General Purpose Radar Simulator based on Blender Cycles Path Tracer

Rômulo Fernandes da Costa, Diego da Silva de Medeiros, Raíssa Andrade, Osamu Saotome, and Renato Machado

***Abstract*— This paper proposes a general-purpose radar simulation tool based on the path tracer Cycles used in Blender, an open-source 3D computer graphics software. The scenarios, object's geometry, and materials can be defined within Blender. The propagation of waves in the scenery is simulated by the Cycles renderer, so that a second computational tool, such as Matlab or Octave, can be used for reconstructing the received signal. A simulated experiment using the proposed tool is presented. The results indicate that the tool has a great potential to be used in a variety of radar applications, including cross-section measurement, simulated radar data generation, and simulated synthetic aperture radar imaging.**

***Keywords*— Radar signals, 3D modeling, RGB channel modulation.**

I. Introduction

Computational simulation is an important tool for evaluating the propagation of radar signals through the environment. While it is possible to model radar targets as single point scatterers of fluctuating cross-section, in a real environment, signals are expected to be absorbed, refracted, and reflected multiple times before returning to the receiver antenna. These effects may lead to an overestimation of the returning signal power.

A common method for simulating the propagation of radar signals is through electromagnetism simulators, which numerically solve differential electromagnetic field equations. While this method is accurate, objects can be too complex to be modeled, which usually is very time consuming [1].

An alternative approach is the use of ray-tracing algorithms, which has been slowly becoming more commonplace in the last decade due to their increased availability and computational efficiency. As examples, ray tracing was used for wall-penetrating radar using RAPSOR [2], aircraft detection using Flames [1], and for urban environment SAR imaging using POV-Ray [3].

However, many ray tracing radar simulators are limited in evaluating material penetration and multipath propagation, limiting its use to line-of-sight applications [4]. Another common difficulty is creating accurate models of the terrain [3] or target geometries [5].

This paper presents a ray tracing radar simulation tool based on the cycles render engine, from the 3D modeling open-source software Blender. The RGB channels of each ray are modulated onto the desired radio frequency to properly simulate the effects of penetration and multipath propagation of radar waves. A Python script is used for rendering images for several successive subsections of the scene, containing power and range information for each subsection. The rendered images are then exported to a computational environment such as MATLAB, where the received signal is reconstructed by treating each pixel as a single point target.

II. Simulator Description

A. Path Tracing

Blender is an open-source computer graphics suite designed for 3D modeling, animation, and visual effects. Internally, Blender utilizes a rendering engine called Cycles, which simulates how light should propagate in a scene, based on physical principles. To accomplish this task, Cycles employs a path tracing algorithm, enabling it to accurately simulate diffuse and specular reflections, refraction, absorption, and transmission effects for each ray [6]. Materials can be configured using Blender's node-based interface, or programmed directly using Open Shading Language (OSL) [7].

Although Cycles was initially designed for simulating light propagation for 3D graphics, it can be exploited to simulate radar waves, by treating light emitters as transmitting antennas and Blender cameras as receiving antennas [5].

Thus, the object materials and emitters in Cycles can be reconfigured to encode amplitude and traveled distance for each ray in its RGB channels, as explained in the following sections, which allows Cycles to provide a measurement of incoming power and range for each pixel of the images captured by the Blender camera.

B. Range modulation in RGB channels

When rendering a scene in Cycles, rays are cast from the camera and are reflected in the scene until an emitter is reached or the maximum number of bounces (reflections) allowed per ray is reached.

The color of each ray is composed of three amplitude components, red (R), green (G) and blue (B). Between each bounce, the amplitude of all three components decays in a square-falloff over distance. Each time a ray interacts with the

Rômulo Fernandes da Costa and Diego da Silva de Medeiros, Graduate Program in Electronics and Computer Engineering, Aeronautics Institute of Technology (ITA), São José dos Campos-SP, e-mails: rfcosta@ita.br; medeiros@ita.br; Raíssa Andrade, Electronic Engineering course, Aeronautics Institute of Technology (ITA), São José dos Campos-SP, e-mail: raissa.andrade@ga.ita.br; Osamu Saotome, Department of Applied Electronics, Aeronautics Institute of Technology (ITA), São José dos Campos-SP, e-mail: osamu@ita.br; Renato Machado, Department of Telecommunications, Aeronautics Institute of Technology (ITA), São José dos Campos-SP, e-mail: rmachado@ita.br; This work was funded by the Brazilian Council for Scientific and Technological Development (CNPq) and IACIT, in the form of a research stipend for the first and second authors.

surface of a material, the amplitude of each color component is multiplied by the material's corresponding color amplitude.

Figure 1 exemplifies how rays are traced in a given scene within Cycles. All rays travel backwards from camera to emitter, as this provides a more efficient computation of the scene lighting. It must be noted that transmission rays (rays penetrating an object) are not depicted in Fig. 1 for clarity.

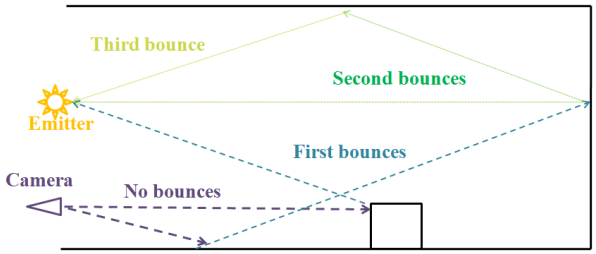

Fig. 1. Propagation of rays in a hypothetical scene in Cycles.

We can choose a color component as a referential for wave amplitude. In this paper, we chose the B component to be the amplitude referential, u_b. The amplitude u_b after n bounces in the scenery is given by

$$u_b = u_0 \frac{k_0}{r_0} \frac{k_1}{r_1} \cdots \frac{k_n}{r_n}, \tag{1}$$

where k_0 is a constant dependent on transmission antenna's gain, $k_1 \cdots k_n$ are constants dependent on the geometry and materials of the reflecting objects, and $r_1 \cdots r_n$ are the distances traveled between bounces. If considering for a single bounce and amplitude squared (i.e., power), Equation (1) takes the same format as the radar equation.

Cycles does not keep track of the total traveled distance by a ray since its emission. Instead, it only has direct access to the distance traveled since the last ray bounce, making it necessary to measure it through indirect means. One can encode the total traveled distance in the RGB channels, by imposing an additional exponential decay over distance on one of the color components [8]. This can be done by setting a color component of all simulated materials to have an exponential decay α_i, which is dependent on the distance traveled r_i from the last i-th bounce by:

$$\alpha_i = \exp\left(-\frac{r_i}{k_{dist}}\right), \tag{2}$$

where k_{dist} is a constant for the exponential decay. We imposed this exponential decay over the R channel. Its amplitude is given by u_r.

Due to the n bounces, the decay factors α_0, α_1 ...α_n are all multiplied over the R channel. The amplitude u_r is then given by

$$u_r = u_0 \alpha_0 \frac{k_0}{r_0} \cdots \alpha_n \frac{k_n}{r_n}. \tag{3}$$

Therefore, the ratio u_r/u_b falls exponentially over the distance traveled by the ray. With this approach, we can indirectly measure the total distance traveled by applying a logarithm over the ratio u_r/u_b as follows:

$$r_{sum} = \sum_{i=1}^{n} r_i = -k_{dist} \ln\left(\frac{u_r}{u_b}\right). \tag{4}$$

C. Emitter configuration

A problem that must be taken into consideration, especially in cluttered scenarios, is object occlusion. During rendering, objects closer to the camera may obstruct objects further away in the same line of sight, causing their range and amplitude measurements to occupy the same pixels in the image. This superposition can nullify the echoes of occluded objects.

To prevent this, the emitter was configured to progressively illuminate the scene in successive steps. This is done by restricting the emitter ray's length (emission radius) $r_{emitter}$ to

$$\Delta r_{frame} n_{frame} < r_{emitter} < \Delta r_{frame}(n_{frame} + 1), \tag{5}$$

where n_{frame} is the index of the current frame and Δr_{frame} is the step size per frame.

In this manner, objects in the same line of sight from the camera can be rendered without the problem of superposition in the camera, as the objects are rendered at different frames (provided that the step size is sufficiently small). During the signal reconstruction, the contribution of all rendered frames must be summed to account for the echoes of the entire scene.

The illumination procedure is exemplified in Fig. 2. In this Figure, a plane located at $z = 0$ is illuminated by a point emitter located over the plane, with the step size Δr_{frame} set to 2.5 blender units per frame. Due to the constraint applied over the emitter's ray length, a ring is projected over the plane.

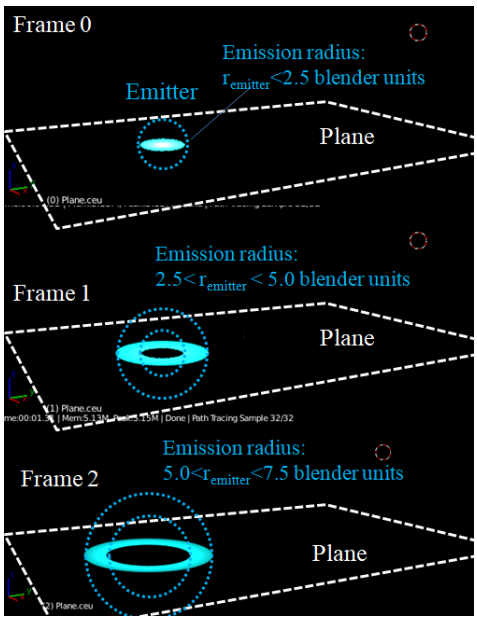

Fig. 2. Three rendered frames showing a emitter illuminating a plane in separate steps.

D. Materials' configuration

As previously explained, the material's color components must be modulated as a function of traveled distance from the last ray bounce. In addition to color manipulation, materials in Cycles must be reconfigured to show the same reflectivity, refraction, absorption, and opacity they would have in the simulated radar frequency. All of these properties are controlled by a subroutine called shader, which dictates how the material should propagate an incident ray.

The properties of internal attenuation, refraction, and reflectivity of a material can be described by the complex index of refraction as given by [9]:

$$\eta_c = \sqrt{\epsilon_r \mu_r} = \eta_{ref} + j\kappa_{ext}, \tag{6}$$

where ϵ_r and μ_r are the complex relative electrical permittivity and complex relative magnetic permeability, respectively, η_{ref} is the real part of the index of refraction, and κ_{ext} is the extinction coefficient, which indicates the amount of attenuation related to ray penetration in the material. Thus, the attenuation factor α_{trans} in ray amplitude when a ray travels through a material is given by

$$\alpha_{trans} = \exp\left(-\frac{2\pi\kappa_{ext}}{\lambda_0} z\right), \tag{7}$$

where λ_0 is the wavelength of the simulated radar signal, and z is penetration depth. This exponential decay needs to be implemented manually within the shader.

Moreover, the amplitude reflectance of a surface Γ for a normal incidence angle is given by:

$$\Gamma = \left|\frac{\eta_{ref1} - \eta_{ref2}}{\eta_{ref1} + \eta_{ref2}}\right|, \tag{8}$$

where η_{ref1} and η_{ref2} are the real index of refraction of the materials. Fortunately, the reflectance is calculated natively within Cycles for all angles of incidence, so it is not necessary to reimplement it in the shader.

Some works on the fields of Physics and Chemistry, as well as some studies in the topic of ground penetrating radar, provide tables of measured η_c [10] or ϵ_r [11] of a material in several wavelengths. Moreover, if a measurement table is unavailable for a certain frequency range, it is possible to use models to interpolate values of ϵ_r [12], such as the Debye relaxation model for dielectric materials [13].

In this study, a shader was developed for simulating the radar materials, using Blender's nodes editor. The shader is used to create absorption, refraction, and reflectivity, and therefore is used as the basis for all other materials in the simulation. Figure 3 shows a block diagram of the proposed shader.

The shader takes as input the material's index of refraction η_{ref}, a normalized extinction coefficient (to account for model scale), as well as specularity and roughness factors. Those input parameters can be either set manually or by using a Python script. The input interface of the shader can be seen in Fig. 4.

E. Rendering and signal reconstruction

Blender can save rendered frames in a dynamic range file format, such as Radiance HDR and OpenEXR files. These files can store amplitude values for each color channel in floating point, allowing the reconstruction of the signal captured by the Blender camera in a computational environment such as MATLAB.

Currently, the proposed simulator export amplitude and ray traveled distance data into four separate HDR files. One pair of files stores amplitude and ray distance for the current simulation, as defined by the user. A second pair stores amplitude and ray distance for a second pass of the renderer through the scene, assuming that δt_{slow} seconds have passed in-simulation. This second pass is used for calculating Doppler effects in the signal caused by movement of objects during the δt_{slow} interval.

The radar signal can be reconstructed by calculating a point target signal for each pixel of the rendered frames, and then summing the signals according to the superposition principle. Assuming that the transmitted waveform is a chirp, the point target signal for each pixel is given by:

$$s_{pixel} = A_0 w(\tau - \tau_0) \exp\left[j\phi(\tau - \tau_0)\right], \tag{9}$$

where A_0 is the signal amplitude, w is a windowing function, ϕ is the signal's phase, and τ is the time since the signal's emission from the transmitting antenna (fast time). Both w and ϕ depend on the simulated waveform, while

$$A_0 = u_b \tag{10}$$

and

$$\tau_0 = \frac{r_{sum}}{c} \tag{11}$$

are obtained from the Cycles rendering process. c is the speed of light in vacuum.

The Doppler frequency associated with the pixel is obtained by the variation in r_{sum} between simulations, and it is defined as:

$$f_{Doppler} \triangleq \frac{r_{sum}(t + \delta t_{slow}) - r_{sum}(t)}{\delta t_{slow} \lambda_0}. \tag{12}$$

To reconstruct the signal, it is necessary to compute a weighted sum of the contributions of every pixel in the rendered frames. The reconstructed signal is given by:

$$s_{sum} = \sum_{frames} \sum_{pixels} A_{pixel} s_{pixel}, \tag{13}$$

where A_{pixel} is the weight of the pixel in the weighted sum. Due to perspective projection, objects further away from the camera occupy a smaller area in the rendered image. Therefore it is necessary to compensate this effect by defining the weight as

$$A_{pixel} = r^2. \tag{14}$$

III. Results

A. Simulated environment

A simple scene was modeled to demonstrate the simulator capabilities, such as the simulation of multipath propagation, material penetration and Doppler effects in the signal. The

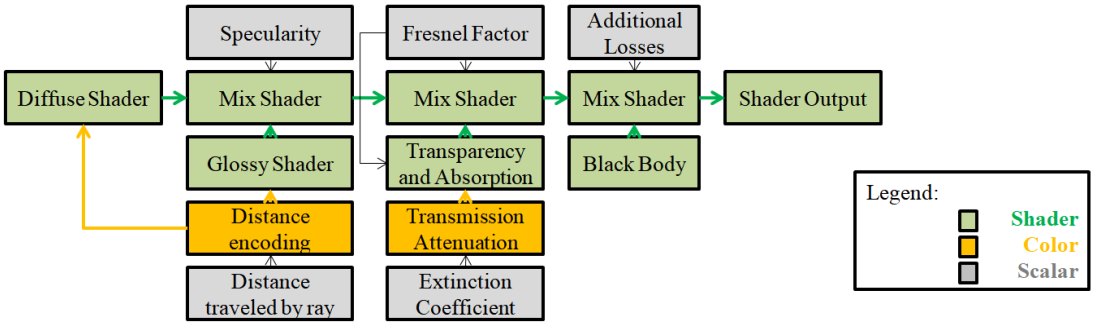

Fig. 3. Block diagram of the shader developed for radar simulations.

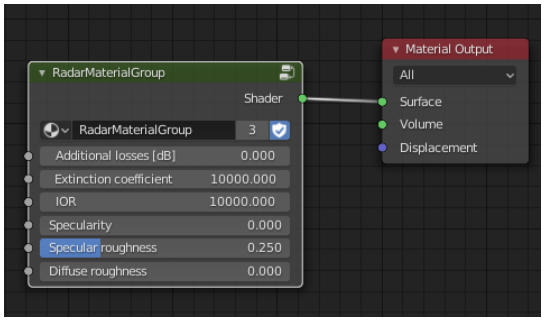

Fig. 4. Radar shader input interface.

scene comprises a monostatic radar and two large immobile objects, positioned at 1 km and 2.5 km away from the radar.

Between the two larger objects, a small object was placed as a target, moving away from the camera at a velocity of 10 m/s. The emitter and target were positioned 100 m above the surface, represented by a square plane.

The larger objects were assigned partially transparent materials to demonstrate material penetration effects modeled in the simulator. The target and floor were assigned highly reflective materials, also for demonstration purposes.

Figure 5 shows the considered scene. The scene was modeled on a scale of 1 Blender unit to 100 meters.

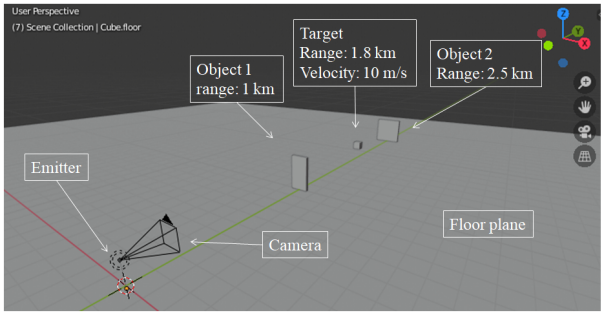

Fig. 5. Scene modeled for the experiment.

Table I lists the main parameters used in the simulation.

B. Simulation results

The scene was rendered in 21 frames, with each frame simulating a 250 m subsection of the scenery. Figure 6 shows three of the rendered frames, showing each one of the three objects in the scene illuminated by the emitter.

TABLE I
SIMULATION PARAMETERS.

parameter	value	units
Obj. 1 and 2 complex IOR	$1.4 + j1.0$	-
Target complex IOR	$10^4 + j10^4$	-
Pulse Repetition Frequency	1000	Hz
Sampling Frequency	60×10^6	Hz
Center Frequency	5.3×10^9	Hz
Chirp Duration	2.5×10^{-6}	s
FM rate	20×10^{12}	Hz/s

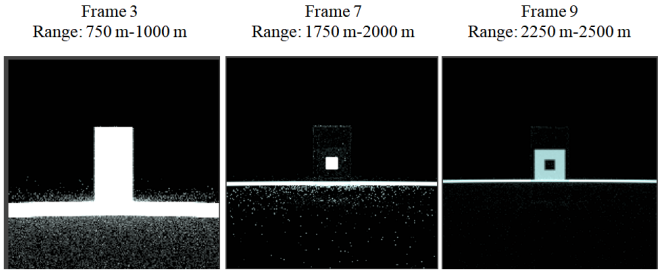

Fig. 6. Examples of frames rendered by Cycles.

After rendering, a Matlab script was used for reconstructing the signal captured by the camera. The reconstructed signal is shown in Fig. 7. Figure 8 shows the compressed signal after the matched filter. It can be seen on both figures that the objects in the scene have caused peaks in the received signal, at the same distances as they were placed in Blender, thus demonstrating that the simulator can measure target distances appropriately.

Figure 9 presents a range Doppler power spectrum of the signal. The stripe at 0 Hz corresponds to echoes of the static objects in the scene (the surface plane and the two larger objects), while the moving target appears as a single dot, at the left side of the diagram.

IV. Conclusion

This paper presented how the Blender Cycles path tracer can be used as a radar simulator, allowing the user to recreate complex scenery using the Blender modeling tools and then

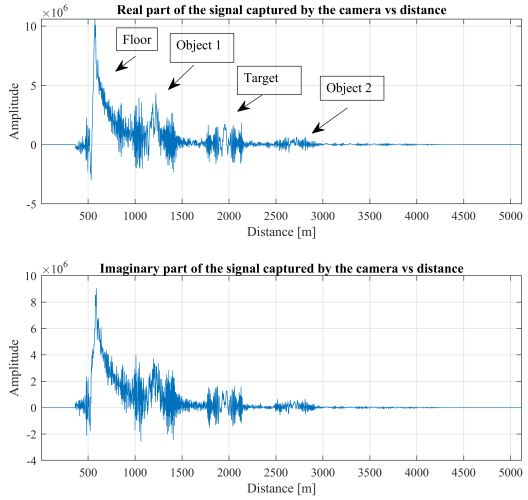

Fig. 7. Signal synthesized in the simulation.

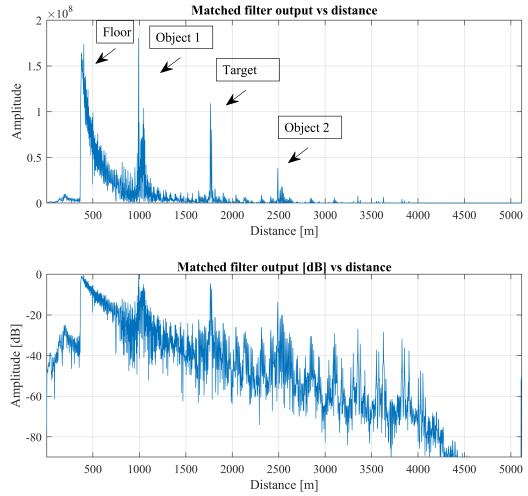

Fig. 8. Matched filter output for the synthesized signal.

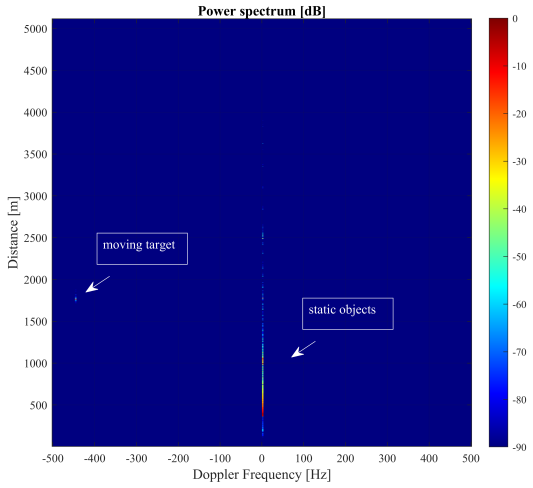

Fig. 9. Simulated radar power spectrum

extract information on how radar echoes would return in that scenario. The Python interface available in Blender could allow the implementation of complex tasks, for example, generating synthetic radar data to be used as input of machine learning and deep learning algorithms.

In its current form, the simulator possibly could be used for simulating SAR and ground-penetrating radar, as radar penetration and multipath propagation can be modeled in the simulation. Also, moving target indication (MTI) radar applications such as airspace surveillance can be simulated as well [14]. However, it is recommended for these applications to switch the Blender output files from Radiance HDR files to OpenEXR files, as those allow for higher precision.

Some constraints are imposed on the simulator in caustics and nonlinear modeling, which are current limitations of the Cycles engine.

This simulator is still under development. We expect to make the simulator (Blender files, scripts, and manuals) available for download at ITA's Electronic Warfare Laboratory webpage, free of charge, once the project is sufficiently mature.

ACKNOWLEDGMENTS

The authors are grateful for the financial support provided by the Brazilian Council for Scientific and Technological Development (CNPq) and the company IACIT. We also would like to express our gratitude to the Blender community, whose combined efforts shaped Blender into the powerful tool that it is today.

REFERENCES

[1] J. Agnarsson, *Simulation of a radar in Flames: a ray based radar model*. Master's Thesis, Uppsala university, 2013.

[2] C. Liebe, P. Combeau, A. Gaugue, Y. Pousset, L. Aveneau, R. Vauzelle, J. M. Ogier, "Ultra-wideband indoor channel modelling using ray-tracing software for through-the-wall imaging radar," *International Journal of Antennas and Propagation*, 2010.

[3] S. J. Auer, *3D Synthetic Aperture Radar Simulation for Interpreting Complex Urban Reflection Scenarios*. Doctoral Thesis, Technische Universitat München, 2011.

[4] M. Ouza, M. Ulrich, B. Yang, "A Simple Radar Simulation Tool for 3D Objects based on Blender," *The 18th International Radar Symposium IRS 2017*, pp. 1-10, 2017.

[5] C. Romero, *High Resolution Simulation of Synthetic Aperture Radar Imaging*. Master's Thesis, California Polytechnic State University, 2010.

[6] Blender Documentation Team, *Cycles Introduction*. Available in: https://docs.blender.org/manual/en/latest/render/cycles/introduction.html.

[7] Blender Foundation, *Cycles Features*. Available in: https://www.cycles-renderer.org/about/.

[8] R. Sedman, *How to calculate for every ray the total distance it has traveled from camera to emitter*. October 2017, available in: https://blender.stackexchange.com/a/91760.

[9] M. S. Dresselhaus, *Solid state physics part II optical properties of solids*. Lecture Notes, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[10] D. J. Segelstein, *The complex refractive index of water*. Doctoral Thesis, University of Missouri, 1981.

[11] H. J. Liebe, G. A. Hufford, T. Manabe, "A model for the complex permittivity of water at frequencies below 1 THz," *International Journal of Infrared and Millimeter Waves*, vol. 12, issue 7, pp. 659-675, 1991.

[12] V. M. Radivojević, S. Rupčić, M. Srnović, G. Benšić. "Measuring the Dielectric Constant of Paper Using a Parallel Plate Capacitor," *International journal of electrical and computer engineering systems*, vol. 9, issue 1, pp. 1-10, 2018.

[13] H. M. Jol. *Ground penetrating radar theory and applications*. Elsevier, 2008.

[14] R. F. Costa, D. S. Medeiros, R. B. Andrade, O. Saotome, R. Machado. "Aplicação de Simulador Radar Baseado em Blender Cycles em vigilância de espaço aéreo," *XXII Simpósio de Aplicações Operacionais em Áreas de Defesa*, pp. 11-16, 2020.