Synthetic Aperture Radar Ray Tracer

Benji Northrop
Benji.northrop@gmail.com

ABSTRACT

Ray tracing normally applies to the visible light spectrum, but it can also be used to simulate other waveforms, such as radar. Changing the wavelength results in different ray behavior as previously known diffuse reflectors become specular reflectors. Given a 3D model, the ray tracer can produce either a realistic image, or a stylized image representing Synthetic Aperture Radar (SAR).

1 INTRODUCTION

Ray tracing is a rite of passage for computer graphics students, offering a robust challenge in linear algebra, physics, and computer programming. Despite these challenges, the results can be breathtaking. As a result, numerous tutorials and resources have been released to help students and self-learners to build their own ray tracer from the ground up. Many take these projects and extend them to learn more advanced techniques, such as object/mesh loading, parallel processing, and other optimizations. We aim to expand this project by introducing a new application for ray tracing mechanics.

1.1 Related Work in SAR modeling

Real SAR imagery is readily available through various environmental agencies (such as TerraSAR-X) [1] and is often used for environmental research. However, simulations are not easily accessible. The Air Force Research Laboratory [2] has a simulation and modeling architecture that contains a SAR module, but it is not accessible to the general public. The only simulator known to us is known as *RaySAR*, an open-source simulator created by *Auer et al.* [3]. This program is easily downloadable and may be run so long as the user has access to MATLAB and can provide the required data to begin modeling an object. We found that this requirement is beyond the scope of many learners and thus becomes a barrier to entry.

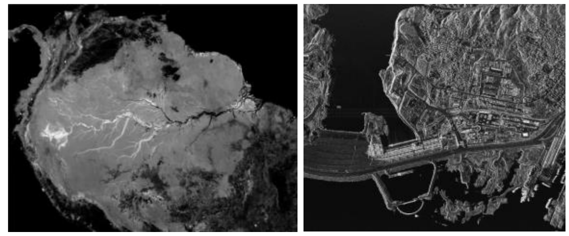


Figure 1. Real examples of SAR imagery of the Amazon (left), and an urban area (right). The white highlights in the left image show areas of high moisture, resulting in stronger returns than dry vegetation areas. Source: Terra-X

1.2 Our Approach

As a result, we propose an alternative solution that creates a simplified SAR imagery system that may be easily incorporated into most basic ray tracer programs. The goal is that this simulator would be provided a snapshot of a scene, with a camera position, scene, and desired resolution, and the program will generate the desired optical or SAR image.

Potential applications for this program are in modern terraforming or military games, where satellite and SAR imagery provides information on the state of a planet's water system, vegetation, and other natural phenomena, or an airborne radar can provide imagery as part of an intelligence briefing (Figure 1).

To achieve our goal, we ensure the implementation remains lightweight by limiting dependencies to two header-only libraries.

1.3 Overview

In section 2, we will cover a brief synopsis of the history and physics of SAR, as well as its similarities to ray tracing. Section 3 will discuss our program's architecture and pipeline. Section 4 covers the programs current limitations and sections 5 and 6 will conclude with our results and future work plans.

2 SAR Overview

SAR's history begins with the side-looking radars of the 1950s. A radar beam, shaped like a cone from the antenna, illuminates a patch of the ground and measures the magnitude of the return (Figure 2). This *footprint, S,* is proportional to the wavelength, λ, antenna size, L, and range to the target, R. See equation 1:

$$S = \frac{\lambda}{L} R \; [m] \tag{1}$$

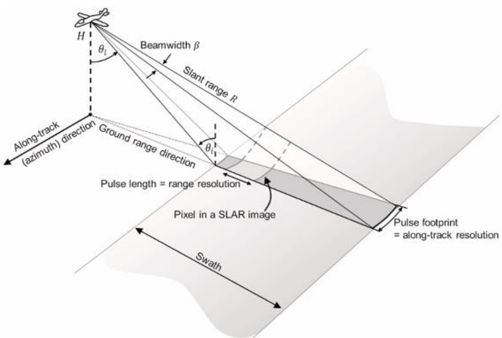


Figure 2. An example of airborne SAR geometry.

For example, an X-band radar with wavelength 3cm from an aircraft at 3,000m and an antenna length of 3m would achieve a 60m azimuth resolution. However, a satellite at 800km would require an 800m long antenna to achieve the same resolution [4].

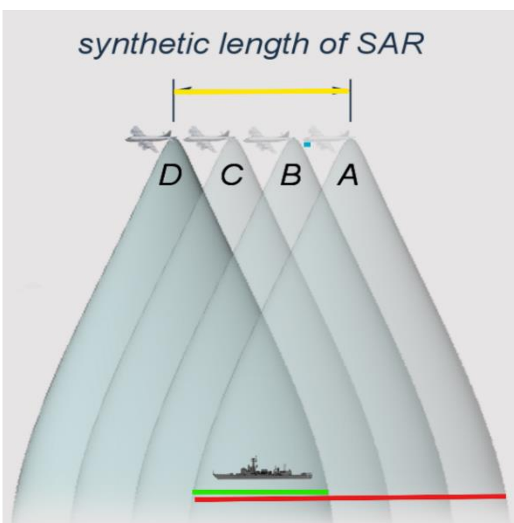


Figure 3. The red line shows the original beamwidth, and the green line is the synthesized beam width. The blue dot is the actual radar aperture size, and yellow is the synthetic aperture size.[5]

In 1952, Goodyear Aircraft Corporation engineer Carl Wiley discovered a relationship between the along-track coordinate of an object and the instantaneous doppler shift of its return signal. As a result, one could use frequency analysis to develop a much higher resolution along the track of the radar image. This became what we know as "synthetic aperture," because the aircraft's track across the ground and multiple samples gave the equivalent results of a single long antenna (Figure 3).

2.1 Correlating SAR and ray tracing

The SAR image model has many similarities to ray tracing. Both involve a light source (the antenna), a camera (the receiver), and a scene. In fact, SAR is a simpler model, as there is only one light source which is collocated with the camera. The result is equivalent to freezing the aircraft in time, sending a radar pulse through every pixel of a viewport into a scene, and accumulating the results.

2.2 Adjusting Models for Radar

Both visible light and radar are part of the electromagnetic spectrum, with visible light ranging from 400nm to 700nm and microwave bands between 1mm and 1m. Typical wavelengths used for SAR imaging are in the X-, C-, and L- band regions, which are approximately 3cm, 6cm, and 23cm long, respectively (Figure 4).

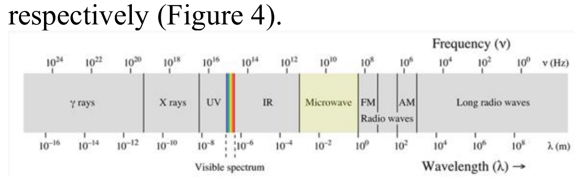


Figure 4. Electromagnetic spectrum.

The electromagnetic reflectivity of an object depends on the material's smoothness and its dielectric constant, which determines how permeable a material is to electromagnetic waves. Bennett and Porteus [6] confirmed this correlation, and we have two possible criterion to use to determine it. The Rayleigh criterion determines smoothness by comparing the root mean square height variation of the material to the wavelength and its angle of incidence.

$$\Delta h < \frac{\lambda}{8 cos\theta} \tag{2}$$

The Fraunhofer criterion reduces the smoothness threshold and introduces a middle zone for slightly rough surfaces that exhibit both specular and diffuse behaviors (Figure 5). In our implementation, each ray sample stochastically determines whether to compute the ambient or specular term based on a uniform probability distribution. This method ensures that the resulting reflections accurately capture the surface's mixed behavior, blending diffuse and specular contributions.

$$\Delta h < \frac{\lambda}{32 cos\theta} \quad (3)$$

$$\Delta h > \frac{\lambda}{2 cos\theta} \quad (4)$$

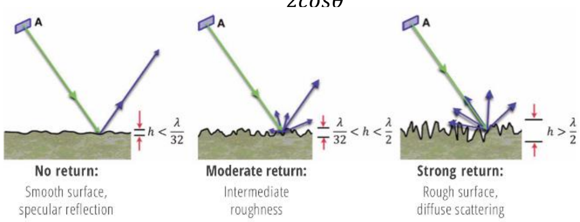


Figure 5. The various scatter patterns of radar energy.

Since visible wavelengths are anywhere between 400 and 700 microns, nearly all materials save objects such as highly polished metal and mirrors are diffuse reflectors. As a result, most 3D models for graphics and animation do not require height variance data. Instead, they use basic shapes and texture mapping to optimize rendering speed. For example, a rough brick wall appears textured, but the underlying data is a smooth, flat polygon with an RGB texture map.

In our program, we simulate X-band radar, with a wavelength of 3cm. Therefore, smooth surfaces will have approximately 1mm of height variation, rough surfaces will have 1.5cm of height variation, and slightly rough materials are anything in between.

3 System Architecture

3.1 Ray Tracer

Our ray tracer is based on Peter Shirley's book series *Ray Tracing in a Weekend* [8] as it includes an easy-to-follow tutorial and maintains a relatively simple code-base. Once one completes the series, they'll have some key features in their ray tracing system to include: Monte Carlo sampling methods, importance sampling to ensure enough secondary rays reach towards a light source, and a boundary volume hierarchy to optimize ray-object intersections.

Some challenges users will discover with the system is the lack of matrix multiplication commonly found in graphics APIs, so we must deduce the inverse transformation matrix in order to transform the instance of a ray in model coordinates to determine a hit. Therefore, the system provides a means of rotating about the y axis, but the student is left to determine how to transform the other axes.

Other additions we made to the ray tracer include an .obj and .mtl file importer, triangle mesh class, and scaling features – allowing us to render more complex scenes.

3.2 Object Loader

The object loader API we chose is TinyObjLoader [9], a lightweight, header-only .obj file loader that very easily integrates into the program. It converts the obj files into vertices, normals, and texture coordinates. Due to the wide variety of scale of the model coordinates in each individual .obj file, we normalize the vertices and normals to the coordinate range [-1, -1, -1] – [1, 1, 1] which give the user a known starting point to simplify model transformations.

If normals are not provided, they are automatically calculated by taking the cross-product of two edges, assuming a CCW vertex winding.

Wavelength is determined via a spectral map, which contains the average wavelength of an enumerated set of frequency bands. Currently, the supported bands are visible, X, C, and L.

Material roughness is contained within a roughness map, which contains a string name of each material, and a double that represents its root mean square height variance. This map could be easily abstracted to a separate file and imported into the program at startup.

The user includes the wavelength of the emitter in the load_model_from_file function call, and the program will automatically determine if it should use the .mtl provided lighting terms, or calculate new terms based upon the provided wavelength and material roughness value. This allows for dynamic allocation of material types depending on expected interaction patterns between the wavelength and the material.

4 Limitations

We have encountered some side effects with this system, that may be acceptable depending on the desired level of fidelity from the image. Currently, we use standard RGB values for both the ray tracer and SAR process. To transform to gray scale, we manually equalize each channel to create grayscale in the range [0.0, 1.0] as the albedo, which determines the amount of light that will be scattered or reflected. Our texture map is statically coded into our program for examples, but could be abstracted to a file for maintenance separately. Unfortunately, .mtl files do not support a height variance field, so this data would have to be imported separately and maintained based upon the specific name of the material.

5 Future Work

The first desired aspect of future work would be to incorporate parallel processing via CUDA or other hardware acceleration. The Rungholt town rendering, at a resolution of 1600 x 1000 pixels, took roughly 20 hours to render based. This would be required to incorporate into any game system.

We also intend to shift from standard RGB into a spectral ray tracer, so that the system can intuitively operate at a much broader spectral range. Real SAR allows for penetration of materials depending on the wavelength. For example, X-band radar cannot penetrate vegetation, so it is not able to see through forest canopies, but L-band radar, 8 times longer, can pass through the vegetation to the surface. It can even penetrate deeper into the ground and has been used to detect underground rivers.

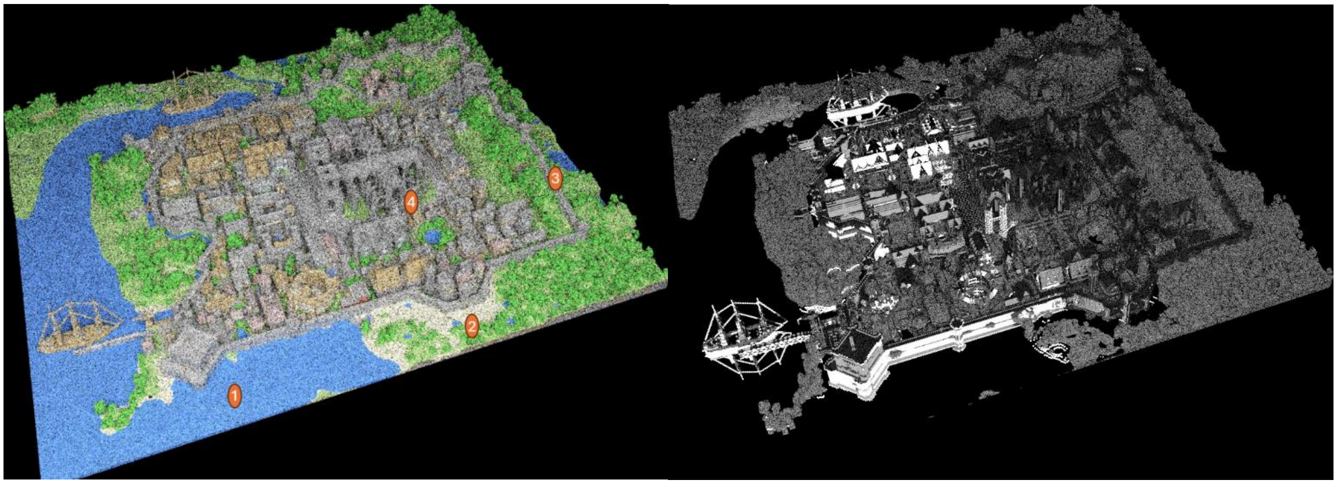
To include more wavelengths and step towards more physically realistic modeling, we would also need to incorporate a dropoff for the emitters.

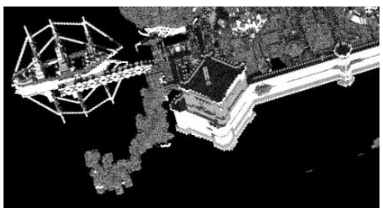
6 Results

This image was parsed from Minecraft to .obj using the Mineways tool [10] by McGuire et al., originally created by the user “kescha.” It contains 6.7 million triangles and took approximately 20 hours to render on a 13th Gen Intel(R) Core(TM) i7-13700H at 2.40 GHz and 32 GB of RAM. We rendered two images at 1600 x 1000 resolution, one using the visual light field and another using X-band radar. Each image samples 30 rays per pixel and allows up to 10 bounces of depth per ray.

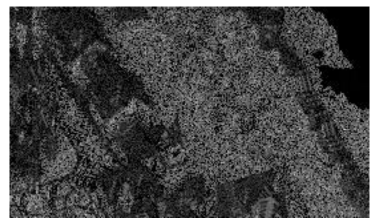
7 Conclusion

Our simplified SAR ray tracer provides an accessible tool for learners and developers interested in radar simulations. By incorporating lightweight dependencies and future support for parallel processing, this tool can bridge the gap between traditional ray tracing and SAR modeling.

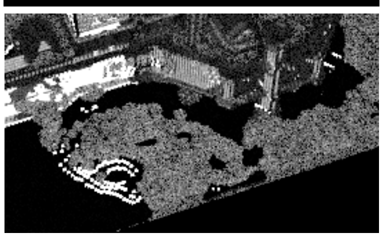




1 Notice the reflections in the SAR image that aren't in the visual



3 Vegetative canopies become indistinguishable blobs as X-band becomes uniformly scattered by the leaves.



2 Smooth sand, visible in light, becomes a reflector in X-band, causing highlights along the shoreline.

4 Towers and roads, with their different materials, have much higher contrast than their visual counterparts

8 References

[1] "TerraSAR-X and TanDEM-X - Earth Online," *earth.esa.int*. https://earth.esa.int/eogateway/missions/terrasar-x-and-tandem-x

[2] J. Zeh *et al.*, "Advanced Framework for Simulation, Integration, and Modeling (AFSIM) ," Air Force Research Laboratory, Wright-Patterson Air Force Base, Aug. 2016. Accessed: Dec. 11, 2024. [Online]. Available: https://imlive.s3.amazonaws.com/Federal%20Government/ID156700649043486100007468415908987547373/Attachment_1_-_AFSIM.pdf

[3] S. Auer, R. Bamler, and P. Reinartz, "RaySAR - 3D SAR simulator: Now open source," *elib (German Aerospace Center)*, Jul. 2016, doi: https://doi.org/10.1109/igarss.2016.7730757.

[4] F. Meyer, "The SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation," no. 1, Apr. 2019, doi: https://doi.org/10.25966/nr2c-s697.

[5] C. Wolff, "Radar Basics - Synthetic Aperture Radar," *Radartutorial.eu*, 2019. https://www.radartutorial.eu/20.airborne/ab07.en.html

[6] H. E. Bennett and J. O. Porteus, "Relation Between Surface Roughness and Specular Reflectance at Normal Incidence," *Journal of the Optical Society of America*, vol. 51, no. 2, p. 123, Feb. 1961, doi: https://doi.org/10.1364/josa.51.000123.

[7] Q. Manzoor, "Controlling Surface Roughness to Enhance or Degrade Image Appearance in Synthetic Aperture Radar (SAR) - DSIAC," *DSIAC*, 2020. https://dsiac.dtic.mil/articles/controlling-surface-roughness-to-enhance-or-degrade-image-appearance-in-synthetic-aperture-radar-sar/ (accessed Dec. 11, 2024).

[8] P. Shirley, "Ray Tracing in One Weekend Series," *raytracing.github.io*, Aug. 31, 2024. https://raytracing.github.io/

[9] "tinyobjloader/tinyobjloader," *GitHub*, Feb. 21, 2024. https://github.com/tinyobjloader/tinyobjloader

[10] M. McGuire, "Computer Graphics Archive," *casual-effects*, Jul. 2017. https://casual-effects.com/data/index.html (accessed Dec. 11, 2024).