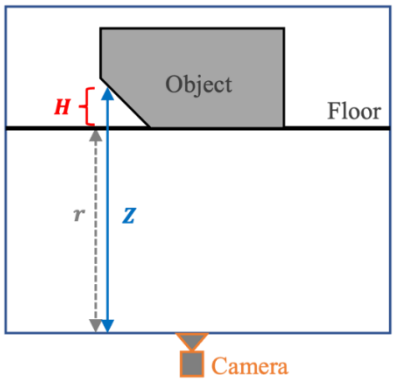

Figure 12: Geometric representation of floor-plane regularization.

$$H = r - Z \tag{11}$$

$$L_{floor} = \mathbb{E}[H^2] \,, \tag{12}$$

where r is the distance between the camera and the scene's floor-plane. It is assumed that the floor-plane has a height of zero. Figure 12 provides a geometric interpretation of the quantities in equation 11.

6.2 Mesh Optimization Loop and Settings

The 3D object reconstruction is done in two levels, to reconstruct the mesh in a coarse-to-fine manner. Both levels use the stochastic gradient descent optimizer with a learning rate of 1.0, momentum of 0.9, and dampening of 0.9. Each level updates the mesh iteratively as follows:

1. A label SAR image of the target object and its aspect angles are chosen.
2. The estimated mesh is rendered via our renderer at the same aspect angles.
3. The total (regularized) loss is computed by equation 7, using the prediction rendered by the model and the chosen label SAR image.
4. The mesh is updated based on backpropagated gradients from the loss.

During the first level of mesh-fitting, the estimated mesh is initialized with a large dome shape of 42 vertices and 80 triangle faces, as seen in Figure 13. We choose a low vertex count to help the optimization loop avoid local minima early on. The soft rasterizer blurring coefficient is σ=1e-3. The mesh at this level is updated for 1080 iterations using a batch size of two.

After the first level of mesh-fitting, the mesh estimate is up-sampled by inserting new vertices in the middle of each edge, resulting in a mesh with 162 vertices and 320 triangle faces. This up-sampled mesh initializes the second level. The soft rasterizer blurring coefficient is also lowered to σ=2.5e-4. The mesh is updated for 540 iterations using a batch size of four. The larger batch size of the second level reduces variance of the gradients, helping the mesh fine-tune.

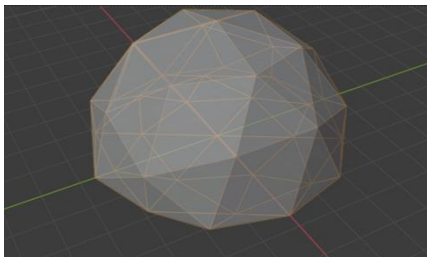

Figure 13: Initial dome mesh.

6.3 Results

The experiment is performed on three objects: Large Ramp, Long Cuboid, and Elliptic Cone. The primary object of interest to test is the Elliptic Cone because it was reserved exclusively for testing. Results on the Long Cuboid (validation object) and Large Ramp (training object) serve as points of reference to understand how much the mesh-fitting performance may be impacted by hyperparameter over-tuning and Neural Shader overfitting, respectively. The experiment is run five times for each object at different random seeds. The results for each object shown in Figure 14 are sourced from the median-scoring trial.

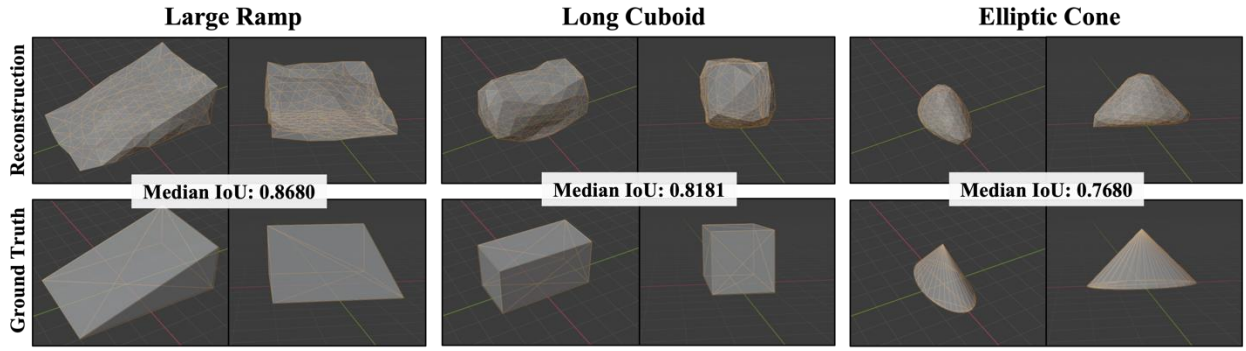

Figure 14: Meshes from median-scoring trial of 3D reconstruction for Large Ramp (train), Long Cuboid (validation), and Elliptic Cone (test).

For quantitative evaluation, we use voxelized intersection over union (IoU), which is a common metric for 3D reconstruction tasks [22]. The IoU scores of the three objects trend similarly to what was observed with the L_{L2}^{*} metric that compared rendering accuracy in Table 2. It is worth noting that the L_{L2}^{*} score for the Large Ramp by itself was -12.32 dB, which is better than the average amongst all training objects (-10.86 dB). Similar to in the analysis of subsection 5.2, the results suggest the Neural Shader suffers a modest amount of over-fitting to training data and a minor amount of over-tuning to validation data.

7. Future Work and Conclusion

The method proposed in this work has been successful as a proof of principle. However, there are areas for improvement that should be addressed in the future to help this become a more pragmatic tool for real-world use. This includes support for multiple materials and object segments, support for multistatic modalities, and validation on real world data. There are also open research problems related to differentiable rendering for SAR that should be the subject of future work. These include additional applications and an alternative approach based on SAR raw signal simulators.

Future Improvements. In the method's current form, objects are assumed to be of a uniform material, and changing the assumed material requires new corresponding weights for the Neural Shader. Support for multiple material types, as well as backpropagation to material properties, would broaden the tool's utility to other differentiable SAR rendering applications. This could manifest itself as additional feature maps output by the Feature Rasterizer to convey material property context to the Neural Shader. The current method also only supports one object segment (i.e., one mesh) at a time. Support for multiple segments would allow handling of background clutter, which is typically present in real SAR images, and may make support for multiple materials easier to implement.

This work also only considered monostatic SAR collections. Generalizing the approach to accommodate multistatic SAR modalities (where transmitter and receiver are *not* co-located) would expand the coverage of SAR collection modes.

Lastly, the SAR data used to train and evaluate our method was synthetically generated. The cost of collecting large quantities of measured data make it prohibitively expensive to use for training. However, future experiments should at least use measured data for validation and test objects.

Future Directions. We chose the task of 3D object reconstruction for demonstration purposes, but future work should investigate other applications enabled by differentiable rendering. Particularly, we would like to see this applied to the task of generating adversarial examples subject to 3D shape or material constraints. This task may be useful on its own for assisting in vehicle design or augmentation, but also enables the downstream tasks of geometrically-constrained data augmentation and adversarial training for deep neural networks.

An approach based on SAR raw signal simulation would be fully coherent and would allow the user to employ an image formation algorithm of choice. It would also eliminate the need to train a CGAN for shading, which is cumbersome because of the large number of high-fidelity SAR images needed to train it. However, such an approach would inevitably be computationally more costly than the one proposed in this paper.

Conclusion. In this work, we described an approach for differentiable rendering of SAR imagery which is a composition of techniques from computer graphics and neural rendering. We presented proof-of-concept results on the task of 3D object reconstruction and discussed the approach's current limitations. To the best of our knowledge, this is the first successful demonstration of differentiable rendering for SAR-domain imagery, which we hope will serve as a useful starting point for others seeking to use differentiable SAR rendering in their research.

References

1. Stimson, G.W.: Introduction to Airborne Radar, chap. 1: Overview, p. 13. SciTech Publishing, 2nd edn. (1998)
2. Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., Gaidon, A.: Differentiable rendering: A survey. CoRR abs/2006.12057 (2020)
3. Franceschetti, G., Migliaccio, M., Riccio, D.: The SAR simulation: an overview. In: 1995 International Geoscience and Remote Sensing Symposium, IGARSS '95. Quantitative Remote Sensing for Science and Applications. vol. 3, pp. 2283–2285 vol.3 (1995)
4. Auer, S., Bamler, R., Reinartz, P.: Raysar – 3D SAR simulator: Now open source. In: Proceedings of IEEE International GeoScience and Remote Sensing Symposium (2016)
5. Hammer, H., Kuny, S., Schulz, K.: Amazing SAR imaging effects - explained by SAR simulation. In: Proceedings of 10th European Conference on Synthetic Aperture Radar (2014)
6. Balz, T.: Real-time SAR simulation of complex scenes using programmable graphics processing units. In: Proceedings of International Society for Photogrammetry and Remote Sensing (2006)
7. Balz, T., Hammer, H., Auer, S.: Potentials and limitations of SAR image simulators – A comparative study of three simulation approaches. ISPRS Journal of Photogrammetry and Remote Sensing 101, 102–109 (2015)
8. Phong, B.T.: Illumination for computer generated pictures. Communications of the ACM 18, 311–317 (1975)
9. Ulaby, F.T., Moore, R.K., Fung, A.K.: Microwave Remote Sensing: Active and Passive, vol. Volume III: From Theory to Applications, Microwave Remote Sensing: Active and Passive, vol. III: From Theory to Applications. Artech House (1986)
10. Hao, L., Chou, R.C., Lee, S.W.: Shooting and bouncing rays: Calculating the RCS of an arbitrarily shaped cavity. IEEE Transactions on Antennas and Propagation 37(2), 194–205 (1989)
11. Andersh, D.J.: Xpatch 4: the next generation in high frequency electromagnetic modeling and simulation software. In: Proceedings from IEEE International Radar Conference (2000)
12. Zhang, R., Hong, J., Ming, F.: CASpatch: A SAR image simulation code to support ATR applications. In: 2009 2nd Asian-Pacific Conference on Synthetic Aperture Radar. pp. 502–505 (2009)
13. Pritchard, A., Ringrose, R., Clare, M.: Prediction of SAR images of ships. In: IEE Colloquium on Radar System Modelling (Ref. No. 1998/459). pp. 2/1–2/5 (1998)
14. Smit, J.C.: Sigmahat: A toolkit for RCS signature studies of electrically large complex objects. In: 2015 IEEE Radar Conference. pp. 446–451 (2015).
15. Bingle, M., Garcia-Aguilar, A., Illenseer, F., Jakobus, U., Lezar, E., Longtin, M., van Tonder, J.: Overview of the latest electromagnetic solver features in FEKO suite 7.0. In: 2015 31st International Review of Progress in Applied Computational Electromagnetics (ACES). pp. 1–2 (2015)
16. Garrido, E.E., Jenn, D.C.: A matlab physical optics RCS prediction code. ACES Newsletter 15(3) (2000)
17. Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J.M., Niessner, M., Pandey, R., Fanello, S.R., Wetzstein, G., Zhu, J.,

Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D.B., Zollhofer, M.: State of the art on neural rendering. Computer Graphics Forum 39, 701–727 (2020)
18. Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P.P., Tretschk, E., Wang, Y., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., Simon, T., Theobalt, C., Niessner, M., Barron, J.T., Wetzstein, G., Zollhofer, M., Golyanik, V.: Advances in neural rendering. CoRR abs/2111.05849 (2021)
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 405–421. Springer International Publishing, Cham (2020)
20. Kato, H., Ushiku, Y., Harada, T.: Neural 3D mesh renderer. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
21. Petersen, F., Bermano, A.H., Deussen, O., Cohen-Or, D.: Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. CoRR abs/1903.11149 (2019)
22. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In: Proceedings of IEEE/CVF International Conference on Computer Vision (2019)
23. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. ACM Transations on Graphics 37(6), 1–11 (2018)
24. Nimier-David, M., Vicini, D., Zeltner, T., Jakob, W.: Mitsuba 2: A retargetable forward and inverse renderer. ACM Transations on Graphics 38(6), 1–17 (2019)
25. Wang, L., Xu, X., Yu, Y., Yang, R., Gui, R., Xu, Z., Pu, F.: Sar-to-optical image translation using supervised cycle-consistent adversarial networks. IEEE Access 7, 129136–129149 (2019)
26. Li, Y., Fu, R., Meng, X., Jin, W., Shao, F.: A SAR-to-optical image translation method based on conditional generation adversarial network (CGAN). IEEE Access 8, 60338–60343 (2020)
27. Yang, X., Zhao, J., Wei, Z., Wang, N., Gao, X.: Sar-to-optical image translation based on improved CGAN. Pattern Recognition 121, 108208 (2022)
28. Niu, S., Qiu, X., Lei, B., Fu, K.: A sar target image simulation method with dnn embedded to calculate electromagnetic reflection. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 14, 2593–2610 (2021)
29. Community, B.O.: Blender - A 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2021), http://www.blender.org
30. Goodman, R., Carrara, W., Majewski, R.: Spotlight Aperture Radar: Signal Processing Algorithms. Artech House (1995)
31. Gorham, L.A., Moore, L.J.: SAR image formation toolbox for matlab. In: Proceedings of SPIE Defense, Security, and Sensing: Algorithms for Synthetic Aperture Radar Imagery XVII (2010)
32. Agency, N.G.I.: SarPy. National Geospatial-Intelligence Agency, Springfield, VA, 1.1 edn. (2021), https://github.com/ngageoint/sarpy
33. Johnson, J., Ravi, N., Reizenstein, J., Novotny, D., Tulsiani, S., Lassner, C., Branson, S.: Accelerating 3D deep learning with PyTorch3D. In: SIGGRAPH Asia 2020 Courses. SA ’20, Association for Computing Machinery, New York, NY, USA (2020).
34. de Vries, J.: Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion. Kendall & Welling (June 2020), https://learnopengl.com/

35. Goodman, R., Carrara, W.: Handbook of Image and Video Processing, chap. 10.1: Synthetic Aperture Radar Algorithms, pp. 749–769. Academic Press, 1st edn. (2000)
36. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2017)
37. Ioffe, S.: Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Proceedings of 31st International Conference on Neural Information Processing Systems (2017)
38. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of 3rd International Conference on Learning Representations (2015)
39. M. Desbrun, M. Meyer, P.S., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: 26th International Conference on Computer Graphics and Interactive Techniques (1999)
40. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Laplacian mesh optimization. In: International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (2006)