ISSN: 1983 7402 ITA, 29 a 30 SET 2020

Aplicação de Simulador Radar Baseado em *Blender Cycles* em Vigilância de Espaço Aéreo

Rômulo Fernandes da Costa, Diego da Silva de Medeiros, Raíssa Brasil Andrade,
Osamu Saotome, e Renato Machado
Laboratório de Guerra Eletrônica (LAB-GE), Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos/SP - Brasil

***Resumo*— Este artigo demonstra o uso de um simulador de radar, baseado no motor de renderização *Cycles* do *software open source* de modelagem 3D *Blender*, para uma aplicação de vigilância de espaço aéreo. Determinam-se a geometria e os materiais de cenário e alvos diretamente a partir do *Blender*, enquanto o motor de renderização *Cycles* é utilizado para simular a propagação de ondas no ambiente modelado. Os sinais de radar são extraídos a partir das imagens renderizadas pelo programa. Para demonstrar o funcionamento, é simulado um cenário simples contendo modelos de aeronave executando o sobrevoo de aquisição de dados de um radar de vigilância aérea. Os resultados indicam que a ferramenta pode ser utilizada em outras aplicações, tais como vigilância marítima ou sensoriamento remoto.**

***Palavras-Chave*— Simulador Radar, *Blender Cycles*, Vigilância Aérea.**

I. Introdução

Uma das maneiras mais eficientes, em termos de custo de projeto, de se avaliar técnicas de detecção de sinais radar é através de simulação. Embora seja possível modelar o eco de alvos de radar como alvos pontuais em um percurso simples considerando apenas ida e volta, em um cenário real, vários fenômenos físicos podem contribuir para o desvanecimento do sinal, tais como, reflexão, difração, absorção e espalhamento. De uma forma geral, pode-se dizer que a interação do sinal com o meio depende do comprimento de onda do sinal, da ordem do tamanho dos objetos e do tipo de material desses objetos.

Uma forma de realizar a simulação de um sinal radar é considerando equações diferenciais que modelam a propagação da onda eletromagnética e sua interação com o meio. Este método permite a simulação fidedigna da onda propagante e de sua interação com os objetos. No entanto, esse tipo de simulação é extremamente complexa e de alto custo em termos de tempo de processamento computacional [1].

Uma alternativa a este método é se utiilizar algoritmos de tracejamento de raios (*ray tracing*), que vem sendo cada vez mais utilizado pela academia em função de sua eficiência computacional e por estar disponível em pacotes de *softwares* livres [2]. Apenas para citar alguns exemplos, a técnica foi utilizada no contexto de radares de imageamento através de paredes com o uso do algoritmo *RAPSOR* [3], detecção de aeronaves com *Flames* [1] e para imageamento de ambientes urbanos com radares de abertura sintética (*SAR*) [4].

Por sua vez, muitos dos simuladores que empregam *ray tracing* não podem avaliar corretamente penetração e reflexões múltiplas, o que limita seu uso apenas à aplicações de linha de visada [5]. Outra dificuldade é criar geometrias corretas de terreno [4] e de alvos [6].

Considerando estes problemas, foi desenvolvida uma ferramenta de simulação de radar baseada no motor de renderização *Cycles*, que implementa um algoritmo de traçado de raios para o programa de modelagem 3D *Blender*.

O programa *Blender* é utilizado para fazer a modelagem do cenário e definir todos os materiais utilizados na simulação. Durante a renderização do cenário pelo *Cycles*, os canais *RGB* de cada raio são utilizados para gravar informações de fase e amplitude do sinal em cada *pixel* das imagens renderizadas. Por fim, uma terceira ferramenta computacional, tal como *MATLAB*, *Octave* ou *Python* pode então ser utilizada para reconstruir o sinal de radar a partir das imagens renderizadas em formato *HDR* ou *EXR*. Um fluxograma geral do processo de simulação é apresentado na Fig. 1.

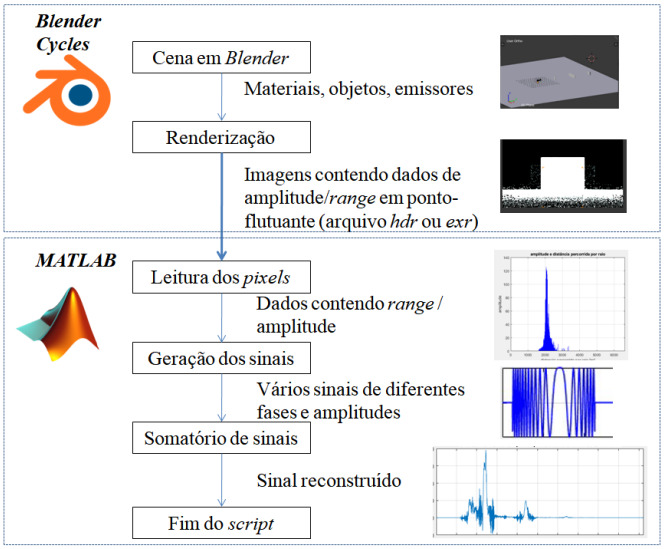

Fig. 1. Visão geral do simulador baseado em *Cycles*.

Neste trabalho, o funcionamento do simulador é demonstrado considerando-se um cenário de vigilância aérea. Nesta simulação, foram utilizados os parâmetros do radar SABER M60, um radar de banda L desenvolvido pelo Centro Tecnológico do Exército (CTEx) [7].

II. Descrição do Simulador

A. *Blender Cycles*

O *Blender* é um programa de código aberto voltado para modelagem 3D, animação e efeitos visuais, entre outras

R.F. Costa, rfcosta@ita.br; D.S. Medeiros, medeiros@ita.br; R.B. Andrade, raissa.andrade@ga.ita.br; O. Saotome, osamu@ita.br; R. Machado, rmachado@ita.br. Este trabalho foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e pela empresa IACIT.

aplicações. Este *software* utiliza um motor de renderização chamado *Cycles*, que simula como a luz deve se propagar em um cenário, baseando-se em princípios da óptica. Para isto, o *Cycles* emprega um algoritmo de *path tracing*, que faz a integração do percurso de uma grande quantidade de raios pelo cenário, considerando os fenômenos de reflexão, refração, absorção e difração para cada raio transmitido. Os materiais utilizados pelo *Cycles* podem ser configurados pela interface de nós disponibilizada pelo *Blender*, ou utilizando-se a linguagem *Open Shading Language (OSL)* [8].

Apesar do *Cycles* ser originalmente projetado para geração de gráficos tridimensionais, ele pode ser modificado para simular ondas de radar, tratando os emissores de luz como antenas transmissoras e a câmera do *Blender* como uma antena receptora [6]. Os canais de cor *RGB* são utilizados como referências para a estimação de amplitude e fase das ondas de radar.

B. Codificação dos canais RGB

Durante a renderização, a trajetória dos raios são obtidas partindo-se da câmera (receptor). Nesse caminho reverso, os raios vão realizando reflexões até alcançar um emissor, ou até que um número limite de reflexões seja alcançado. Escolhendo-se um número apropriado de reflexões como critério de parada, pode-se dizer que o fenômeno físico é representado de forma fidedigna. Adota-se essa trajetória reversa para simular a trajetória dos raios por questão de maior eficiência ao se computar a iluminação do cenário simulado.

Conforme mencionado anteriormente, o Blender simula na faixa de frequências ópticas. Portanto, utiliza-se três canais, i.e., *RGB*, para se representar o fenômeno de propagação nessa faixa de frequências. Para a modelagem, assume-se que a amplitude de cada canal *RGB* sofre um desvanecimento com decaimento de nível de potência proporcional ao quadrado da distância, conforme previsto nos modelos de perda por propagação [9]. Para se estimar a contribuição de cada canal, *R*, *G* e *B*, na composição do sinal recebido, ponderam-se os canais de acordo com propriedades do material refletor que sofreu interações com o sinal.

A Fig. 2 exemplifica como os raios são tracejados em um cenário renderizado pelo *Cycles*. Deve-se notar que a Fig. 2 omite os fenômenos de transmissão por clareza.

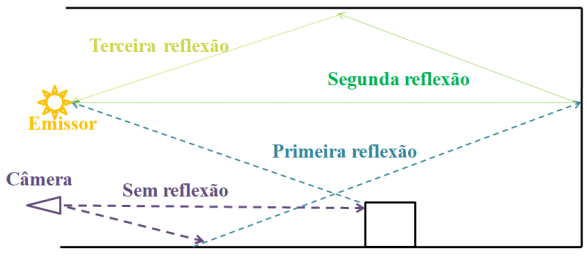

Fig. 2. Propagação de raios numa cena hipotética do *Cycles*.

No simulador proposto, adota-se o canal *B* como referencial de amplitude do sinal recebido, u_b. Considerando que a onda seja refletida n durante o seu percurso entre transmissor e receptor, u_b é obtido da seguinte forma:

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

em que k_0 é dependente do formato da antena e comprimento de onda, $k_1 \cdots k_n$ são constantes dependentes da geometria e dos materiais dos objetos refletores, e $r_1 \cdots r_n$ são as distâncias percorridas pelos n percursos entre transmissor e receptor. Pode-se verificar que para o caso de uma única reflexão, (1) toma o mesmo formato da equação de radar.

A fase do sinal pode ser derivada a partir de um somatório das distâncias percorridas pelos raios. No entanto, o *Cycles* não mantém registro da distância total percorrida pelo raio, tendo apenas acesso a distância percorrida pela última reflexão do raio.

Para obtermos acesso à distância total percorrida pelo raio, é necessário codificar a distância percorrida desde a última reflexão em um dos canais *RGB*. Isso é feito considerando-se um fator α_i, que decai exponencialmente com a distância r_i na i-ésima reflexão, e é dado por:

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

em que k_{dist} é uma constante. No simulador radar proposto, o fator α_i é aplicado à componente *R*.

Assim, com as n reflexões do raio pelo cenário, os fatores α_0, α_1 ...α_n são multiplicados para a ponderação dos múltiplos percursos associados ao canal *R*. A amplitude u_r desta componente é então dada por:

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

Assim, a razão u_r/u_b cai exponencialmente com a distância total percorrida pelo raio. Esta distância pode ser então extraída pela relação logarítmica:

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

C. Configuração do emissor

Como o Cycles foi projetado com o objetivo de se gerar uma imagem e não um sinal no tempo, objetos que estejam mais próximos da câmera podem esconder objetos que estejam mais afastados na mesma linha de visada, suprimindo a contribuição desses objetos na composição do sinal reconstruído.

Para evitar tal problema, o emissor é configurado para iluminar o cenário separadamente em vários *frames* (quadros), em passos sucessivos. Isto é feito limitando-se o comprimento $r_{emitter}$ dos raios do emissor a um intervalo dado por:

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

em que o n_{frame} é o índice do quadro atual e Δr_{frame} é o tamanho do passo por quadro adotado.

Com este procedimento, evita-se o problema causado pela oclusão, pois os objetos na mesma linha de visada da câmera serão renderizados em quadros diferentes, desde que o passo adotado seja suficientemente pequeno. Ao se reconstruir o sinal, a contribuição de todos os quadros renderizados devem ser somados para contabilizar as contribuições de reflexões de todo o cenário.

O procedimento é demonstrado na Fig. 3, onde um plano horizontal é iluminado por um emissor pontual, em passos sucessivos de 2,5 unidades (internas do *Blender*) por quadro. Devido aos limites em (5), um anel é projetado sobre o plano.

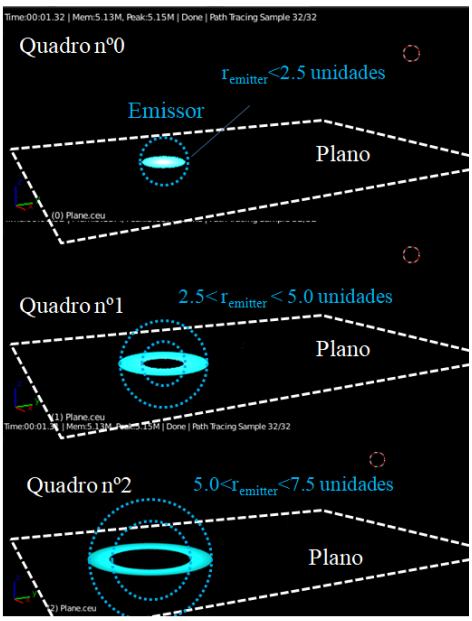

Fig. 3. Quadros iluminando diferentes regiões da cena separadamente.

D. Configuração dos materiais

Conforme descrito anteriormente, as componentes *RGB* de cada raio precisam codificar a distância percorrida a cada reflexão de material. Além desta codificação, é necessário que os materiais sejam configurados para apresentarem a mesmas propriedades de refletividade, refratividade e absorção que teriam na frequência de radar simulada. Essas propriedades são controladas por uma subrotina denominada *shader* (sombreador), que regula a forma como um material deve refletir ou propagar raios incidentes.

As propriedades de refletividade, refração, e absorção estão relacionadas através do índice de refração complexo [10]:

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

em que ϵ_r e μ_r são a permissividade elétrica complexa relativa e permeabilidade magnética complexa relativa, respectivamente. η_{ref} é a parte real do índice de refração e κ_{ext} é o coeficiente de extinção, que determina a atenuação da onda ao atravessar o material.

O fator de atenuação α_{trans} do raio ao atravessar o material deve ser modelada como:

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

em que λ_0 é o comprimento de onda do sinal de radar, e z é a distância percorrida dentro do material. Este fator foi implementado manualmente no *shader*.

A refletividade de uma superfície para um ângulo de incidência normal é dada por:

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

em que η_{ref1} e η_{ref2} são as partes reais dos índices de refração dos materiais envolvidos. Este valor de refletividade é calculado nativamente dentro do *Cycles* para todos os ângulos de incidência, assim não é necessário recalculá-lo no *shader*.

Um *shader* padrão foi desenvolvido para simular os materiais de radar, utilizando-se a interface de nós do *Blender*. O *shader* padrão implementa a codificação de distância, refletividade, refração e atenuação, servindo assim como base para todos os materiais do simulador. Um diagrama de blocos desse *shader* é apresentado na Fig. 4.

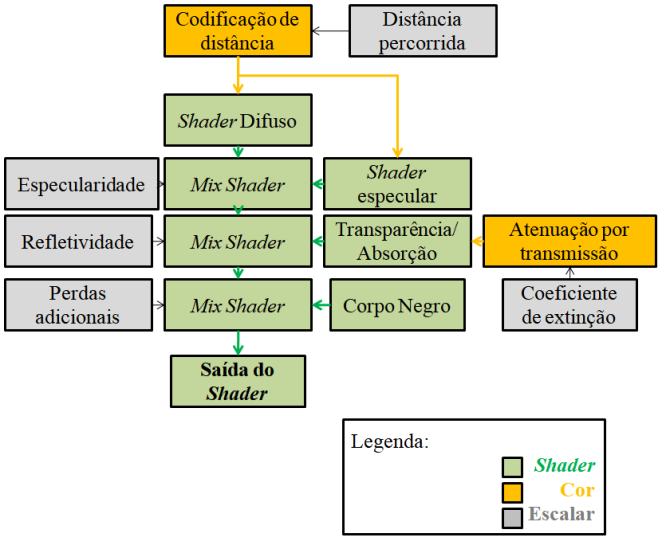

Fig. 4. Diagrama de blocos do *shader* padrão.

O *shader* admite como parâmetros de entrada o índice de refração η_{ref}, o coeficiente de extinção κ_{ext}, além de parâmetros para controle de especularidade, rugosidade e eventuais perdas adicionais do material. A interface de entrada do *shader* é apresentada na Fig. 5.

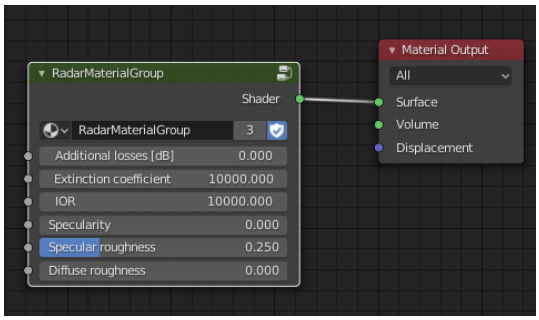

Fig. 5. Interface de entrada do *shader* padrão.

E. Renderização e reconstrução do sinal no tempo

O *Blender* pode salvar as imagens renderizadas em formatos que armazenem informação em ponto flutuante, tais como os formatos *HDR* e *OpenEXR*. Isso permite exportar as informações de fase e amplitude computadas pelo *Cycles* para serem processadas em um ambiente computacional tal como o *MATLAB*. Um *script* pode então reconstruir o sinal no tempo a partir dos dados contidos na imagem.

O simulador exporta os valores de amplitude e de distância percorrida em quatro arquivos *HDR*. Um par de arquivos armazena os valores de amplitude e de distância percorrida para as posições iniciais dos objetos conforme definido pelo usuário. O segundo par de arquivos armazena os valores de amplitude e de distância percorrida considerando um deslocamento de todos os objetos após um pequeno intervalo de tempo δt_{slow}, também definido pelo usuário. Este segundo

par de arquivos é utilizado para o cálculo de efeito Doppler no sinal.

O sinal radar no tempo pode ser reconstruído somando-se a contribuição gerada por cada *pixel* de cada quadro renderizado, assumindo que cada *pixel* contribui como um alvo pontual.

O sinal pontual de cada *pixel* pode ser escrito como :

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

em que A_0 é a amplitude do sinal, w é a função de janelamento, ϕ é a fase do sinal, τ é o tempo percorrido desde a emissão da onda a partir da antena (tempo rápido), e τ_0 é o atraso da onda. Tanto w quanto ϕ dependem do formato da forma de onda simulada, enquanto que as variáveis

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

e

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

podem ser obtidas diretamente do processo de renderização, e c é a velocidade da luz no vácuo.

A frequência Doppler é associada à variação de r_{sum} para cada *pixel* no intervalo δt_{slow}:

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

Para reconstruir o sinal, faz-se necessario computar uma soma ponderada das contribuições dos *pixels* para cada quadro renderizado. O sinal reconstruído no tempo é então dado por:

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

em que A_{pixel} é o peso de cada pixel no somatório.

Devido à projeção perspectiva na câmera, objetos mais distantes ocupam áreas menores na imagem final, sendo necessário compensar este efeito durante o somatório. Isto pode ser feito definindo-se o peso A_{pixel} como

$$A_{pixel} = k_{norm} r_{sum}^2, \tag{14}$$

em que r_{sum} é a distância associada a cada pixel e k_{norm} é uma constante para normalização do somatório.

III. Experimento

A. Simulação

Para demonstrar o funcionamento do simulador, foi construído um cenário simples no *Blender*, considerando-se uma aplicação de vigilância do espaço aéreo. Neste cenário, três aviões sobrevoam uma área plana, com diferentes velocidades. Para realizar esta simulação, projetou-se um modelo 3D de um avião de carga, baseado na geometria do modelo Embraer C-390. O modelo é mostrado na Fig. 6.

O primeiro avião foi posicionado a 20 km de distância do radar, se aproximando do radar a uma velocidade de 100 m/s. O segundo avião está localizado a 30 km do radar, se aproximando a uma velocidade de 50 m/s. O terceiro avião foi posicionado a 40 km de distância, se afastando do radar com velocidade de 100 m/s. O cenário é mostrado na Fig. 7, considerando a escala de unidade interna do Blender igual a 1 km.

A Tabela I lista os parâmetros principais da simulação. Os parâmetros de radar utilizados foram baseados no SABER M60 [7]. Assumiu-se a transmissão de pulsos modulados linearmente em frequência (*chirps*).

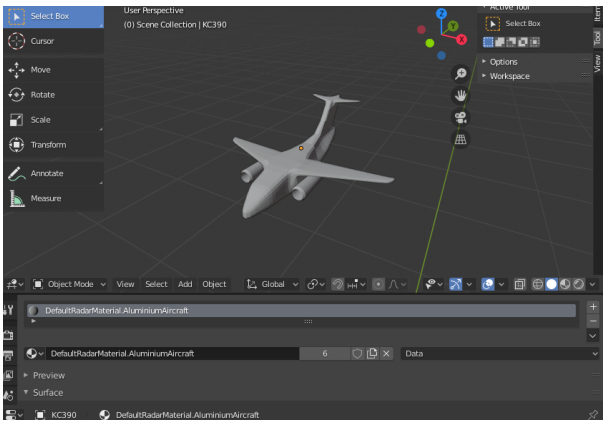

Fig. 6. Modelo de avião de carga modelado em *Blender*.

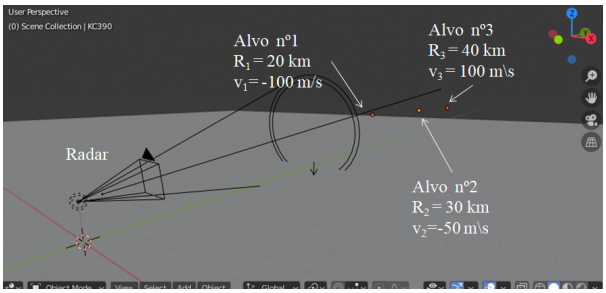

Fig. 7. Cenário construído para simulação.

TABELA I. Parâmetros de simulação

Parâmetro	***Valor***	***Unidades***
Indice de refração do cenário	$2,0 + j0,1$	-
Perdas adicionais do cenário	50	dB
Indice de refração do alvos	$4,0 + j6,0$	-
Perdas adicionais por alvo	0	dB
Frequência de repetição de pulso	2500	Hz
Frequência de amostragem	$2,14 \times 10^6$	Hz
Comprimento de onda	$2,5 \times 10^{-}1$	m
Duração de *chirp*	22×10^{-6}	s
Taxa de *chirp*	90×10^9	Hz/s

B. Resultados

Como o cenário foi modelado sem objetos oclusos, esse cenário foi renderizado em um único quadro. A imagem capturada pela câmera e a regiões iluminadas nos aviões são mostradas na Fig. 8.

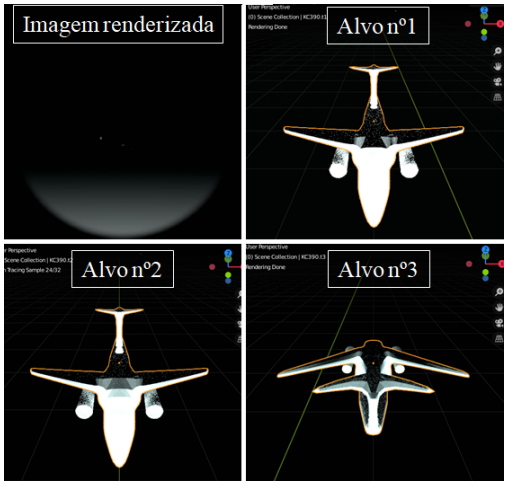

Fig. 8. Imagem renderizada pelo *Cycles*.

Após a renderização, um *script* em *MATLAB* foi utilizado para reconstruir o sinal no tempo capturado pela câmera. A Fig. 9 apresenta o sinal bruto capturado pela câmera. A Fig. 10 apresenta a amplitude do sinal após o emprego de filtro casado em função da distância, e a Fig. 11 mostra o espectro Doppler do sinal comprimido.

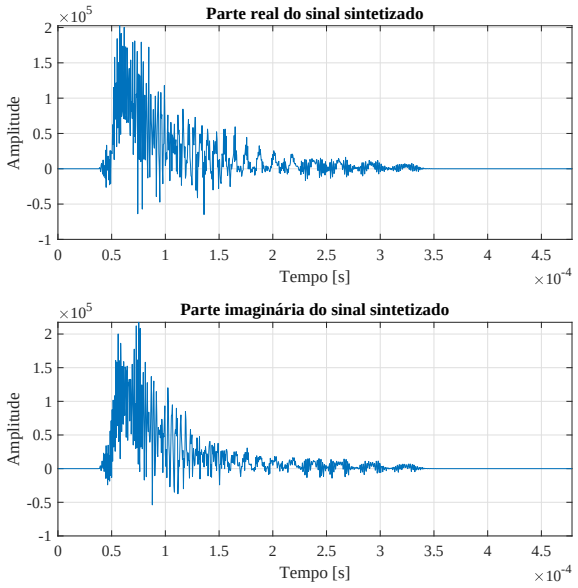

Fig. 9. Sinal sintetizado no tempo.

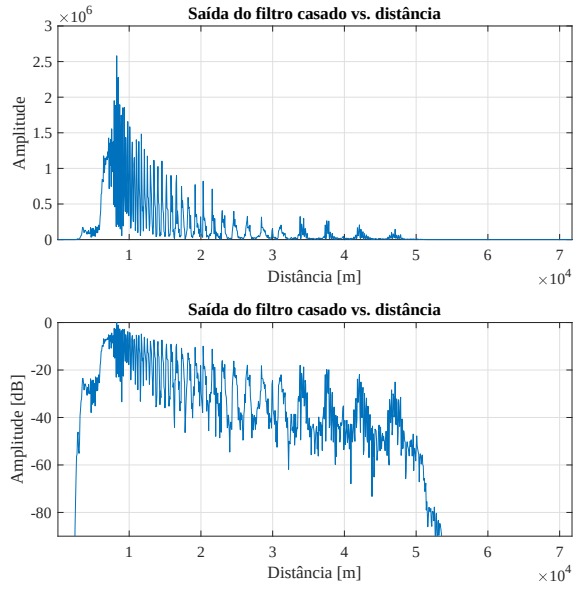

Fig. 10. Amplitude do sinal sintetizado na saída do filtro casado.

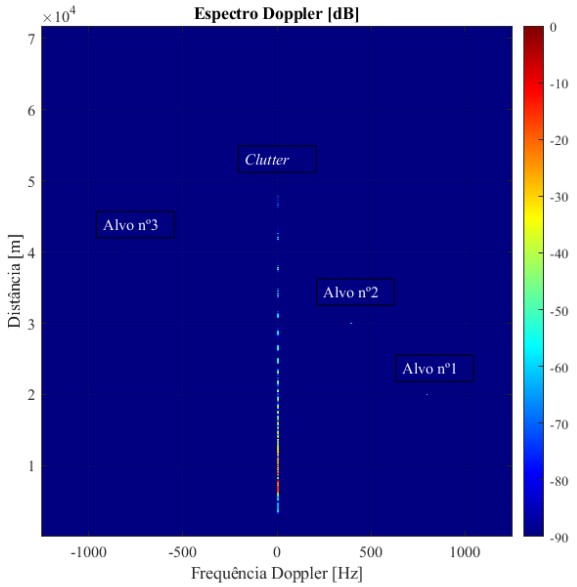

Fig. 11. Espectro Doppler do sinal comprimido.

Os alvos estão parcialmente ocultados pelo forte eco gerado pelo terreno. Ao se suprimir o *clutter* pela remoção do sinal de frequência Doppler zero, pode-se visualizar claramente os alvos. A saída do filtro casado após a supressão do *clutter* em função da distância é mostrada na Fig. 12.

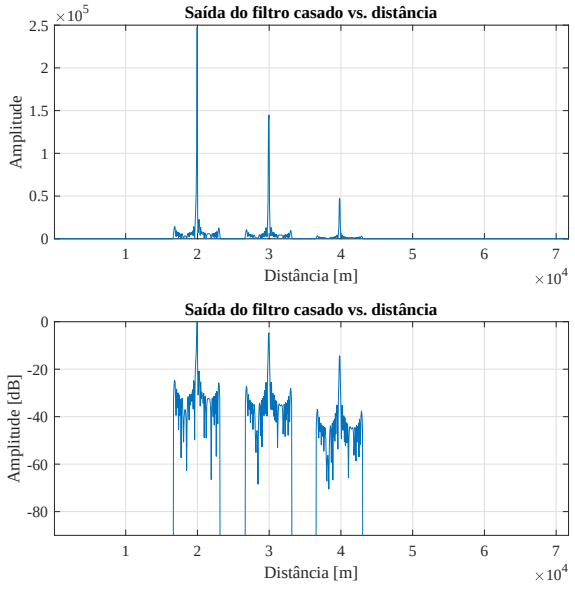

Fig. 12. Amplitude do sinal sintetizado na saída do filtro casado, com supressão de *clutter*.

O espectro Doppler deste sinal é mostrado na Fig. 13. Pode-se verificar na Fig. 13 que o sinal indica corretamente a posição dos alvos, com a velocidade de aproximação também podendo ser derivada a partir da frequência Doppler.

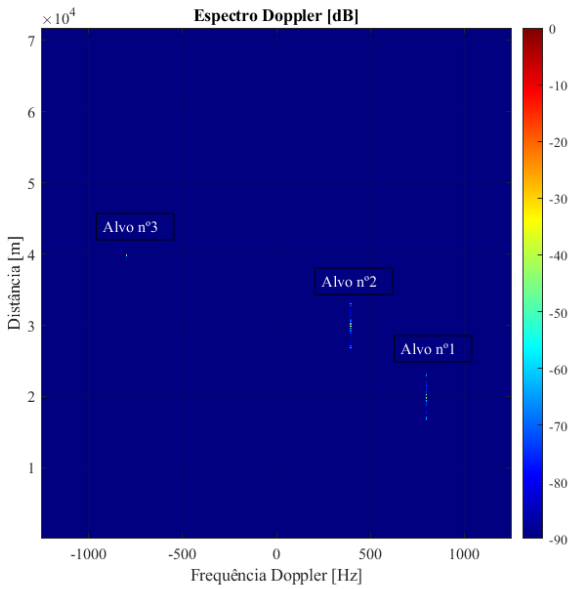

Fig. 13. Espectro Doppler do sinal comprimido, com supressão do *clutter*.

IV. Conclusão

Este trabalho apresentou a utilização do *Blender Cycles* como um simulador de radar em um cenário de vigilância de espaço aéreo. O algoritmo de renderização do *Cycles*, em conjunto com as ferramentas de modelagem do *Blender* podem permitir a simulação de outros cenários mais complexos.

A interface *Python* disponibilizada pelo *Blender* foi utilizada para simular o deslocamento dos alvos, porém poderia também ser aproveitada para automatizar a aquisição de dados em simulações complexas, como por exemplo, na extração da assinatura de radar de um modelo em vários ângulos possíveis, a fim de se gerar um banco de dados para treinamento de algoritmos de detecção.

O simulador ainda encontra-se em desenvolvimento. Os autores pretendem disponibilizar o simulador abertamente para o público acadêmico assim que o projeto atingir maturidade.

Agradecimentos

Este trabalho foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e pela empresa IACIT, e também recebeu apoio institucional do Laboratório de Guerra Eletrônica (LAB-GE), do Instituto Tecnológico de Aeronáutica (ITA).

Referências

[1] J. Agnarsson, "Simulation of a radar in flames: a ray based radar model," Master's thesis, Uppsala University, 2013.

[2] N. Douchin, C. Ruiz, J. Israel, and H.-J. Mametsa, "SE-workbench-RF: Performant and high-fidelity raw data generation for various radar applications," in *20th International Radar Symposium (IRS)*. IEEE, 2019, pp. 1–10.

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

[4] S. J. Auer, "3D synthetic aperture radar simulation for interpreting complex urban reflection scenarios," Ph.D. dissertation, Technische Universität München, 2011.

[5] M. Ouza, M. Ulrich, and B. Yang, "A simple radar simulation tool for 3D objects based on blender," in *18th International Radar Symposium (IRS)*. IEEE, 2017, pp. 1–10.

[6] C. G. Romero, "High resolution simulation of synthetic aperture radar imaging," Master's thesis, California Polytechnic State University, 2010.

[7] B. de Carvalho, R. Jorge, J. da Silva, F. de Bastos, H. Costa, B. Pimento, A. Medella, B. de Carvalho, M. Pralon, and V. Santa Rita, "Desdobramentos tecnológicos no desenvolvimento do radar SABER M60," in *X Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE)*. ITA, São José dos Campos/SP, 2008.

[8] B. O. Community, *Blender - A 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[9] S. Y. Seidel and T. S. Rappaport, "914 mhz path loss prediction models for indoor wireless communications in multifloored buildings," *IEEE transactions on Antennas and Propagation*, vol. 40, no. 2, pp. 207–217, 1992.

[10] M. Dresselhaus, "Solid state physics part II optical properties of solids," *Lecture Notes (Massachusetts Institute of Technology, Cambridge, MA)*, vol. 17, 2001.